Завдання

IV етапу Всеукраїнських учнівських олімпіад з навчальних предметів

2023/2024 навчальний рік

ІНФОРМАТИКА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Київ — 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
НАЦІОНАЛЬНИЙ ЦЕНТР «МАЛА АКАДЕМІЯ НАУК УКРАЇНИ»

Завдання

IV етапу Всеукраїнських
учнівських олімпіад
з навчальних предметів

2023/2024 навчальний рік

ІНФОРМАТИКА

**ІНФОРМАЦІЙНІ
ТЕХНОЛОГІЇ**

Практикум

Київ
Національний центр
«Мала академія наук України»
2024

УДК 373
З-13

Автори:
І. Г. Бондік, Є. А. Варзар, Ю. В. Горошко, д. пед. наук, проф.*, В. В. Гриненко, Г. Ю. Громко, К. І. Денисов, М. М. Кузічев, В. А. Кушнір, Т. Г. Ніколаєв, К. Р. Савчук, А. С. Столітній, О. В. Шаравара*

Відповідальна за випуск:
А. І. Грітчина, канд. пед. наук

Упорядники:
О. В. Роговець, канд. пед. наук, *Д. О. Ужвенко*

Рекомендовано для використання в освітньому процесі рішенням науково-методичної ради Національного центру «Мала академія наук України» (протокол № 3 від 26.06.2024)

Завдання IV етапу Всеукраїнських учнівських олімпіад з навчальних
З-13 предметів. 2023/2024 навчальний рік. Інформатика, інформаційні технології : практикум / І. Г. Бондік, Є. А. Варзар, Ю. В. Горошко ; відп. за вип. А. І. Грітчина ; упоряд.: О. В. Роговець, Д. О. Ужвенко. – Київ : Національний центр «Мала академія наук України», 2024. – 96 с.

Практикум, створений для ефективної підготовки учнів до участі в олімпіадах, містить завдання IV етапу Всеукраїнських учнівських олімпіад з інформатики та інформаційних технологій у 2023/2024 навчальному році. Зміст завдань спрямований на перевірку рівня володіння основними знаннями з інформатики та інформаційних технологій, здатності їх практичного застосування, формування в учнів логічного, критичного і креативного мислення в процесі розв'язування задач.

Збірник адресований учням 10–11 класів, педагогічним працівникам, які супроводжують їх під час підготовки до участі в інтелектуальних змаганнях.

УДК 373

ЗМІСТ

ВСТУП

В епоху цифрових технологій, коли інформація стає цінним ресурсом, знання з інформатики й інформаційних технологій набувають особливої актуальності. Уміння безпечно користуватися інтернетом, захищати свої персональні дані та розпізнавати фейки стає життєво необхідним у сучасному світі. Тож знання з інформатики є ключем до успішної кар'єри та повноцінного життя.

У практикумі представлені завдання фінального етапу Всеукраїнських учнівських олімпіад з інформатики й інформаційних технологій, що проводилися для учнів 10–11 класів закладів загальної середньої освіти у 2023/2024 навчальному році. Вивчення інформатики в старших класах дає змогу закріпити здобуті раніше знання та підготуватися до подальшого навчання у закладах професійно-технічної та вищої освіти.

Здобуття знань з інформатики – це інвестиція в майбутнє, ключ до успіху в сучасному світі, де технології розвиваються з неймовірною швидкістю. Розв'язуючи алгоритмічні задачі, учні розвивають логічне мислення, вчаться аналізувати інформацію, виявляти закономірності та будувати логічні ланцюжки, тобто вдосконалюють навички, які стануть у пригоді в будь-якій сфері діяльності.

Участь в олімпіадному русі відіграє важливу роль у формуванні всебічно розвиненої особистості. Це не тільки допомагає учням засвоїти навчальний матеріал, але й сприяє стресостійкості, формуванню навичок практичного застосування своїх знань, творчого підходу до виконання поставлених завдань тощо. Тому включення інтелектуальних змагань в освітній процес є необхідною умовою для забезпечення якісної освіти.

Сподіваємося, що видання сприятиме заохоченню учнів до вивчення цих предметів, розвитку їхніх талантів та здібностей, допоможе визначитися зі своєю майбутньою професією та зрозуміти, чи хочуть пов'язати своє життя зі сферою інформаційних технологій.

Анна Грітчина,
заступниця директора
з методичної роботи НЦ «МАНУ»,
кандидатка педагогічних наук

ІНФОРМАТИКА

Умови задач

I тур

Задача A. Точки на прямій

Інтерактивна задача

Є n точок на числовій прямій, що мають цілі координати $x_1, x_2, ..., x_n$. Гарантується, що $1 \le x_i \le n$ для $1 \le i \le n$.

Вважається, що точка x_k лежить між точками x_i та x_j тоді й тільки тоді, коли вона належить відрізку, побудованому на точках x_i та x_j. Формально точка x_k розташована між точками x_i та x_j тоді й тільки тоді, коли $x_i \le x_k \le x_j$ або $x_j \le x_k \le x_i$.

Вам потрібно знайти будь-які два індекси i та j, щоб всі n точок лежали між точками x_i та x_j.

Ви можете користуватися таким запитом: вибрати три індекси (i, j, k) та дізнатись, чи розташована точка x_k між точками x_i та x_j.

Вам дозволяється використати не більше ніж 22 222 запити.

Гарантується, що координати точок зафіксовані перед початком взаємодії. Іншими словами, *інтерактор не є адаптивним*.

Формат вхідних даних

У першому рядку задано одне ціле число n $(3 \le n \le 2 \cdot 10^4)$ – кількість точок.

Протокол взаємодії

Для виконання запиту потрібно вивести «? i j k» $(1 \le i, j, k \le n)$, після чого вивести символ кінця рядка та виконати операцію flush.

У відповідь на запит програма журі виведе одне ціле число f $(f \in \{0, 1\})$. Якщо $f = 1$, то точка x_k розташована між точками x_i та x_j; якщо $f = 0$, то точка x_k не розташована між ними.

Якщо запит невалідний (тобто перевищено максимальну кількість запитів або параметри запиту є невалідними), то програма журі виведе –1 та припинить роботу. У такому випадку завершіть роботу програми, щоб отримати вердикт «Неправильна відповідь».

Коли ви будете готові надати відповідь, виведіть її у форматі «! i j» ($1 \le i, j \le n$), де i та j – шукані індекси точок. Після цього завершіть роботу програми.

Операція flush виконується таким чином:

- fflush(stdout) або cout.flush() в C++;
- System.out.flush() в Java;
- flush(output) в Pascal;
- sys.stdout.flush() в Python.

Система оцінювання

1. (17 балів): $n \le 20$;
2. (16 балів): $n \le 100$;
3. (30 балів): $n \le 10\,000$;
4. (23 бали): $n \le 20\,000$, $x_i \le 2$;
5. (10 балів): $n \le 12\,000$;
6. (4 бали): без додаткових обмежень.

Приклад

standart input	standart output
4 1 1	? 1 4 2 ? 1 4 3 ! 1 4

Примітка. У прикладі точки мають координати $x = [1, 2, 3, 4]$.

Задача В. Подарунок Леді

На свій день народження Леді отримала подарунок – мережу. Мережа містить n вузлів, пронумерованих цілими числами від 1 до n. На кожному вузлі записана певна літера, для вузла з номером i позначимо цю літеру як s_i.

Між деякими парами вузлів є односторонні зв'язки. З кожного вузла виходить рівно один односторонній зв'язок. Нехай для вузла з номером i такий зв'язок веде у вузол з номером x_i. Зауважте, що x_i може бути рівним i – у такому випадку вважається, що зв'язок з вузла з номером i веде в цей самий вузол.

Нехай $p_{i,0} = i$ та $p_{i,k} = p_{x_{i,k-1}}$. Тобто $p_{i,k}$ – номер вузла, у якому опиниться фішка, якщо її помістити у вузол з номером i та k разів перенести по зв'язку з поточного вузла.

Леді створила матрицю a розміром $n \times (3 \cdot n)$, де $a_{i,j} = s_{p_{i,j}-1}$. Тобто i-й рядок матриці a – це послідовність з $3 \cdot n$ літер, де перша літера дорівнює s_i, друга літера – s_{x_i}, третя – $s_{x_{x_i}}$ і так далі... Леді повідомила деякі рядки матриці a та просить вас побудувати будь-яку мережу, що відповідає відомим рядкам матриці a.

Формат вхідних даних

У першому рядку задано одне ціле число n ($1 \le n \le 2 \cdot 10^3$) – кількість вузлів мережі.

У наступних n рядках задано опис матриці a. У кожному з рядків задано $3 \cdot n$ маленьких літер латинського алфавіту, що позначають відповідний рядок матриці a, або один символ «?», якщо відповідний рядок Леді не повідомила.

Гарантується, що існує хоча б одна мережа, яка відповідає заданим умовам.

Формат вихідних даних

У першому рядку виведіть рядок s з n маленьких літер латинського алфавіту – літери записані на вузлах мережі.

У другому рядку виведіть n цілих чисел $x_1, x_2, ..., x_n$ ($1 \le x_i \le n$) – номери вузлів, у які ведуть зв'язки з відповідних вузлів.

Матриця, яка відповідає цій мережі, має бути рівною матриці a в усіх рядках, які повідомила Леді.

Якщо існує кілька правильних відповідей, дозволяється вивести будь-яку з них.

Система оцінювання

Нехай q – кількість рядків, які Леді не повідомила.

Назвемо мережу набором пар та одиничних вузлів, якщо мережу можна розбити на вузли, з яких зв'язок веде у самих себе (тобто $x_v = v$), та на пари таких вузлів (a, b), що $x_a = b$ та $x_b = a$.

Назвемо мережу набором зірок, якщо мережу можна розбити на окремі зірки, кожна з яких складається з головного вузла v, з якого зв'язок веде у самого себе (тобто $x_v = v$), та такого набору другорядних вузлів $y = \{y_1, y_2, ..., y_k\}$, що $x_{y_i} = v$ для $1 \le i \le k$. Зауважте, що зірки в мережі можуть мати різні розміри та складатися лише з одного головного вузла.

Назвемо мережу деревом з коренем у вузлі v, якщо з вузла v зв'язок веде у самого себе (тобто $x_v = v$) та для кожного іншого вузла можливо потрапити до вузла v, використовуючи зв'язки мережі (тобто для кожного $1 \le i \le n$ існує таке k, що $p_{i,k} = v$).

Назвемо мережу циклом, якщо, почавши у будь-якому вузлі, можливо потрапити до будь-якого іншого вузла, використовуючи зв'язки мережі (тобто для усіх $1 \le i, j \le n$ існує таке k, що $p_{i,k} = j$).

1. (10 балів): $n \le 5$, $q = 0$;
2. (6 балів): $n \le 300$, $q = 0$, $x_{x_i} = i$ для $1 \le i \le n$ (мережа є набором пар та одиничних вузлів);
3. (6 балів): $n \le 300$, $q = 0$, $x_{x_i} = x_i$ для $1 \le i \le n$ (мережа є набором зірок);
4. (9 балів): $n \le 300$, $q = 0$, $x_1 = 1$ та $x_i < i$ для $2 \le i \le n$ (мережа є деревом з коренем у вузлі 1);
5. (9 балів): $n \le 300$, $q = 0$, для усіх $1 \le i, j \le n$ існує таке k, що $p_{i,k} = j$ (мережа є циклом);
6. (13 балів): $n \le 300$, $q = 0$;
7. (25 балів): $n \le 300$;
8. (10 балів): $n \le 2 \cdot 10^3$, $q = 0$;
9. (12 балів): без додаткових обмежень.

Приклади

standart input	standart output
4 abaaaaaaaaaa baaaaaaaaaaa aaaaaaaaaaaa ccccccccccc	abac 2 3 3 4
3 axaxaxax xxxxxxxxx ?	axx 3 2 1

Примітка. У першому прикладі $x_1 = 2$ і $x_2 = 3$, тому $p_{1,0} = 1$, $p_{1,1} = 2$, $p_{1,2} = 3$. Через те що $x_3 = 3$, усі $p_{1,k}$ для $k \ge 3$ теж дорівнюють 3. Відповідно друга літера першого рядка дорівнює s_2, а третя – s_3.

Нижче зображені мережі з прикладів. Числа в кутку позначають номери вузлів, літери – записані на відповідних вузлах значення, а стрілки – односторонні зв'язки.

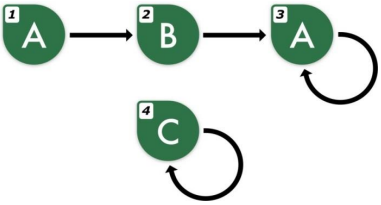


Рис. 1. Мережа з першого прикладу

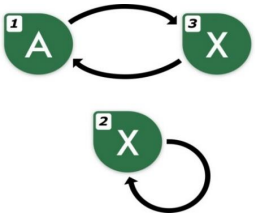


Рис. 2. Мережа з другого прикладу

Задача C. Запити красот підмасивів

Назвемо *вагою* масиву цілих чисел b довжини m модуль суми його елементів, тобто $|b_1 + b_2 + ... + b_m|$.

Визначимо *красу* розбиття масиву c на кілька підмасивів як мінімальну *вагу* серед *ваг* підмасивів. Формально *красою* розбиття масиву c на k підмасивів $[c_1, ..., c_{p1}]$, $[c_{p1} + 1, ..., c_{p2}]$, ..., $[c_{pk-1} + 1, ..., c_{pk}]$, де $1 \le p_1 < p_2 < ... < p_{k-1} < p_k = n$, є значення $\min(|c_1 + ... + c_{p1}|, |c_{p1} + 1 + ... + c_{p2}|, ..., |c_{pk-1} + 1 + ... + c_{pk}|)$. Наприклад, розбиття масиву [3, –6, 4, 5, –8] на підмасиви [3, –6], [4], [5, –8] має *красу* $\min(|3 + (–6)|, |4|, |5 + (–8)|) = \min(3, 4, 3) = 3$.

Визначимо *красу* масиву c як максимальну *красу* серед усіх можливих його розбиттів на підмасиви.

Задано масив цілих чисел a довжини n.

Необхідно виконати q запитів. Запити бувають двох типів:

1) знайти *красу* масиву, що складається з елементів $[a_l, a_{l+1}, ..., a_r]$, де (l, r) – параметри запиту;

2) замінити елемент a_x на v, де (x, v) – параметри запиту.

Формат вхідних даних

У першому рядку задано два цілі числа n, q ($1 \le n, q \le 10^6$) – довжина масиву a та кількість запитів відповідно.

У другому рядку задано n цілих чисел a_1, a_2, ..., a_n ($-10^9 \le a_i \le 10^9$) – елементи масиву a.

У наступних q рядках задано по три цілі числа. Перше з чисел $type_i$ ($1 \le type_i \le 2$) позначає тип запиту. Запити першого типу задані у форматі «1 l r» ($1 \le l \le r \le n$), а запити другого типу задані у форматі «2 x v» ($1 \le x \le n$, $-10^9 \le v \le 10^9$).

Формат вихідних даних

Для кожного запиту першого типу в окремому рядку виведіть одне ціле число – *красу* відповідного масиву.

Система оцінювання

1. (4 бали): $type_i = 1$ для $1 \le i \le q$; $a_i > 0$ для $1 \le i \le n$;
2. (14 балів): $type_i = 1$ для $1 \le i \le q$; $n, q \le 1000$;
3. (10 балів): $type_i = 1$ для $1 \le i \le q$; $n, q \le 2 \cdot 10^5$, для кожного запиту існує оптимальне розбиття на не більше ніж 2 підмасиви;
4. (10 балів): $type_i = 1$ для $1 \le i \le q$; $q \le n \le 2 \cdot 10^5$, $l_i = 1, r_i = i$ для $1 \le i \le q$;
5. (11 балів): $type_i = 1$ для $1 \le i \le q$; $n, q \le 2 \cdot 10^5$, $-5 \le \sum_{j=1}^{i} a_j \le 5$ для $1 \le i \le n$;
6. (18 балів): $type_i = 1$ для $1 \le i \le q$; $n, q \le 2 \cdot 10^5$;
7. (9 балів): $type_i = 1$ для $1 \le i \le q$;
8. (16 балів): $n, q \le 2 \cdot 10^5$;
9. (8 балів): без додаткових обмежень.

Приклади

standart input	standart output
6 4 1 -3 4 2 -5 6 1 1 6 1 2 3 1 2 5 1 1 1	5 3 3 1
5 6 1 -2 3 -4 5 1 1 4 1 2 3 2 3 -6 1 2 4 2 4 2 1 1 5	2 2 12 7

Примітка. У першому прикладі для третього запиту максимальна краса розбиття масиву [–3, 4, 2, –5] досягається при розбитті на підмасиви [–3], [4, 2], [–5].

У другому прикладі для першого запиту максимальна краса розбиття масиву [1, –2, 3, –4] досягається при розбитті на підмасиви [1, –2, 3], [–4].

Задача D. AND масив

Задано ціле число b та масив невід'ємних цілих чисел a_1, a_2, ..., a_n. Всі елементи масиву a менші за 2^b.

Визначимо $f(s, p)$ $(1 \le s \le n, 0 \le p < b)$ як результат виконання такого псевдокоду:

```
res = 0
x = power (2, p)
for i = s  to  n:
  if ((x AND a[i]) = = 0):
  x = (x OR a[i])
  res = res + i
повернути res
```

Тут «power (2, p)» позначає 2^p, «AND» позначає операцію *побітового І*, а «OR» позначає операцію *побітового АБО*.

Побітове І невід'ємних цілих чисел a та b дорівнює невід'ємному цілому числу, у якого у двійковому записі на певній позиції розташована одиниця тоді і тільки тоді, коли у двійкових записах a та b на цій позиції розташовані одиниці. Наприклад, 3_{10} AND $5_{10} = 0011_2$ AND $0101_2 = 0001_2 = 1_{10}$.

Побітове АБО невід'ємних цілих чисел a та b дорівнює невід'ємному цілому числу, у якого у двійковому записі на певній позиції стоїть нуль тоді й тільки тоді, коли у двійкових записах a та b на цій позиції розташовані нулі. Наприклад, 3_{10} OR $5_{10} = 0011_2$ OR $0101_2 = 0111_2 = 7_{10}$.

Для кожного i від 1 до n знайдіть $f(i, 0) + f(i, 1) + ... + f(i, b - 1)$.

Формат вхідних даних

У першому рядку задано два цілі числа n та b $(1 \le n \le 10^5, 1 \le b \le 20)$ – довжина масиву a та обмеження на елементи масиву відповідно.

У другому рядку задано n цілих чисел a_1, a_2, ..., a_n $(0 \le a_i < 2^b)$ – елементи масиву a.

Формат вихідних даних

Виведіть n цілих чисел – шукані значення.

Система оцінювання

1. (10 балів): $n \le 2000$;
2. (10 балів): $a_i = 2^k$, де k – ціле число;
3. (15 балів): $b \le 8$;
4. (15 балів): $b \le 12$;
5. (25 балів): $b \le 16$;
6. (25 балів): без додаткових обмежень.

Приклади

standart input	standart output
5 3 0 2 1 3 4	23 20 16 14 10
3 2 1 3 2	4 3 3

Примітка. У першому прикладі $f(1, 0) = 1 + 2 + 5 = 8$, $f(1, 1) = 1 + 3 + 5 = 9$, $f(1, 2) = 1 + 2 + 3 = 6$, а перше з шуканих значень рівне $8 + 9 + 6 = 23$.

II тур

Задача А. Кольорова таблиця

Задано таблицю a розміру $n \times m$, що складається із символів R, G, B.

Також задані цілі числа c ($2 \le c \le 3$) та q, де c – це кількість різних символів, що можуть зустрічатись у таблиці. Якщо c дорівнює 2, то доступні лише символи R та G; якщо ж c дорівнює 3, то доступні символи R, G, B.

Вам потрібно змінити значення не більше ніж q елементів таблиці так, щоб не існувало пари сусідніх по стороні клітинок, які мають однакове значення. Зауважте, якщо $c = 2$, то забороняється використовувати символ B при зміні значень клітинок таблиці.

Гарантується, що при заданих обмеженнях існує спосіб змінити не більше ніж q елементів таблиці так, щоб не існувало пари сусідніх по стороні клітинок, які мають однакове значення.

Зауважте, що в задачі немає блоку «без додаткових обмежень».

Формат вхідних даних

У першому рядку задано два цілі числа n та m ($1 \le n, m \le 100$) – кількість рядків і стовпців таблиці a відповідно.

У другому рядку задано два цілі числа c ($2 \le c \le 3$) та q, які позначають кількість доступних символів і кількість дозволених змін у таблиці відповідно.

У наступних n рядках задано по m символів – елементи таблиці a. Якщо $c = 2$, то $a_{ij} \in R, G$. Якщо $c = 3$, то $a_{ij} \in R, G, B$.

Формат вихідних даних

Виведіть n рядків по m символів кожен, що описують таблицю після виконаних змін. Якщо існує кілька правильних відповідей, дозволяється вивести будь-яку з них.

Система оцінювання

1. (7 балів): $n = 1$, $c = 3$, $q = \left\lceil \frac{n \cdot m}{2} \right\rceil$;
2. (7 балів): $n = 1$, $c = 2$, $q = \left\lceil \frac{n \cdot m}{2} \right\rceil$;
3. (3 бали): $c = 3$, $q = n \cdot m$;
4. (7 балів): всі рядки таблиці a однакові, $a[1][j] \neq a[1][j + 1]$ (для $1 \le j < m$), $c = 3$, $q = \left\lceil \frac{n \cdot m}{2} \right\rceil$;

5. (7 балів): всі рядки таблиці a однакові, $c = 3$, $q = \left[\frac{n \cdot m}{2}\right]$;

6. (13 балів): $c = 3$, $q = \left[\frac{2 \cdot n \cdot m}{3}\right]$;

7. (19 балів): $c = 3$, $n \le 5$, $m \le 100$, $q = \left[\frac{n \cdot m}{2}\right]$;

8. (17 балів): $c = 2$, $q = \left[\frac{n \cdot m}{2}\right]$;

9. (20 балів): $c = 3$, $q = \left[\frac{n \cdot m}{2}\right]$.

Приклади

standart input	standart output
3 3 3 4 RRR RRR RRR	RGR GRG RGR
3 2 2 3 RG GG GR	RG GR RG

Задача В. Футбол

У футбольній команді n гравців, пронумерованих цілими числами від 1 до n. Рівень гри гравця з номером i описується цілим числом c_i.

Гравці вишикувались по колу в такому порядку, що наступним з правого боку для гравця з номером i є гравець з номером $i + 1$ (для $1 \le i < n$), а для гравця з номером n наступним є гравець з номером 1.

Визначимо силу ігрової комбінації, яка характеризується масивом цілих чисел $k = [k_0, k_1, k_2, \ldots, k_{m-1}]$, таким чином:

- початково м'яч перебував у гравця з номером 1;
- гравці по черзі передають м'яч нескінченно довго: виконуючи передачу з номером i, гравець, який наразі контролює м'яч, передає його гравцю, що перебуває на x позицій правіше по колу, де $x = k_{((i-1) \bmod m)}$;

• силою ігрової комбінації вважається мінімальна сила гри серед сил всіх гравців, які володіли м'ячем у деякий момент описаного процесу.

Задано масив цілих чисел $a_0, a_1, \ldots, a_{q-1}$. Для кожного i від 0 до $(q-1)$ знайдіть силу ігрової комбінації, яка характеризується масивом $[a_0, a_1, \ldots, a_i]$.

Формат вхідних даних

У першому рядку задано два цілі числа n та q $(1 \le n, q \le 3 \cdot 10^5)$ – кількість гравців та довжина масиву a.

У другому рядку задано n цілих чисел $c_1, c_2, \ldots, c_n$ $(1 \le c_i \le n)$ – рівні гри гравців.

У третьому рядку задано q цілих чисел $a_0, a_1, \ldots, a_{q-1}$ $(1 \le a_i \le n-1)$ – елементи масиву a.

Формат вихідних даних

Виведіть q цілих чисел – шукані значення сил ігрових комбінацій.

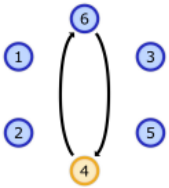
Система оцінювання

1. (10 балів): $n, q \le 100$;
2. (4 бали): усі значення a_i однакові;
3. (11 балів): n – просте число;
4. (12 балів): $n, q \le 1000$;
5. (16 балів): $n, q \le 1{,}5 \cdot 10^5, n = 2^k$ для деякого цілого k;
6. (25 балів): $n, q \le 10^5$;
7. (22 бали): без додаткових обмежень.

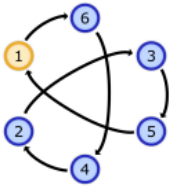
Приклад

standart input	standart output
6 3 6 3 5 4 2 1 3 1 2	4 1 2

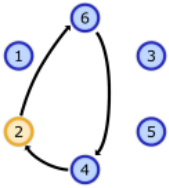
Примітка. У прикладі передачі м'яча для ігрових комбінацій виглядають таким чином:



$k = [3]$



$k = [3, 1]$



$k = [3, 1, 2]$

Рис. 1–3. Приклади передачі м'яча для ігрових комбінацій

Задача C. Герої та монстри

Існує n героїв та n монстрів. Герої та монстри пронумеровані цілими числами від 1 до n. Сила i-го героя дорівнює a_i, а сила i-го монстра рівна b_i. Гарантується, що всі значення $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$ *попарно різні*.

Всього відбудеться n боїв. У кожному бою візьме участь рівно один герой та рівно один монстр, при чому кожен герой і кожен монстр візьмуть участь рівно в одному бою. Нехай у бою беруть участь герой з номером i та монстр з номером j. Якщо $a_i > b_j$, то герой з номером i буде щасливим, а інакше він засмутиться.

Визначимо ans_k – кількість таких різних множин героїв S розміру k, що існує розподіл на бої, при якому будуть щасливими усі герої з S та засмутяться усі інші герої.

Задано q запитів виду l, r. Для кожного запиту знайдіть $(\sum_{i=l}^{r} ans_i) \bmod 998244353$.

Формат вхідних даних

У першому рядку задано одне ціле число n $(1 \le n \le 5 \cdot 10^3)$ – кількість боїв, що відбудуться.

У другому рядку задано n цілих чисел $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 2 \cdot n)$ – сили героїв.

У третьому рядку задано n цілих чисел $b_1, b_2, \ldots, b_n (1 \le b_i \le 2 \cdot n)$ – сили монстрів.

Гарантується, що всі значення $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$ попарно різні.

У четвертому рядку задано одне ціле число q $(1 \le q \le n + 1)$ – кількість запитів.

У наступних q рядках задано по два цілі числа l та r $(0 \le l \le \mathrm{r} \le n)$ – параметри відповідного запиту.

Гарантується, що всі значення $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$ попарно різні.

Формат вихідних даних

Для кожного запиту в окремому рядку виведіть одне ціле число – шукане значення $(\sum_{i=l}^{r} ans_i) \bmod 998244353$.

Система оцінювання

1. (3 бали): $a_i < b_j$ для $1 \le i, j \le n$;
2. (9 балів): $q = 1$, $l = 1$, $r = 1$;
3. (6 балів): $a_i = 2 \cdot i - 1, b_i = 2 \cdot i$ для $1 \le i \le n$;
4. (16 балів): $n \le 500, q = 1, l = 0, r = n$;
5. (14 балів): $q = 1, l = 0, r = n$;
6. (15 балів): $q = 1, l = r$;
7. (17 балів): $n \le 500$;
8. (20 балів): без додаткових обмежень.

Приклад

standart input	standart output
3 3 4 6 1 2 5 3 1 2 2 3 3 3	2 3 1

Примітка. На малюнку нижче зображені герої та монстри першого прикладу. Герої розташовані зверху, а монстри – знизу. Число всередині квадрата позначає силу відповідного героя чи монстра.

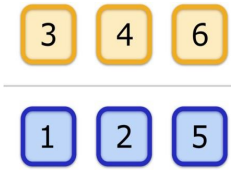


Рис. 1. Герої та монстри першого прикладу

У прикладі існує три можливі множини щасливих героїв: {1, 2, 3}, {2, 3} та {1, 3}. Знизу зображені три варіанти розподілу боїв, в яких будуть щасливими відповідні множини героїв. Зауважте, що може існувати декілька розподілів на бої, при яких буде щасливою та сама множина героїв.

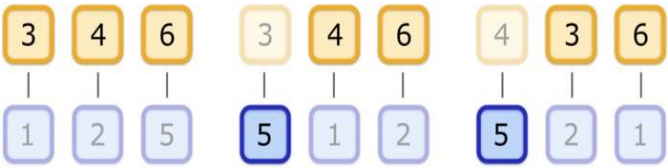


Рис. 2. Варіанти розподілу боїв, в яких будуть щасливими відповідні множини героїв

Задача D. Занулити підвідрізок

Задача з градерами

Для масиву додатних цілих чисел b довжини m визначимо $f(b)$ таким чином:

- нехай початково деяка змінна x рівна 0;
- за одну монету дозволяється збільшити значення x на 1;
- за одну монету дозволяється обрати елемент масиву $b_i (1 \le i \le m)$ та замінити його на $(b_i \oplus x)$, де $\oplus$ позначає операцію *побітового виключного АБО*;
- $f(b)$ дорівнює мінімальній кількості монет, необхідній для того, щоб зробити всі елементи масиву b одночасно рівними нулю.

Побітове виключне АБО невід'ємних цілих чисел a та b $(a \oplus b)$ дорівнює невід'ємному цілому числу, у якого у двійковому записі на певній позиції стоїть одиниця тоді і тільки тоді, коли у двійкових записах a та b на цій позиції знаходяться різні значення. Наприклад, $3_{10} \oplus 5_{10} = = 0011_2 \oplus 0101_2 = 0110_2 = 6_{10}$.

Задано масив додатних цілих чисел a довжини n та q запитів виду l, r. Для кожного запиту потрібно знайти $f([a_l, a_{l+1}, \ldots, a_r])$.

Протокол взаємодії

Вам потрібно реалізувати такі функції:

void init(integer n, array of integers a)

- n – ціле число, яке позначає довжину масиву;
- a – масив цілих чисел довжиною n;
- ця функція нічого не повертає.

integer ask(integer l, integer r)

- l – ціле число, яке позначає ліву межу запиту;
- r – ціле число, яке позначає праву межу запиту;
- ця функція повертає ціле число – $f([a_l, a_{l+1}, \ldots, a_r])$.

array of integers askAll(integer q, array of integers l, array of integers r)

- q – ціле число, яке позначає кількість запитів;
- l – масив цілих чисел довжиною q; l_i позначає ліву межу i-го запиту;
- r – масив цілих чисел довжиною q; r_i позначає праву межу i-го запиту;
- ця функція повертає масив цілих чисел; i-те число має бути рівне відповіді на i-й запит.

Формат вхідних даних

У першому рядку задано три цілі числа n, q та t $(1 \le n, q \le 2 \cdot \cdot 10^5; 1 \le t \le 2)$ – кількість чисел, кількість запитів і формат запитів відповідно.

У другому рядку задано n цілих чисел $a_1, a_2, ..., a_n$ ($1 \le a_i < 2^{60}$) – елементи масиву a.

У наступних q рядках задано по два цілі числа l та r ($1 \le l \le r \le n$) – параметри i-го запиту. Спочатку буде викликана функція init рівно один раз.

Якщо $t = 1$, то буде викликана рівно один раз функція askAll з усіма запитами. Якщо $t = 2$, то функція ask буде викликана рівно q разів.

Формат вихідних даних

Градер виведе в окремих рядках q цілих чисел – відповіді на запити.

Система оцінювання

1. (3 бали): $t = 1$, $a_i = a_1$ для $1 \le i \le n$;
2. (8 балів): $t = 1$, $a_i \ne a_j$ для $i \ne j$;
3. (3 бали): $i = 1$, $2^m + n \le a_i < 2^{m+1}$ для деякого натурального m;
4. (9 балів): $t = 1$, $a_i \le a_{i+1}$ для $1 \le i < n$;
5. (10 балів): $t = 1$, $n, q \le 1000$;
6. (11 балів): $t = 1$, $l_i = 1$ та $r_i = i$ для $1 \le i \le q$.
7. (10 балів): $t = 1$, $n, q \le 50\,000$;
8. (25 балів): $t = 1$;
9. (9 балів): $t = 2$, $n, q \le 10^5$;
10. (12 балів): $t = 2$.

Приклади

standart input	standart output
7 6 1 5 4 3 5 7 7 7 1 4 4 7 3 7 1 7 2 6 1 1	9 11 12 14 12 6
7 6 2 5 4 3 5 7 7 7 1 4 4 7 3 7 1 7 2 6 1 1	9 11 12 14 12 6

Розв'язання задач

I тур

Задача A. Точки на прямій

Блок 1

Для вирішення цього блоку достатньо перебрати всі можливі пари i, j та за n запитів перевірити, чи всі точки розташовані між цією парою. Тоді матимемо рішення за n^3 запитів, що з великим запасом проходить цей блок.

Блок 2

Нехай ми знаємо лише одне число x з шуканої пари точок x, y, тобто одну з крайніх точок. Знайдемо y. Будемо поступово розглядати точки від точок з меншими індексами до більших і при цьому підтримуватимемо точку v, бо відрізок з точок x, v містить всі точки, оброблені на даний момент. Початково покладемо $v := x$. Далі коли приходить нова точка i, то перевіряємо запитом, чи точка i розташована між точками x та v. Якщо ні, то присвоїмо $v := i$, оскільки пара точок v, x вже не містить нової точки i. В іншому випадку v змінювати не треба. Таким чином після оброблення всіх точок між парами точок x, v будуть знаходитися всі інші, отже, ми знайшли $y = v$.

Оскільки ми не знаємо x, то будемо його перебирати. Для кожного x запустимо алгоритм знаходження y (припускаючи, що x – крайня точка) та додатково за n запитів перевіримо, чи дійсно між парою точок x, y розташовані всі інші точки (перевірка припущення). Тоді маємо рішення, що виконує $2n^2$ запитів, що вкладається в заданий ліміт.

Блок 3

Поступово розглядатимемо наші точки й підтримуватимемо пару точок x, y, між якими розташовані всі розглянуті точки. Нехай розглядаємо точку i, тоді запитом дізнаємося, чи ця точка розташована між поточною парою точок x, y. Якщо розташована, то очевидно, що оновлювати пару x, y з надходженням цієї нової точки не треба. Якщо ж не розташована, то дізнаємося запитом, чи точка y розташована між парою точок x, i. Якщо розташована, то між парою точок x, i розташовані всі розглянуті точки, інакше між парою точок y, i. Коли розглядаємо

нову точку, то в гіршому випадку витрачаємо 2 запити, щоб оновити поточну пару точок, що містить всі інші. Тоді маємо рішення, що використовує не більше $2 \cdot n$ запитів.

Блок 6

Рішення на повний бал є покращенням рішення з блоку 3, тому варто спершу прочитати його.

Нехай m – кількість точок, на які ми витратили 2 запити, щоб оновити поточну пару x, y.

Іншими словами, це такі точки, що поточний максимум або мінімум координат розглянутих точок змінився. Краще, якби таких точок було небагато.

Давайте спробуємо переглянути наші точки в рандомному порядку, а не у фіксованому. Виявляється, що очікувана кількість оновлень префіксних максимумів у рандомному масиві буде $O(\log(n))$. Аналогічно з мінімумами. Тоді якщо переглядати наші точки в рандомному порядку, то очікуване значення m буде $O(\log(n))$. Відтак матимемо рішення, що очікувано використовує $n + O(\log(n))$ запитів, що на практиці із запасом у понад 2000 запитів вкладається в заданий ліміт.

Задача B. Подарунок Леді

Блок 1

У першому блоці $n \leq 5$ та $q = 0$, а це означає дві речі:

1) через те що нам відомі усі рядки, ми знаємо символ у кожному вузлі. Символ у вузлі під номером i дорівнює першому символу в рядку i;

2) у кожного вузла є n варіантів, куди може вести зв'язок із цього вузла. Тому сумарна кількість мереж, не враховуючи символи у вузлах, дорівнює n^n.

Маючи ці два факти, ми можемо написати просте рішення.

Спочатку знаходимо символи у кожному вузлі. Потім перебираємо усі варіанти зв'язків. Це можна зробити рекурсією або будь-яким іншим методом.

Потім для кожного варіанта перевіряємо, чи згенерована таблиця збігається з тою, що у вхідних даних. Якщо так, виводимо символи і зв'язки цієї мережі та закінчуємо виконання. Якщо ні, продовжуємо пошук.

Асимптотика рішення: $O(n^{n+2})$.

Блок 2

У другому блоці сказано, що мережа є набором пар та одиничних точок. Розглянемо ці два варіанти окремо.

Як буде виглядати рядок a_i, якщо $x_i = i$? Ми починаємо з вузла під номером i та рухаємося по зв'язках у мережі. Але зв'язок з i веде назад в i! Тому весь рядок a_i буде складатися з символу, записаного на вузлі i – s_i.

Тепер, як буде виглядати рядок для пари a, b, якщо $x_a = b$ і $x_b = a$? Розглянемо рядок a. Перший символ дорівнює s_a, другий символ s_b (тому що ми перейшли з вузла a у вузол b через зв'язок з вузла a), третій – s_a і так далі.

Ми застрягаємо у циклі між вершиною a та b, тому рядок a дорівнює s_a, s_b, s_a..., а рядок b – s_b, s_a, s_b… Як тепер відновити початкову мережу?

Якщо ми маємо рядок із символів *abababa*..., ми маємо цикл між двома вузлами, тому десь має бути рядок *bababab*... Ми можемо знайти усі такі пари та з'єднати їх між собою.

У випадку, коли рядок складається з одного символу, ми кажемо, що вузол з'єднаний сам із собою, тобто $x_i = i$. Зауважте, що можлива ситуація, коли в початковій мережі дві вершини були у парі та мали однакові символи. Це означає, що їхні рядки складаються з такого самого символу. Якщо ми з'єднаємо їх самих із собою, мережа буде відрізнятись, але згенерована таблиця – ні, тому це не проблема.

Асимптотика рішення: $O(n^3)$.

Блок 3

У третьому блоці мережа є набором зірок. У такому випадку в кожному рядку усі символи однакові, крім першого, який може відрізнятись.

Якщо у нас є рядок i типу *abbb*..., вузол i має вести в інший вузол, чий рядок дорівнює *bbbb*... Тоді знайдемо для кожного символу c номер рядка $root_c$, який повністю складається із символу c. Якщо такого рядка не існує, значення $root_c$ для цього символу нас не цікавить.

Тепер для кожного вузла i поставимо $x_i = root_c$, де c – другий символ рядка i. Відомо, що рядок $root_c$ повністю складається із символу c, тому згенерована таблиця буде абсолютно ідентичною до тої, що у вхідних даних.

Асимптотика рішення: $O(n^2)$.

Блок 4

У четвертому блоці мережа є деревом з коренем у вузлі 1. Для кожної вершини v знайдемо вершину p таку, що $a_{v,[1,3n-1]} = a_{p,[0,3n-2]}$, де $a_{v,[l,r]}$ – символи рядка v з l-го до r-й включно (символи в рядку пронумеровані від 0 до $3n - 1$). Цієї умови достатньо, щоб поставити $x_v = p$.

Довести це досить легко, можемо зауважити, що «шлях» кожної вершини закінчується в циклі з однієї вершини 1, тому що мережа – це дерево (не існує циклів крім кореня) і $x_1 = 1$. Тоді ми можемо подивитися на рядок a_v і побачити, що в наступної вершини x_v рядка a_{x_v} повинна збігатися з рядком a_v, починаючи з другого елементу.

Знайти таку вершину можна перебором, для кожної вершини є всього $O(n)$ кандидатів. Перевірку двох рядків на рівність виконуємо за $O(n)$.

Асимптотика рішення: $O(n^3)$.

Блок 5

Мережа є повним циклом довжини n. Тоді ми знаємо, що кожен рядок a_i складається з послідовності символів усіх вершин, повтореної тричі. Тоді для вершини v наступна вершина x_v повинна мати такий самий рядок, тільки циклічно зсунутий на один вліво. Тобто $a_{v,1} = a_{x_v,0}$, $a_{v,2} = a_{x_v,1}$, ..., $a_{v,0} = a_{x_v,3n-1}$.

Знайти наступну вершину можна перебором.

Асимптотика рішення: $O(n^3)$.

Блоки 6–9

Візьмемо рішення для блоку 4, але трохи його розвинемо. Мережа необов'язково буде деревом, але шлях кожної вершини точно закінчується в циклі (з кожної вершини є перехід, і вершин скінченна кількість, тому наявність циклів гарантована). Тоді давайте спробуємо схожий підхід.

Для вершини v знайдемо вершину p, таку що $a_{v,[1,3n-1]} = a_{p,[0,3n-2]}$. У правильній мережі завжди буде існувати така вершина, але що як їх декілька? В такому випадку підійде будь-яка. Це все тому, що максимальна довжина циклу, в якому закінчується шлях з вершини v, дорівнює n, тому якщо в рядках збігаються перші $3n - 1$ символів, буде збігатися і більше.

Але що робити з невідомими рядками? Якщо в якийсь момент ми не можемо для вершини v знайти потрібну вершину, це означає, що це одна з вершин з невідомими рядками. Немає різниці, яку вершину ми виберемо, тому візьмемо будь-яку таку вершину та присвоїмо їй рядок $a_{v,[1,3n-1]}$. Але тепер маємо проблему – ми не знаємо, який має бути останній символ!

Вирішити це досить просто – нам не потрібно знати останній символ. У найгіршому випадку найкоротший рядок буде завдовжки $2n + 1$, чого достатньо за тією самою логікою, що і раніше: серед перших $2n$ символів будуть символи усіх вершин на шляху, включаючи повний цикл мінімум два рази. Це означає, що якщо перші $2n$ символів рядків збігаються, їхні шляхи повністю однакові, тому наш алгоритм гарантовано побудує правильну мережу.

У випадку, коли усі рядки невідомі, ми можемо вивести будь-яку мережу з n вершин.

Тепер, щоб більш ефективно знаходити підходящий рядок, будемо рахувати хеш рядків. Тобто для кожної вершини v додамо у список пару з номером вершини на хешу рядка $a_{v,[0,2n-1]}$. Тепер, щоб знайти x_v, порахуємо хеш $a_{v,[1,2n]}$ та пройдемося по списку, шукаючи однаковий хеш.

Також існує рішення з префіксним деревом (бором) замість хешів або зі звичайним map.

Асимптотика рішення: $O(n^2)$.

Задача C. Запити красот підмасивів

Введемо функцію $sign(x)$, яка повертає 1, якщо $x \geq 0$, та 0 інакше.

Нехай масив c розбито на k підмасивів $[c_1, ..., c_{p1}]$, $[c_{p1} + 1, ..., c_{p2}]$, ..., $[c_{pk-1} + 1, ..., c_{pk}]$. Позначимо за s_i $(1 \leq i \leq k)$ суму i-го підмасиву. Тоді краса цього розбиття – $\min(|s_1|, |s_2|, ..., |s_k|)$.

Якщо є такий індекс j, що $s_j = 0$, то просто об'єднаємо цей підмасив з будь-яким із сусідніх.

Тоді краса розбиття зміниться з $\min(s_i)$, $1 \leq i \leq k$ на $\min(|s_i|)$, $1 \leq i \leq$ k, $i \neq j$, що покращить (або, у найгіршому випадку, не змінить) відповідь.

Нехай у нас є два послідовні підмасиви, сума яких однакова за знаком. Тобто є таке j $(1 \leq j \leq k)$, що $sign(s_j) = sign(s_{j+1})$. Об'єднаємо ці два підмасиви в один. Їх сума стане $s_j + s_{j+1}$, а краса розбиття зміниться з $\min(|s_1|, |s_2|, ..., |s_j|, |s_{j+1}|, ..., |s_k|)$ на $\min(|s_1|, |s_2|, ..., |s_j + s_{j+1}|, ..., |s_k|)$.

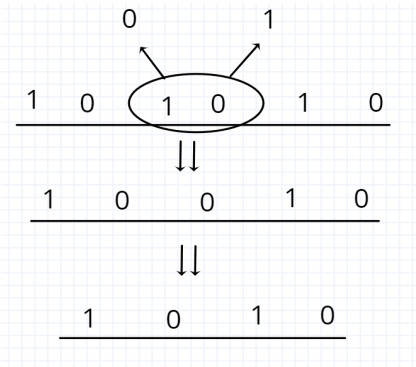
Зрозуміло, що оскільки знаки сум s_j та s_{j+1} однакові, то $|s_j + s_{j+1}| \geq \min(|s_j|, |s_{j+1}|)$, отже, відповідь не погіршиться. Це означає, що точно знайдеться розбиття, краса якого найбільша, а $sign(s_i) \neq sign(s_{i+1})$ для $1 \leq i < k$.

Блок 1

$a_i > 0$. Всі числа додатні, отже, для будь-якого розбиття масиву на підмасиви сума елементів кожного підмасиву також буде додатною. Тому, як ми попередньо довели, можна по черзі об'єднувати сусідні підмасиви, допоки не залишиться один, при цьому тільки покращуючи красу розбиття. Тому відповідь для кожного запиту – сума елементів від a_l до a_r. Через те що оновлень тут немає, таку задачу можна вирішити за допомогою префіксних сум. Складність – $O(n + q)$.

Блок 2

Нехай у нас є якесь розбиття на підмасиви, s_i $(1 \leq i \leq k)$ – сума i-го підмасиву, а також $sign(s_i) \neq sign(s_{i+1})$ $(1 \leq i < k)$. Візьмемо таке j, $1 < j < k - 1$. Об'єднаємо підмасиви j та $j + 1$. Тоді утвориться ситуація, як показано на рисунку.



Масив, розбитий на підмасиви, та один з варіантів його змінення. На малюнку 0 та 1 означають *sign*() для відповідного підмасиву.

Зауважимо, що отриманий підмасив за знаком точно дорівнює або лівому його сусіду, або правому. Об'єднаємо його з тим із сусідів, з яким знаки збігаються.

Як зміниться відповідь за всю цю операцію: спочатку в функції мінімуму було k змінних. З них $k - 3$ не змінились, 2 видалились,

а 1 змінна збільшилась за модулем. Тобто краса розбиття або стане краща, або не зміниться.

Нескладно помітити, що такі перетворення можна робити, поки $k \geq 3$. Це означає, що точно знайдеться таке розбиття, краса якого – оптимальна, а кількість підмасивів у розбитті менша або дорівнює трьом.

Для кожного запиту i розбиваємо масив на $[l_i, k_1]$, $[k_1 + 1, k_2]$, $[k_2 + 1, r_i]$, для цього перебираємо $l_i \leq k_1 \leq k_2 \leq r_i$ та рахуємо красу за допомогою префіксних сум. Складність – $O(q \cdot n^2)$.

Блок 5

Нехай p – масив префіксних сум масиву a. У цьому блоці $|p_i| \leq 5$. Суму будь-якого підмасиву $[l, r]$ можна представити як $p_r - p_{l-1}$. Це значить, що сума будь-якого підмасиву для цього блоку ≤ 10, отже, і відповідь ≤ 10.

Враховуючи доведене, що завжди оптимально розбивати на не більше ніж 3 підмасиви, можемо для кожного запиту i перебрати s_1 та s_2 і перевірити, чи можемо ми розбити масив $[l_i, r_i]$ на три підмасиви із сумами s_1, s_2 та $(a_{l_i} + a_{l_i+1} + ... + a_{r_i}) - s_1 - s_2$. Для цього треба прорахувати матрицю $d[i][s] = j$, j – перший такий індекс, що сума елементів з i-го до j-й дорівнює s.

Також треба не забути перевірити розбиття на 1 та 2 підмасиви аналогічним способом.

Рішення використовує $O(n \cdot C)$ пам'яті та має складність $O(n \cdot C + q \cdot C^2)$ часу, де $C = 21$.

Блок 3

Тут сказано, що існує оптимальне розбиття на не більше ніж 2 підмасиви. Випадок з розбиттям на один підмасив легко перевіряється за допомогою префіксних сум. Тому розбираємо випадок з двома масивами. Припустимо, що перший підмасив має від'ємну суму, другий додатну (навпаки вирішується аналогічно).

Нехай m – це межа наших підмасивів. Нескладно помітити, що завжди оптимально брати таку m ($l_i \leq m \leq r_i$), щоб p_m був мінімальний. У такому випадку значення другого підмасиву дорівнює $p_r - p_m$, і воно, як видно, максимізується, а отже, і мінімум з модулів цих величин досягає максимуму. Реалізувати це можна за допомогою структури даних, що вміє відповідати на запити мінімуму на відрізку, без оновлень.

Будуємо структуру по масиву $p[]$. Складність – $O(q \cdot \log_2 n)$, якщо використовувати дерево відрізків.

Блоки 4, 6–9

Відповідати на запит за $O(n^2)$ ми вже вміємо – для цього треба перебирати 2 межі розбиття.

Нехай у нас є ці дві межі – k_l та k_r. Аналогічно до розбору блоку 3 можна довести, що не гірше буде взяти такі дві межі $k_{l\min}$ та k_2, де $l_i \leq \leq k_{l\min} \leq k_l$ та значення $p_{k_{l\min}}$ є найменшим на цьому відрізку. Аналогічно замінюємо межу k_r на $k_{r\max}$, $k_r \leq k_{r\max} \leq r_i$ (беремо максимум, адже нам треба мінімізувати третій відрізок: $p_{r_i} - p_{k_r}$).

Тобто з початкових меж k_l та k_r ми знайшли не гірші. Через те що на відповідь впливають лише лівий та правий відрізки, то можна ще поліпшити відповідь, якщо починати пошук не з двох меж, а з однієї.

Виходить, для якогось j: краса = $min(|minp\ (l_i..j)|, |maxp(j+1 \ldots r_i) - sum|(l_i \ldots r_i)|)$. Перебравши j, можемо відповісти на запит за $O(n)$.

Останнє, що треба зробити, це подивитись на графіки функцій $|minp(l_i \ldots j)|$ та $|maxp(j \ldots r_i) - sum(l_i \ldots r_i)|$, $l_i \leq j \leq r_i$. Обидві функції, по суті, спадають, але через наявність модуля спочатку спадають, а потім зростають. Це дає змогу використати бінарний пошук для знаходження j.

У результаті маємо підтримувати на вершинах дерева відрізків (побудованого по масиву префіксних сум) значення *minp* та *maxp*. Це дає складність $O(q \cdot \log_2^2 n)$. Запити типу 2 перетворюють звичайне дерево відрізків на ліниве. Аби ж вирішити задачу за $O(q \cdot \log_2 n)$, треба застосувати каскадний спуск по дереву відрізків замість бінарного пошуку.

Задача D. AND масив

Припустимо, що у вхідних даних немає нулів.

Почнемо з підзадач.

Блок 1

Потрібно реалізувати те, що написано в умові, у будь-який спосіб. Наприклад, переписати псевдокод з умови на вашу мову програмування.

Блок 2

Спостереження: умова if буде виконуватися не більше ніж b разів. Це означає, що якщо ми можемо їх швидко знаходити, то ми розв'язали задачу.

Замість того щоб перебирати позицію наступного числа, як в алгоритмі з умови, можемо перебирати значення наступного. Серед позицій із цим значенням нас цікавить найперша, яка більша за вже позначені числа. Через те що всі числа степеня 2, перебирати значення можна за $O(b)$.

Блок 3

Навчимося швидко обчислювати $f(1, p)$.

Спосіб 1

Побудуємо табличку jump1[mask] = найменший індекс i, де a_i = = mask, або ∞, якщо не існує. Тоді будемо перебирати значення наступного числа (повинно бути mask & x = 0) і виберемо те, що зустрічається раніше за всіх.

Можна побачити, що ми завжди беремо позиції з більшим індексом за попередній, якщо візьмемо менший, то це протиріччя, бо могли взяти на попередній ітерації.

Спосіб 2

Побудуємо табличку jump2[mask] = найменше значення i, так що (a_i&mask) = a_i (тобто a_i є підмаскою mask).

Тоді наступне число має індекс jump2[$2^b - 1 - x$].

Вирішуємо для всіх s. Можна перебирати значення s від n до 1 та оновлювати таблиці, так можна обчислювати $f(s, z)$ для всіх аргументів.

Асимптотика

	оновлення	пошук
спосіб 1	$O(1)$	$O(2^b)$
спосіб 2	$O(2^b)$	$O(1)$

Блок 4

Нехай nxt = jump2[$2^b - 1 - msk$].

Можемо побачити, що значення $f(pos, msk) = pos + f(nxt, msk|a_{nxt})$. Будемо так рахувати й зберігати значення f.

Блок 5

Можна побачити, що час роботи цих алгоритмів значно відрізняється. Виявляється, їх можна поєднати й отримати щось посередині – $2^{b/2}$ на обидві операції.

Більш загально нам потрібна структура, яка вміє робити таке: зменшити last(Y) для певного значення Y до i.

Нехай last(Y) – це найменший індекс такий, що $a_{\text{last}(Y)} = Y$ або 10^{100}, якщо такого немає. Обчислити $f(X) = \min_{Y \subset X} \text{last}(Y)$.

Розділимо бітмаску X на 2 маски: $X_{\text{low}} = X\&(2^{b/2} - 1)$ $X_{\text{high}} = X\& \sim (2^{b/2} - 1)$.

Тобто X_{low} – нижні біти, а X_{high} – верхні.

Тоді можна розписати визначення f таким чином:

$$f(X) = \min_{Y_{\text{low}} \subset X_{\text{low}}} \min_{Y_{\text{high}} \subset X_{\text{high}}} \texttt{last}(Y_{\text{low}} \cup Y_{\text{high}}).$$

Визначимо функцію

$$g_1(X) = \min_{Y_{\text{high}} \subset X_{\text{high}}} \texttt{last}(X_{\text{low}} \cup Y_{\text{high}}).$$

Тепер маємо:

$$f(X) = \min_{Y_{\text{low}} \subset X_{\text{low}}} g_1(Y_{\text{low}} \cup X_{\text{high}}).$$

Якщо g перераховані, то f можна порахувати за $O(2^{b/2})$, використовуючи формулу з g. Але при оновленні last g також перераховується за $O(2^{b/2})$, бо одне значення y впливає лише на $g(X)$ для $O(2^{b/2})$ значень X.

Блок 6

Аналогічно

$$g_2(X) = \min_{Y_{\text{low}} \subset X_{\text{low}}} \texttt{last}(Y_{\text{low}} \cup X_{\text{high}});$$

$$f(X) = \min_{Y_{\text{high}} \subset X_{\text{high}}} g_2(X_{\text{low}} \cup Y_{\text{high}}).$$

Якщо вибирати з двох визначень (через g_1 або g_2), то можна обчислювати за $O(2^{\text{popcount}(X)/2})$. Це покращить асимптотику через те, що більшість аргументів мають небагато бітів.

II тур

Задача А. Кольорова таблиця

Символи таблиці будемо асоціювати з кольорами, R – червоний, G – зелений, B – блакитний.

Якщо доступних кольорів лише 2, то існує лише два валідних розфарбування таблиці, які є шаховими розфарбуваннями таблиці (червоно-зелена дошка в даному випадку). Нехай c_1 та c_2 – це кількість змін клітинок таблиці, що треба зробити, щоб таблиця мала перше та друге шахове розфарбування відповідно. Оскільки колір кожної клітинки треба змінити тільки в одному із цих розфарбувань, то маємо, що $c_1 + c_2 = n \cdot m$. Тоді нескладно збагнути, що $\min(c_1, c_2) \leq \frac{n \cdot m}{2}$, тож маємо, що можемо перефарбувати не більше половини клітинок, щоб не існувало пари сусідніх по стороні клітинок однакового кольору, чого і треба було досягти.

Якщо доступно три кольори, то одне з шахових розфарбувань вже не завжди підходить, тому спробуємо трохи модифікувати цю конструкцію. Розглянемо шахове розфарбування дошки у чорно-білий кольори. Спробуємо перефарбувати нашу дошку таким чином: змінюємо колір клітинок, що розміщені на позиціях білих клітинок у чорно-білій шаховій таблиці, на червоний колір, а інші клітинки (у чорно-білій шаховій таблиці вони є чорними) перефарбовуємо у зелений або блакитний (неважливо, який саме із них) колір, якщо вони червоні. Подібних розфарбувань існує 6:

1) клітинки на позиціях білих клітинок фарбуємо у червоний, клітинки на позиціях чорних клітинок фарбуємо у якийсь із інших кольорів, якщо клітинка є червоною (цей варіант було описано вище);

2) клітинки на позиціях білих клітинок фарбуємо у зелений, клітинки на позиціях чорних клітинок фарбуємо у якийсь із інших кольорів, якщо клітинка є зеленою;

3) клітинки на позиціях білих клітинок фарбуємо у блакитний, клітинки на позиціях чорних клітинок фарбуємо у якийсь із інших кольорів, якщо клітинка є блакитною;

4) клітинки на позиціях чорних клітинок фарбуємо у червоний, клітинки на позиціях білих клітинок фарбуємо у якийсь із інших кольорів, якщо клітинка є червоною;

5) клітинки на позиціях чорних клітинок фарбуємо у зелений, клітинки на позиціях білих клітинок фарбуємо у якийсь із інших кольорів, якщо клітинка є зеленою;

6) клітинки на позиціях чорних клітинок фарбуємо у блакитний, клітинки на позиціях білих клітинок фарбуємо у якийсь з інших кольорів, якщо клітинка є блакитною.

Нехай R_0 – кількість червоних клітинок, що розміщені на позиціях білих клітинок на чорно-білій дошці. R_1 – кількість червоних клітинок, що розміщені на позиціях чорних клітинок на чорно-білій дошці. Аналогічно визначаємо G_0, G_1, B_0, B_1.

Тоді розпишемо, скільки змін клітинок треба для кожного з 6 варіантів:

1) $G_0 + B_0 + R_1$;
2) $R_0 + B_0 + G_1$;
3) $R_0 + G_0 + B_1$;
4) $G_1 + B_1 + R_0$;
5) $R_1 + B_1 + G_0$;
6) $R_1 + G_1 + B_0$.

Покажемо, що одне із цих перефарбувань змінює не більше половини клітинок початкової таблиці. Від супротивного. Припустимо, що $G_0 + B_0 + R_1 \geq \left[\frac{n \cdot m}{2}\right] + 1$, аналогічно маємо ще 5 нерівностей.

Додамо всі ці 6 нерівностей. Маємо: $3 \cdot (R_0 + R_1 + G_0 + G_1 + B_0 + B_1) \geq$ $\geq 6 \cdot \left[\frac{n \cdot m}{2}\right] + 6$.

Очевидно, що $R_0 + R_1 + G_0 + G_1 + B_0 + B_1 = n \cdot m$. Тоді маємо, що $3 \cdot n \cdot m \geq 6 \cdot \left[\frac{n \cdot m}{2}\right] + 6 \leftrightarrow \frac{n \cdot m}{2} \geq \left[\frac{n \cdot m}{2}\right] + 1$. Маємо суперечність, що завершує доведення. Отже, одне із цих перефарбувань підходить. Тоді просто знайдемо, яке з них підходить, і виведемо отриману таблицю.

Складність рішення: $O(n \cdot m)$.

Задача В. Футбол

Блок 1

Для кожної комбінації просимулюємо гру, коли знову перший гравець робитиме нульову передачу, та знайдемо мінімальну силу гравця, що зустрівся в процесі, це і буде відповідь. Оскільки в такому

випадку жодного разу не зустрінеться гравець, що двічі зробить одну й ту ж передачу, то для кожної комбінації буде максимум $n \cdot q$ кроків, тому, щоб знайти силу для кожної комбінації, нам буде потрібно $O(n \cdot q^2)$ операцій.

Блок 2

Оскільки всі значення a_i однакові, то відповідь завжди буде рівна першому елементу.

Блок 3

Після виконання m передач м'яч перейде до гравця, розташованого на відстані $l = \sum_{i=0}^{m-1} k_i$ від першого, і знову буде нульова передача. Щоб м'яч повернувся до першого гравця на нульовій передачі, нам потрібно зробити x разів по m передач та $x \cdot l$ має ділитися на n, тобто ми зробимо певну кількість повних кіл. Оскільки n просте, то або l, або x має ділитися на n. Якщо l ділиться на n, то нам достатньо зробити лише перші m передач і знайти мінімум серед них, тому при переході від даної комбінації до наступної нам слід зберігати й оновлювати l, мінімум серед m перших передач, на якому закінчилися перші m передач. В іншому випадку x має дорівнювати n, а оскільки всі гравці, що робили нульову передачу, повинні бути різними, то всі гравці робитимуть нульову передачу, і в нашій комбінації будуть всі гравці, тобто відповіддю в даному випадку буде мінімум серед всіх гравців.

Блок 4

Хоч l може бути більшим за n, проте нам важлива лише змінна позиції в колі, тому ми можемо замінити l на $l \bmod n$. Оскільки $x \cdot l$ має ділитися на n та x має бути мінімально можливим, то $x = \frac{n}{gcd(l, n)}$. Створимо масив b, в якому ми зберігатимемо перших m гравців, що робили передачі. Тоді щоб знайти мінімальну силу гравця, що робив певну передачу, нам слід знайти мінімум серед сил гравців з номерами $b_i + l \cdot y$, де y від 0 до $x - 1$. Тоді відповідь для певної комбінації буде мінімум серед відповідей для всіх передач. Щоб знаходити відповідь для певної комбінації за $O(n)$, а не за $O(n^2)$ операцій, порахуємо для кожного l від 0 до $n - 1$ відповідь для кожної можливої стартової позиції. Зауважимо, що для кожного серед номерів $l \cdot y + 1$ відповідь буде однаковою, оскільки для початкового номера $l + 1$ всі номери будуть зсунуті на 1 та номер $x \cdot y - y + l + 1$ по модулю n буде

дорівнювати 1. Аналогічно, якщо початковим буде інший номер. Тому, знаючи відповідь для одного з номерів, ми можемо проставити відповідь для всіх інших зв'язаних з ним, що дає змогу порахувати табличку за $O(n^2)$ операцій. Тоді складність алгоритму буде $O(n^2)$.

Блок 5

Якщо поглянути на табличку, яку ми зробили в попередньому блоці, можна помітити, що багато рядків у ній ідентичні. Оскільки для взаємно простих l і n, $x = n$ та $(l \cdot y) \bmod n$ дасть всі числа від 0 до $n - 1$, то для довільного l, що є взаємно простим з n, ми отримаємо однакові $(l \cdot y) \bmod n$ числа, проте в різному порядку. Якщо l і n не є взаємно простими, то нехай $l' = \frac{l}{\gcd(l, n)}$ та $n' = \frac{n}{\gcd(l, n)}$, тоді аналогічно $(l' \cdot y') \bmod n'$ дасть всі числа від 0 до $n' - 1$, домноживши їх на $gcd(l, n)$, ми отримаємо всі числа, що дасть $(l \cdot y) \bmod n$. Отже, для різних l, в яких однаковий найбільший спільний дільник з n, наша табличка буде збігатися, тому ми можемо порахувати табличку лише для степенів двійки та всі l замінити на найбільшу степінь двійки, що їх ділить. Тоді, щоб порахувати табличку, буде потрібно лише $O(n \cdot \log(n))$ операцій. Щоб за таку ж асимптотику знаходити силу комбінацій, слід створити додатковий масив і для кожної степені двійки зберігати відповідь у ньому, при переході до наступної комбінації нам потрібно буде лише оновити відповідне значення в масиві на значення з таблички, відповіддю буде значення в масиві для l.

Блок 6

Замінивши степені двійки на дільники числа n в попередньому блоці, ми отримаємо розв'язок до цього. Проте максимальна кількість дільників у числа рівна $\log^2(n)$, тому асимптотика буде $O(n \cdot \log^2(n))$.

Блок 7

Розв'язок попереднього блоку по часу проходить цей, проте табличка, яку ми створюємо, буде завелика. Оскільки всі числа в табличці розбиті на блоки довжини x, що не перетинаються, то їхня кількість дорівнюватиме $\frac{n}{x} = l$, також можна помітити, що перші l чисел будуть у різних блоках, а далі номери блоків будуть іти в тому ж порядку, тому ми можемо зберігати лише відповіді для перших l чисел, значення певного a_i дорівнюватиме значенню $a_i \bmod n$, яке є в нашій табличці.

Задача С. Герої та монстри

Блок 1

Через те що кожен герой слабкіший за кожного монстра, єдиний збір щасливих героїв – пустий.

Тому відповідь дорівнює 1 для $l = 0$ та 0 для $l > 0$.

Асимптотика рішення: $O(n + q)$.

Блок 2

Нам потрібно лише знайти значення ans_1. Давайте відсортуємо усіх монстрів та героїв у порядку зростання їхньої сили.

Тепер будемо для кожного героя окремо перевіряти, чи ми можемо так розставити бої, щоб був щасливий тільки він. Для цього ми повинні поставити його у парі з монстром, який буде слабкішим за нього.

Останньою проблемою залишаються інші герої – усі вони мають бути засмучені. Зауважимо, що найслабший герой має боротися з найслабшим монстром. Якщо він боротиметься з більш сильним монстром, найслабший монстр боротиметься з більш сильним героєм, який може виявитися сильнішим за нього, а значить, буде щасливим, що нам не підходить.

Більш формально, якщо у нас є якийсь розподіл на бої, де усі герої засмучені та найслабший герой стоїть у парі не з найслабшим монстром, також існує розподіл, де усі герої так само засмучені, але найслабший герой стоїть у парі з найслабшим монстром. Тепер найслабший герой та найслабший монстр стоять у парі, а значить другий найслабший герой має стояти у парі з другим найслабшим монстром (за такою самою логікою) і так далі.

Тобто якщо у нас є множина героїв і монстрів і ми хочемо перевірити, чи існує такий розподіл, де усі герої засмучені, то сортуємо монстрів і героїв за їхньою силою і ставимо у відповідні пари. Якщо усі герої засмучені – таке розбиття існує, якщо ні, то ні.

Тоді з яким монстром має боротися вибраний нами щасливий герой? Очевидно, що з найслабшим. За подібною логікою, якщо існують інші підходящі розбиття, де це не так, існує підходяще розбиття, де ця умова виконується.

Тому маємо рішення: проходимо по усіх героях, ставимо вибраного героя у пару з найслабшим монстром та перевіряємо, чи решта героїв будуть засмучені.

Асимптотика рішення: $O(n^2)$.

Блок 3

У цьому блоці фіксовані сили усіх монстрів і героїв. Можемо зауважити, що для кожної множини героїв без найслабшого героя існує розподіл на бої, де ця множина героїв щаслива, а усі інші – засмучені.

Показати це можна конструктивно: для найслабшого героя з множини поставимо найслабшого монстра, для другого найслабшого героя – другого монстра і так далі. Тоді кожен герой з множини буде щасливим, тому що у кожного наступного героя як мінімум на одного слабкішого за нього монстра більше, ніж у попереднього героя. Через те що найслабший герой у множині – другий найслабший серед усіх (перший не може бути в множині, тому що не існує монстра, слабкішого за нього), у нас завжди буде наступний монстр, який слабкіший за нового героя.

Залишилося показати, що можна зробити усіх інших героїв засмученими. Показати це досить легко – будемо ставити найслабшого незайнятого монстра з найслабшим героєм не з множини, доки у нас залишаються незайняті монстри. Легко помітити, що кожен монстр буде сильнішим за героя. Тоді ans_k = кількість підмножин героїв 2, 3, ..., n розміру k, що дорівнює $C_k^{n-1} = \frac{n-1!}{k!(n-1-k)!}$.

Маючи ці значення, легко відповідати на запити, побудувавши, наприклад, масив префіксних сум.

Асимптотика: $O(n + q)$.

Блоки 4 та 7

Читаючи рішення, можна помітити таку тенденцію: якщо ми хочемо перевірити, чи можемо зробити множину героїв S щасливими, а усіх інших – засмученими, потрібно поставити героїв з S із найслабкішими монстрами у тому самому порядку, а інших – з найсильнішими (теж у тому самому порядку). Тоді давайте зразу відсортуємо усіх монстрів та героїв у порядку зростання їхньої сили.

Спробуємо розв'язати цю задачу динамічним програмуванням.

Нехай $dp[i][a][k]$ – кількість таких підмножин героїв 1, 2, ..., i, що a з них щасливі, а k – обмеження на кількість щасливих героїв. Базою буде $dp[0][0][k] = 1$ для усіх $k \in [0, n]$.

Тепер маємо всього два переходи – ми або робимо героя $i + 1$ щасливим, або ні. У першому випадку ми ставимо героя $i + 1$ з монстром

$a + 1$ (через те що це найслабший доступний монстр) та переходимо в $dp[i + 1][a + 1][k]$ (такий перехід можливий, тільки якщо $a + 1 \le k$ та герой $i + 1$ сильніший за монстра $a + 1$).

У другому випадку ми ставимо героя $i + 1$ з монстром $k + i - a + 1$. Чому? Всього у нас буде k щасливих героїв, тому найслабший монстр, який буде боротися із засмученими героями, – монстр під номером $k + 1$. Але у нас уже є $i - a$ засмучених героїв, кожен з яких боровся з такими монстрами, тому найслабший доступний монстр – $k + i - a + 1$. У такому випадку ми переходимо у $dp[i + 1][a][k]$. Такий перехід можливий, тільки якщо герой слабкіший за відповідного монстра та якщо такий монстр існує ($k + i - a + 1 \le n$).

Таким чином ми можемо порахувати усі значення $dp[i][a][k]$. Відповідь: ans_k знаходиться у $dp[n][k][k]$.

Асимптотика: $O(n^3 + q)$.

Блок 5

У цьому блоці нам потрібно лише знати кількість множин S. Тоді давайте спробуємо оптимізувати попереднє рішення.

Спочатку закинемо усіх героїв та монстрів у один масив та відсортуємо його (звісно, зберігаючи, хто герой, а хто монстр). Тоді нам не потрібно окремо тримати k, щоб знати, який монстр буде найслабшим серед тих, що виграють.

Нехай $dp[i][a][t]$ – кількість множин героїв S серед перших i «персонажів» (з масиву відсортованих монстрів і героїв), таких що a з них щасливі. Тепер $t = 0$ означає, що усіх монстрів, яких ми зустрічали, ми поставили у пару з героями, які сильніші за них, а $t = 1$ – хоча б один монстр стоїть у парі зі слабкішим героєм. Легко помітити, якщо ми ставимо монстра зі слабкішим героєм, то усі наступні (сильніші) монстри теж мають стояти у такій парі (інакше ми порушуємо наш оптимальний спосіб розподіляти монстрів та героїв на бої).

Тому ми маємо декілька випадків та декілька переходів. У першому випадку персонаж i – це герой. У такому випадку ми можемо спробувати зробити його або щасливим, або засмученим. Ми можемо зробити його щасливим, тільки якщо $t = 0$ (інакше герой мусить програти) та якщо є вільні монстри, слабкіші за нього. Перевірити це досить просто.

Спочатку нам потрібно знати кількість вже розглянутих героїв та монстрів, нехай ці значення дорівнюють $hero_i$ та $monster_i$ відповідно. Тоді серед $monster_i$ монстрів a найслабкіших уже зайняті (через те що вони стоять у парі із сильнішим героєм), тому ми можемо зробити героя щасливим, тільки якщо $a < monster_i$. У такому випадку ми переходимо в $dp[i + 1][a + 1][t]$ (не забувайте, що $t = 0$ у такому переході).

Якщо ми хочемо зробити героя засмученим, ми просто не робимо його поки що щасливим, тому що ми ще не зустрічали монстра, сильнішого за нього. Тому просто робимо перехід у $dp[i + 1][a][t]$ (у цьому переході t може дорівнювати як 0, так і 1).

Тепер розглянемо випадок, коли персонаж i – монстр.

У нас знову є два випадки. Якщо ми кажемо, що монстр буде засмученим (у парі з більш сильним героєм), то t має дорівнювати 0 (інакше більш слабкий монстр уже був щасливим, а значить, і усі сильніші монстри мають бути щасливі). Тоді ми просто переходимо у $dp[i + 1][a][t]$.

Якщо ж ми кажемо, що монстр буде щасливим, ми маємо спочатку перевірити, чи існує вільний слабкий герой. Усього ми отримали $hero_i$ героїв, a з яких щасливі. Тому $hero_i - a$ з них вільні, так? Не зовсім, у випадку, коли $t = 1$, деякі з них можуть бути уже зайняті іншими монстрами, які сильніші. Через те що у цьому блоці нам неважлива кількість щасливих героїв у множині, давайте просто надалі використовувати a для $t = 1$. Тобто для $t = 1$ a – це кількість героїв, які уже стоять у парі, і неважливо, щасливі вони в тій парі чи ні.

Тоді ми можемо зробити монстра щасливим і перейти у $dp[i + 1][a + 1][1]$.

Таким чином, ми пройдемося по усіх можливих множинах щасливих героїв S, і відповідь буде дорівнювати $dp[2n][n][0] + dp[2n][n][1]$ (через те що ми маємо розглянути усіх персонажів, кожен з n героїв під кінець повинен мати пару та серед усіх монстрів або буде хтось щасливий, або ні).

Асимптотика рішення: $O(n^2 + q)$.

Блок 6

Попереднє рішення виглядало перспективним, але на переході, де монстр стає щасливим, усе ламається. У цьому блоці ми маємо

порахувати лише ans_k, тож чому б не обмежити динамічне програмування?

Якщо ми дозволимо перехід, де перший монстр стає щасливим (перехід з $dp[i][a][0]$ в $dp[i + 1][a + 1][1]$) тільки коли рівно k монстрів уже засмучені, ми під кінець матимемо кількість тільки таких множин S, де кількість щасливих героїв дорівнює k.

Відповідь буде так само дорівнювати $dp[2n][n][0] + dp[2n][n][1]$.

Асимптотика: $O(n^2)$.

Блок 8

Попереднє рішення виглядає ще більш перспективним, але чи ми дійсно маємо окремо перебирати k і рахувати відповідь?

Виявляється, що ні. Давайте будемо рахувати два динамічні програмування замість одного. У першому ми будемо йти зліва направо (від найслабкіших персонажів до найсильніших) та дозволяти монстрам бути лише засмученими. У другому ми будемо йти справа наліво (від найсильніших персонажів до найслабкіших) та дозволяти монстрам лише бути щасливими. У першому динамічному програмуванні усе так само, як і в рішенні 5-го блоку, лише без переходів, де монстр щасливий. У другому усе так само, тільки тепер, коли ми робимо героя засмученим, нам потрібно перевірити, чи існує сильніший за нього монстр. Також i має трохи інше значення – тепер це означає, що ми роздивилися усіх персонажів, окрім i перших (найслабкіших). Ще ми навпаки міняємо a – ми починаємо з $a = n$ та зменшуємо a, коли кажемо, що герой щасливий.

Іншими словами, друге динамічне програмування – те ж саме, що і перше, тільки ми йдемо з кінця до початку і робимо усіх монстрів щасливими, а не засмученими.

Тепер загадка, як знайти ans_k? Ми знаємо, що під час переходу з $t = 0$ до $t = 1$ ми знаємо точну кількість щасливих героїв, чому б нам тоді не перебрати такий момент?

Пройдемося по усіх i, таких що i-й персонаж – монстр. Тепер переберемо a – кількість щасливих героїв у момент переходу. Тоді якщо ми перемножимо $dp_1[i][a]$ та $dp_2[i][a]$, то матимемо кількість таких множин S, де кожен герой із S щасливий, усі інші засмучені, та $|S|$ = кількість засмучених монстрів, тобто $monster_i - 1$ (бо ми зробили усіх монстрів до i засмученими).

Таким чином ми переберемо усі варіанти, де хоча б один монстр щасливий. Варіант, де усі монстри сумні, порахуємо окремо в лоб. Тож під кінець ми маємо усі значення ans_k, порахувавши лише два ДП, тому ми можемо з легкістю відповісти на усі запити.

Асимптотика: $O(n^2 + q)$.

Бонус: розв'яжіть задачу за $O(n + \sqrt{(n \log n)}/64 + q)$.

Задача D. Занулити підвідрізок

Спершу зробимо декілька загальних міркувань, що знадобляться в рішеннях.

Навчимося рахувати $f(b)$. Розглянемо деякий масив b_1, b_2, ..., b_m. Нехай y – найстарший біт, що зустрічається у числах масиву b.

Твердження 1. Необхідно буде збільшити x (змінна з умови) так, щоб він був не менший за 2^y. Це так, оскільки інакше ми жодним чином не зможемо прибрати біт y у числах, в яких він зустрічається у нашому масиві.

Тоді виходить, що всі числа, які менші за 2^y, можемо занулити (зробити рівними нулю), використовуючи рівно одну операцію на цих числах (колись x стане рівним $b_i \le 2^y$, і тоді ми використаємо операцію XOR із цим елементом, щоб зробити його рівним 0). Оскільки менше операцій, щоб занулити ці числа, зробити не можна, тож так робити оптимально.

Отже, тепер залишилося визначити, що робити з числами, які більші за 2^y.

Твердження 2. Числа масиву, що більші за 2^y, можна гарантовано занулити за дві операції. Дійсно, якщо маємо число $b_i > 2^y$, то можемо, коли $x = 2^y$, виконати операцію XOR з b_i та коли $x = b_i \oplus 2^y$.

Нехай k – кінцеве значення змінної x після виконання всіх операцій. $k \ge 2^y$. Тоді можемо всі числа, що менші або дорівнюють k, занулити за одну операцію, а всі інші – за дві операції. Нескладно збагнути, що за такого k оптимальніше не зробити.

Нехай $g(k)$ – кількість чисел у масиві b, що менші або дорівнюють k. Тоді щоб занулити всі числа масиву з таким кінцевим значенням x, що дорівнює k, треба виконати $k + m + (m - g(k))$ операцій: k операцій збільшення змінної x, m операцій XOR на кожне з чисел масиву та ще $m - g(k)$ операцій для чисел, що зануляємо за дві операції.

Тоді треба знайти мінімум функції $k + 2 \cdot m - g(k)$ для $k \geq 2^y$. Оскільки $2 \cdot m$ не залежить від k, то треба знайти мінімум функції $k - g(k)$. Нескладно довести, що мінімум досягається на деякому $k < 2^y + m$.

Блок 1

Кінцеве значення x дорівнюватиме або 2^y, або a_1, де y − найстарший біт числа a_1. Тоді відповідь за запит буде $\min(2^y + 2 \cdot m, a_1 + m)$.

Блоки 2, 3

Кінцеве значення x дорівнюватиме 2^y, де y – найстарший біт чисел з підвідрізка запиту. Далі залишилось визначити, скільки чисел менше або дорівнює 2^y на відрізку, що можна зробити шляхом перерахунку префіксних сум для кожного можливого y.

Блок 5

Достатньо було знайти мінімум функції $k - g(k)$ за $O(n)$, при цьому відсортувавши поточний масив запиту за $O(n\log(n))$.

Блок 4

Оскільки числа на підвідрізка вже відсортовані, то це дає змогу знайти мінімум функції $k - g(k)$ за $O(\log(n))$, використовуючи дерево відрізків.

Блок 6

Створимо структуру даних, що дасть змогу поступово додавати елементи до неї та шукати мінімум функції $k - g(k)$, що допоможе розв'язати цей блок. Наведена далі структура даних використовуватиметься у повному рішенні.

Маємо m можливих значень $2^y \leq k < 2^y + m$. Створимо для кожного із цих значень k окрему вершину. У вершині $k = 2^y$ запишемо значення $2^y - g(2^y)$. У вершині, що відповідає значенню $k = i > 2^y$, запишемо число $i - g(i) - (i - 1 - g(i - 1)) = 1 - cnt(i)$, де $cnt(i)$ – кількість елемента i у масиві b. Між вершинами, що відповідають значенням k та $k + 1$, проведемо ребро. Тобто побудуємо бамбук.

Нескладно помітити, що сума значень вершин на шляху від вершини, що відповідає $k = 2^y$, до вершини з $k = t$ дорівнює $t - g(t)$. Тоді мінімізація $t - g(t)$ еквівалентна знаходженню шляху, що проходить через вершину 2^y, та мінімізує суму вершин на шляху.

Побудуємо структуру даних, що знаходить мінімум $k - g(k)$ та підтримує додавання елемента у масив b. Вважатимемо, що всі числа, що додаватимуться, матимуть один і той самий старший біт y, аналогічно будуватиметься структура, що працює незалежно від того, який старший біт у числах, що додаються.

Побудуємо систему неперетинних множин (СНМ) з n вершин (n – загальна кількість чисел у всьому масиві a), де вершина i відповідає значенню $k = 2^y + i$ (тут i від 0 до $m - 1$ включно). Початково в масиві немає чисел, тому значення записане в кожну вершину, крім першої, $1 - cnt(2^y + i)$ дорівнює 1. Ми намагаємося знайти шлях мінімальної суми значень вершин, що проходить через першу вершину. Неформально кажучи, якщо ми вибрали початкову вершину шляху, то ми хочемо дійти до першої вершини, але на шляху до неї наразі є «штрафні» вершини, що додають 1 до значення шляху. Отже, початково всі вершини, крім першої, є «штрафними». Початково $ans := \min(k - g(k)) = 2^y$ – змінна, що зберігає поточний мінімум у структурі.

Навчимося додавати нове число у структуру. Нехай додається число v, що відповідає вершині i в СНМ. Якщо ця вершина є «штрафною», тобто значення у вершині дорівнює 1, то нове значення $1 - cnt(v)$ дорівнюватиме 0 і вершина перестане бути «штрафною». Якщо ж вершина вже не є «штрафною», то коли ми проходитимемо через неї, значення на шляху зменшуватиметься, чого ми й прагнемо. Спробуємо за рахунок такого «бонусу» цієї вершини прибрати іншу «штрафну» вершину на шляху до першої вершини, а саме: знайдемо найпершу «штрафну» вершину зліва від поточної й приберемо з неї штраф. Якщо ж такої вершини немає, то ми можемо дійти до першої вершини без перешкод («штрафних» вершин), тому віднімемо від поточного мінімуму 1. Найпершу «штрафну» вершину зліва можна шукати СНМ, якщо оновлювати її відповідним чином.

Формально для додавання нового числа й оновлення поточного мінімуму ми виконуємо такі дії:

1) будуємо СНМ на n вершинах та ініціалізуємо $ans := 2^y$;

2) кожна компонента у СНМ буде якимось неперервним відрізком вершин, тож підтримуємо інваріант, що найлівіша вершина у компоненті

буде «штрафною» (крім найлівішої компоненти, оскільки перша вершина ніколи не є «штрафною»);

3) коли додається число v, то знаходимо компоненту в СНМ, де є вершина, що відповідає цьому числу. Також знаходимо сусідню компоненту зліва у СНМ до знайденої та об'єднуємо їх в одну, бо ми прибрали штраф з найлівішої вершини поточної компоненти;

4) якщо ж такої сусідньої компоненти зліва немає, то віднімаємо від ans одиницю і не змінюємо жодним чином структуру СНМ;

5) ans − буде поточним мінімумом шуканої функції.

Якщо використовувати СНМ зі стисненням шляху та ранговою евристикою, то можемо оновлювати структуру амортизовано за $O(\lambda(n))$, де $\lambda(n)$ – обернена функція Акермана.

Блок 7

За допомогою дерева відрізків можемо підтримувати мінімум $k - g(k)$, крім того, воно буде підтримувати видалення елемента, що дає змогу використовувати алгоритм Мо з видаленнями, і матимемо рішення за $O(n\sqrt{n\log(n)})$.

Блок 8

Опишемо рішення задачі, коли запити подаються в офлайні (коли всі запити відомі заздалегідь). Задача в онлайні буде розв'язуватись схожим чином. Наступну техніку можна назвати алгоритмом Мо без видалень.

Розіб'ємо наш масив на блоки довжиною $\sqrt{n}$. Тепер відсортуємо наші запити по блоку, в якому міститься ліва границя запиту. Розглянемо запити, що починаються в одному такому блоці, і навчимося знаходити на них відповіді. Створимо порожню структуру, що описували вище (для кожного блоку вона будується наново). Відсортуємо ці запити по правій границі, будемо відповідати на них у міру її збільшення. Нехай r – права границя блоку, в якому містяться ліві границі запитів. Почнемо додавати до нашої структури числа масиву, починаючи з $r + 1$. Нехай наразі відповідаємо на запит $[x, y]$, тоді додаємо до нашої структури числа з $r + 1$ до y. Оскільки y (права границя запитів) збільшується, то, щоб оновлювати структуру, не треба її перебудовувати кожен раз, а лише додавати нові числа, що не були додані на момент пошуку відповіді на попередній запит. Отже, зараз маємо, що майже всі числа масиву з відрізка $[x, y]$ додані в поточну структуру. Не додані лише числа

з відрізка $[x, \min(r, y)]$. Таких чисел не більше ніж $\sqrt{n}$, що небагато і дає змогу обробити кожен з них окремо при пошуку відповіді на запит. На жаль, ми не можемо додавати ці числа в нашу структуру, бо тоді вона зміниться, а видаляти елементи ми не навчилися. Натомість відкочування змін після додавання цих елементів також не спрацює достатньо швидко, оскільки оцінка додавання в структуру є амортизованою.

Тоді треба якимось спеціальним чином обробити ті елементи, що не додали до структури.

Розглянемо їх у порядку зменшення їхніх значень (щоб не сортувати масив з близько $\sqrt{n}$ чисел кожен раз, відсортуємо елементи кожного блоку окремо і будемо проходити по всіх елементах блоку в порядку спадання значень елементів і перевірятимемо, чи елемент є у поточному відрізку $[x, \min(r, y)]$). Якщо додавати елементи в такому порядку, то компоненти, які ми розглядатимемо в СНМ, будуть також розглядатися в спадному порядку, і тоді ми можемо зберігати вказівник на останню компоненту, що ми мали б об'єднати в СНМ, коли додавали б елемент у структуру. Фактично ми просто моделюємо поведінку структури, не додаючи до неї елементів. Щоб моделювання було простіше, робимо це у порядку спадання елементів.

Можна було б оброблювати ці елементи інакше (схожим чином будемо робити в рішенні для онлайн-запитів). А саме: знайти перші $\sqrt{n}$ компонент у СНМ і далі проходитися по елементах з відрізка $[x, \min(r, y)]$ та ігнорувати елементи, що містяться не в перших $\sqrt{n}$ компонентах, бо вони вже не вплинуть на відповідь. Тобто фактично ми робимо меншу СНМ на $\sqrt{n}$ вершинах, коли відповідаємо на кожен запит, і її вже змінюємо так, як хочемо.

Таким чином ми навчилися відповідати на запити. Оцінимо складність роботи програми: не більше $\sqrt{n}$ разів ми додамо елемент в нашу велику структуру; для кожного запиту ми ще додатково розглядаємо $O(\sqrt{n})$ елементів, тому рішення має складність $O((n + q) \cdot \sqrt{n} \cdot \lambda(n))$.

Блоки 9, 10

Щоб відповідати на запити в онлайні, діятимемо схожим на офлайн чином (блок 8). Оскільки ми не знаємо запити наперед, то побудуємо

нашу структуру для кожного відрізку блоків (тобто беремо якийсь неперервний відрізок блоків і додаємо у структуру даних всі числа, що в ньому містяться).

Тепер, щоб відповісти на запит, треба додати не більше $2 \cdot \sqrt{n}$ елементів зліва та справа від відрізка блоків. Тоді зберігаємо $2 \cdot \sqrt{n}$ перших компонентів структури і відповідаємо на запити аналогічно до другого методу, описаного в рішенні для офлайну. Складність рішення зберігається, але зараз використовуємо $O((n + q) \cdot \sqrt{n})$ пам'яті.

ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

З дитинства пустеля у людини асоціюється з безмежною порожнечею, майже неживим простором піску та нищівного сонячного світла. Та варто зазирнути за лаштунки цього феномену природи, як відкриються дивовижні історії життя витривалих представників флори і фауни. Залежно від того піщана, кам'яна, глиняна, соляна чи снігова пустеля, в кожній з них свої закони виживання, свої герої тваринного та рослинного світів.

Людина ж з давніх-давен навчилась долати суворі безкраї простори пустелі, приймаючи її правила. Є народи, які традиційно проживають у пустелях, вирощують домашніх тварин і рослини, створюють оазиси серед пустель, будують сучасні міста й підприємства, задіюють природний потенціал пустель для отримання енергії.

Пропонуємо учасникам зануритись у світ природи та технологій, дізнатись про можливості, які дає людству пустеля, а під час виконання олімпіадних завдань сфокусуватись на аналітиці даних, побудові моделей і процесів, візуалізації та інтерактивному використанні даних.

І тур

До уваги учасників!

У розв'язку завдання дозволяється використовувати тільки файли з початковими даними, які розміщено у каталозі ***Для учасника***. ЗАБОРОНЕНО вставляти у ***файли-розв'язки*** зображення з ***файлів-зразків*** чи з ***файлів-інструкцій***. Використання редактора VBA та макрорекордера ЗАБОРОНЕНО за винятком вбудованих макрокоманд MS Access.

Перевірка розв'язків учасників передбачає зміну вхідних даних і перевірку результатів обчислень зі зміненими даними.

Всі завдання необхідно виконати за 4 години, зберегти на робочому столі ПК у каталозі з назвою **І_тур*_Прізвище_Ім'я_Область_Клас*** (наприклад, **І_тур*_Петренко_Олесь_Волинська_11***) і передати копію цієї папки під підпис члену журі.

Під час олімпіади учасники мають право ставити запитання виключно у письмовому вигляді. Запитання повинно бути сформульоване таким чином, щоб на нього можна було відповісти однозначно ТАК чи НІ.

У випадку, якщо одна із цих відповідей може стати підказкою або відповідь міститься у тексті завдання, учасники отримують відповідь БЕЗ КОМЕНТАРІВ.

У роботі залишати відомості, які ідентифікують особу учасника(ці) (за винятком назви каталогу з файлами-розв'язками), ЗАБОРОНЕНО!!!

Під час виконання завдань олімпіади учасникам заборонено користуватись будь-якими цифровими обчислювальними чи комунікаційними пристроями, крім персонального комп'ютера, наданого оргкомітетом на робочому місці.

Попереднє співвідношення балів: Access, Excel, Word, PowerPoint відповідно 25:35:15:25.

Задача «Dune»

Завдання виконується виключно засобами MS Access та MS Excel. Результати роботи учасники зберігають у файлі ***Dune_Character.accdb***.

«Через 20 тисяч років людство розпорошилось у Всесвіті, заселивши незліченну кількість планет. Після повстання мислездатних машин відбулася війна з ними, Батлеріанський Джигад. Люди здобули перемогу, і було накладено заборону на конструювання комп'ютерів та штучного інтелекту, але наука й технологія дуже розвинулися у сфері використання ресурсів людського організму. Місце комп'ютерів зайняли люди з високорозвинутим інтелектом – ментати» (Вікіпедія).

До розвитку подій за таким чи іншим сценарієм людству ще доведеться пройти свій шлях. Але відомі письменники-фантасти за багато років передбачали основні винаходи людства та їхній вплив на його подальшу долю.

Щодо пустель: Орегонські дюни поблизу міста Флоренс (штат Орегон, США) надихнули Френка Герберта на створення саги «Дюна», що стала найкультовішим фантастичним романом XX сторіччя, за яким протягом десятиріч створені не менш культові фільми, відео- та настільні ігри. Цей науково-фантастичний роман поза усім містить

сюжетну лінію про те, як головний герой перетворює планету з пустелями і недружніми племенами на гостинний світ.

Учасникам, суворо дотримуючись інструкцій файлу ***Інструкція Access.docx***, необхідно створити базу даних, що містить інформацію про персонажів цього науково-фантастичного роману.

Учаснику відповідно до наведених нижче інструкцій необхідно створити базу даних, що містить інформацію про персонажів науково-фантастичного роману американського письменника Френка Герберта «Дюна».

Інструкція

Підготовчий етап

Вихідні дані та проєкт бази даних містяться у файлах ***duneCharacters.xlsx*** та ***Dune_Character.accdb***.

При виконанні завдання заборонено змінювати типи полів і додавати інші поля у таблицях БД, додавати інші таблиці у БД.

Необхідно:

1) створити ключові поля у таблицях БД;

2) створити зв'язки з типами зв'язків між таблицями БД;

3) імпортувати дані з файлу ***duneCharacters.xlsx***.

Форма DUNE

Створити форму DUNE за зразком (рис. 1).

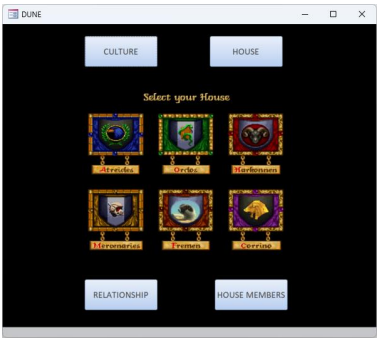


Рис. 1

Форма повинна відкриватися автоматично при відкритті бази даних.

На формі повинні бути розташовані кнопки: CULTURE, HOUSE, RELATIONSHIP та HOUSE MEMBERS.

При натисканні на кнопку CULTURE повинна відкриватися форма Culture, при натисканні на кнопку HOUSE – форма House, RELATIONSHIP – форма Relationship, HOUSE MEMBERS – звіт House Members.

Форма Culture

Створити форму Culture, зовнішній вигляд якої повинен відповідати зразку.

При відкритті користувач бачить пусту форму (рис. 2).

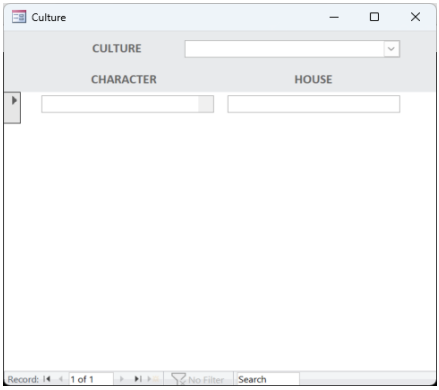


Рис. 2

На формі необхідно передбачити можливість вибору типу культури у полі CULTURE із таблиці БД, окрім значення Unknown (рис. 3).

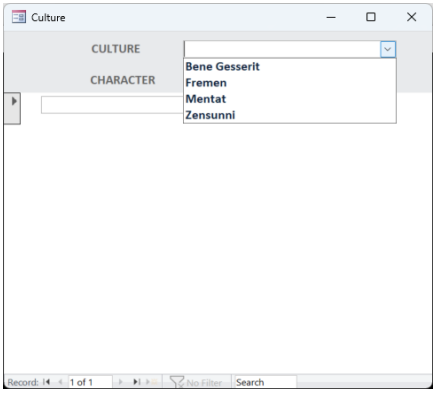


Рис. 3

Після цього у користувача повинна з'явитися можливість переглянути персонажів, які належать до обраної культури, і обрати дім, до якого вони належать (рис. 4).

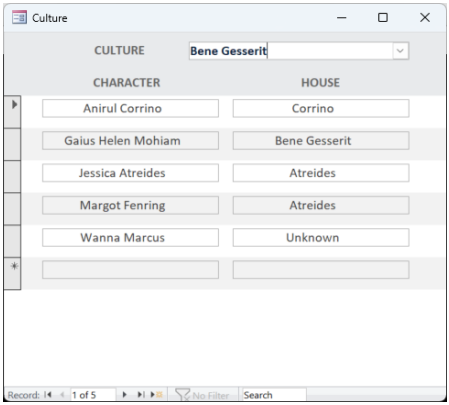


Рис. 4

Форма House

Створити форму House, зовнішній вигляд якої повинен відповідати зразку.

При відкритті користувач бачить пусту форму (рис. 5).

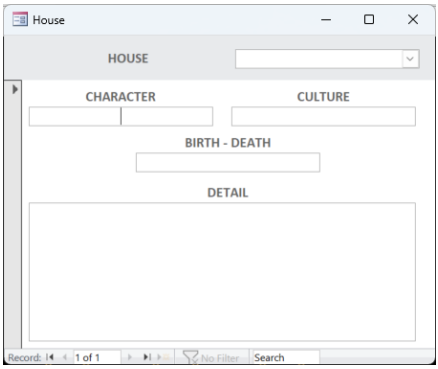


Рис. 5

На формі необхідно передбачити можливість вибору назви дому у полі HOUSE із таблиці БД (рис. 6).

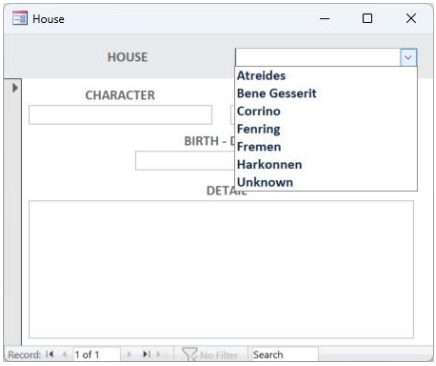


Рис. 6

Після цього у користувача повинна з'явитися можливість переглянути дані персонажів, які належать до обраного дому (рис. 7).

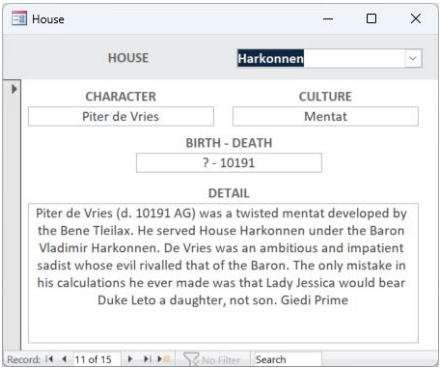


Рис. 7

Поле BIRTH – DEATH заповнюється таким чином: якщо дані відомі, то вони заповнюються із БД, якщо ні, то на їхньому місці ставиться знак питання.

Форма Relationship

Створити форму Relationship, зовнішній вигляд якої повинен відповідати зразку.

При відкритті користувач бачить пусту форму (рис. 8).

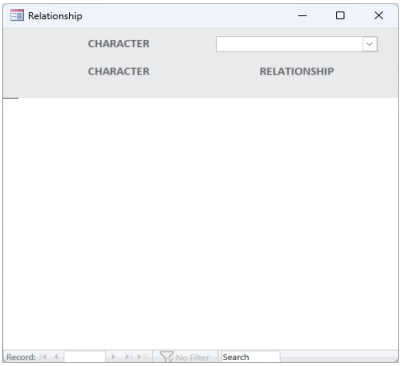


Рис. 8

На формі необхідно передбачити можливість вибору персонажа із таблиці БД (рис. 9).

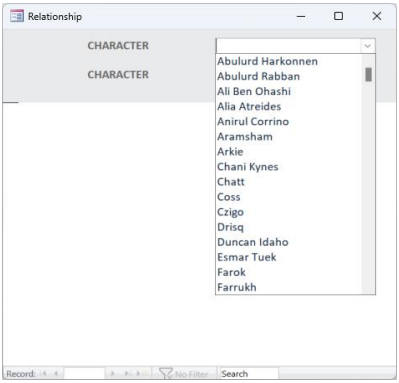


Рис. 9

Після цього у користувача повинна з’явитися можливість перегляду персонажів, чий взаємозв’язок з обраним відомий, а також вибрати вид зв’язку (рис. 10).

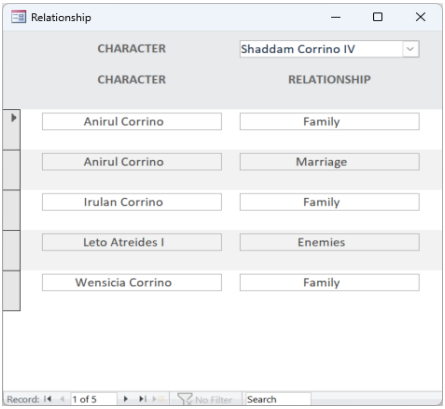


Рис. 10

Звіт House Members

Створити звіт House Members, зовнішній вигляд якого повинен відповідати зразку (рис. 11).

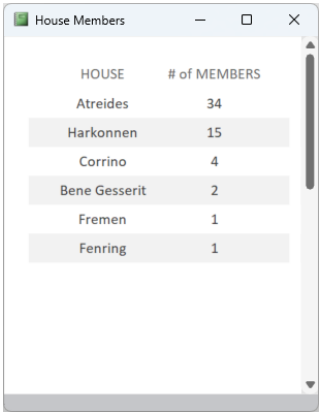


Рис. 11

У звіті повинна відображатися кількість персонажів кожного дому у порядку спадання.

(Матеріали взято з https://www.kaggle.com/datasets/bac3917/frank-herberts-dune-characters)

Завдання Excel

Завдання виконуються виключно засобами MS Excel.

Електронні таблиці зі своїми вбудованими можливостями дають сучасній людині змогу керувати процесами як у невеликих компаніях, так і у потужних корпораціях, проводити дослідження в науці та створювати інтерактивні освітні матеріали, візуалізувати діяльність установ й організацій, створювати прототипи моделей процесів. Перед учасниками олімпіади стоїть завдання максимально використати ці можливості у своїх розв'язках.

Задача «Deserts. Інфопанель»

Результати роботи учасники зберігають у файлі ***deserts_інфопанель.xlsx***.

Аналітика даних є одним із найпотужніших напрямів IT-галузі. Жодна серйозна бізнес-компанія не може сьогодні конкурувати на ринку, не вдаючись до аналітики процесів і результатів як своєї діяльності, так і конкурентів. Візуалізація даних, зібраних аналітиками, дає їм можливість глибше побачити результат досліджень, а компаніям і споживачам – наочно і швидко взяти з візуалізованих даних необхідну інформацію і прийняти виважені рішення.

Тож учасникам пропонується виступити в ролі аналітиків і створити інформаційну панель відповідно до інструкції (файл ***Інструкція dashboard.docx***).

Інструкція

Файл «deserts_інфопанель.xlsx».

Аркуш «Deserts». Набір даних аркуша містить інформацію про відомі пустелі світу.

Аркуш «Список країн за площею території» містить інформацію про площі країн світу (км^2).

Аркуш «Список країн світу за населенням» містить інформацію про населення країн світу (осіб).

Аркуш «Animals» містить інформацію про тварин пустель.

Аркуш «Plants» містить інформацію про рослини пустель.

Аркуш «Dashboard». На аркуші учасник повинен розмістити інформаційну панель (рис. 1). Дані для побудови містять аркуші

«Deserts», «Список країн за площею території», «Список країн світу за населенням», «Animals», «Plants».

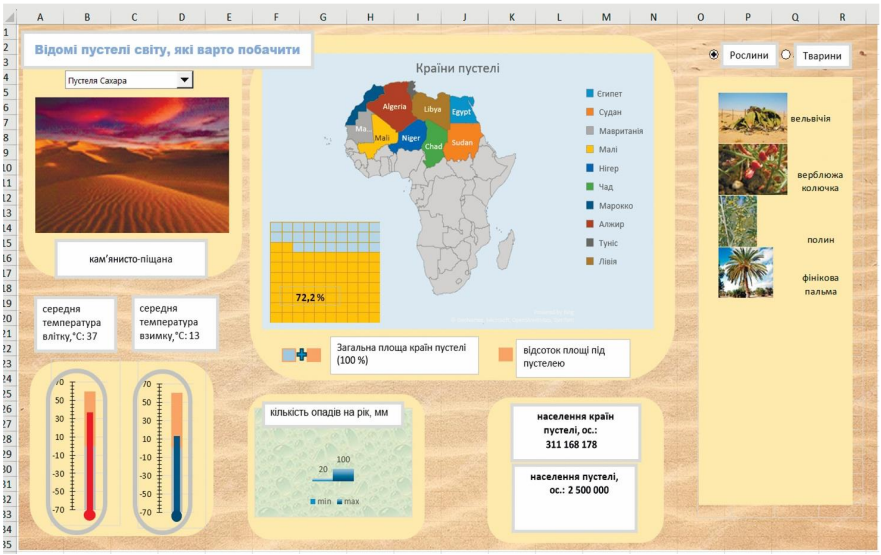


Рис. 1. Інформаційна панель

Панель складається з блоків (див. рис. 2).

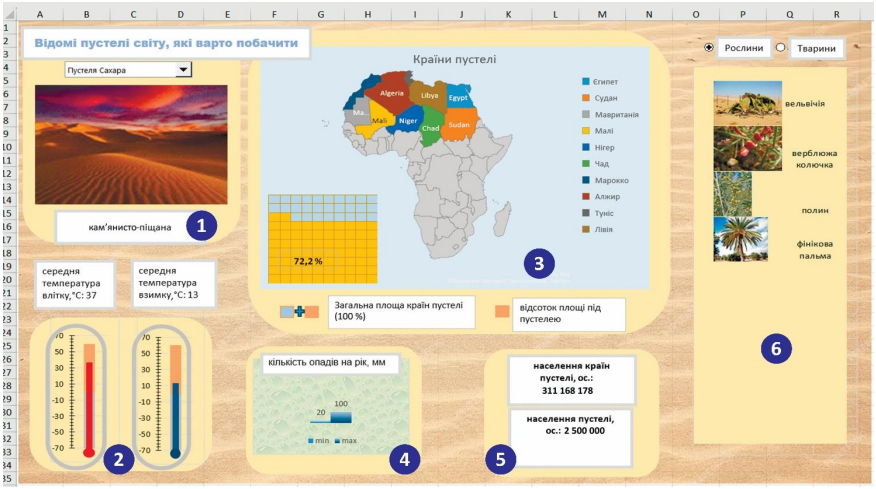


Рис. 2. Блоки інформаційної панелі

Блок 1 містить (рис. 3):

1) напис «Відомі пустелі світу, які варто побачити»;

2) розкривний список з назвами пустель;

3) зображення пустелі, що відповідає назві зі списку;

4) напис з типом відповідної пустелі зі списку.

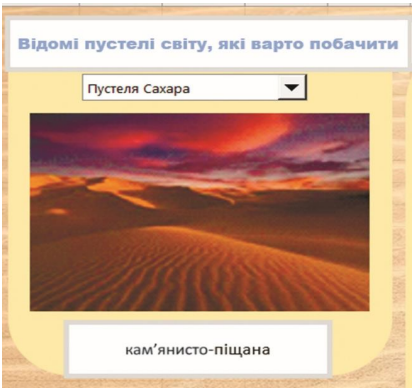


Рис. 3. Блок 1 інформаційної панелі

Блок 2 (рис. 4) містить:

1) два написи зі значеннями середньої температури вибраної пустелі взимку і влітку та пояснювальним текстом;

2) дві діаграми «термометр».

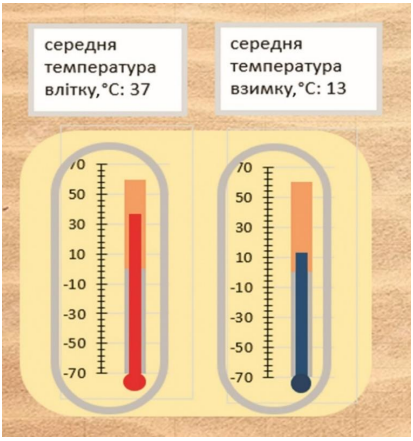


Рис. 4. Блок 2 інформаційної панелі

Блок 3 містить (рис. 5):

1) мапу з країнами, на території яких розташована обрана зі списку блоку 1 пустеля;

2) малюнок, що ілюструє у відсотках співвідношення загальної площі країн пустелі до площі відповідної пустелі;

3) пояснення до рисунка.

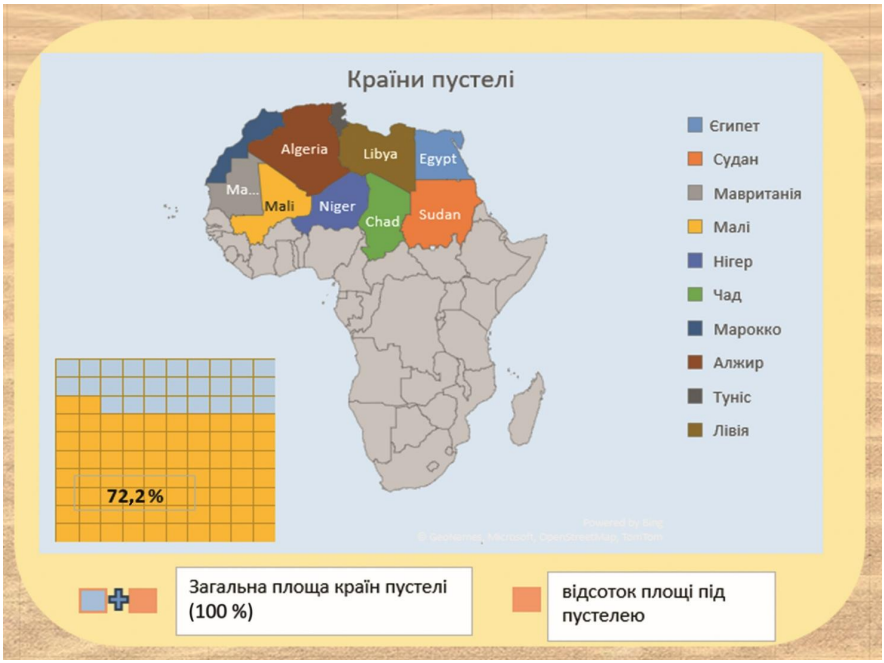


Рис. 5. Блок 3 інформаційної панелі

Блок 4 (рис. 6) містить діаграму, що ілюструє мінімальну і максимальну кількість опадів на рік у вибраній зі списку пустелі (мм).

Якщо маємо текстову інформацію, вона також повинна бути відображена на діаграмі (рис. 7).

Рис. 6. Блок 4 інформаційної панелі

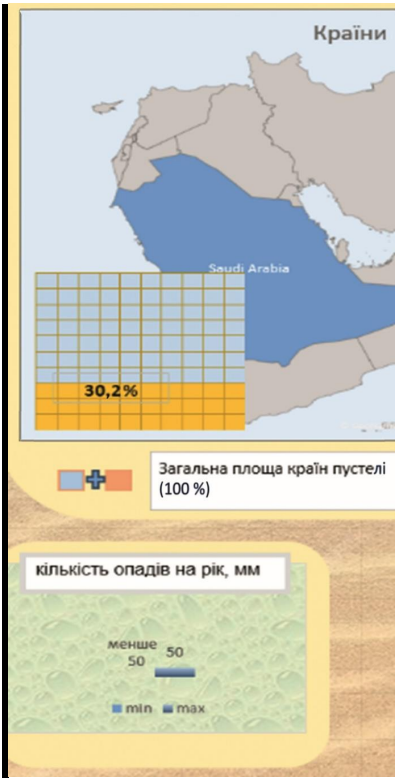


Рис. 7. Блок 4 інформаційної панелі

Блок 5 (рис. 8) містить у написах інформацію про загальну кількість населення країн пустелі й кількість населення, що мешкає в пустелі.

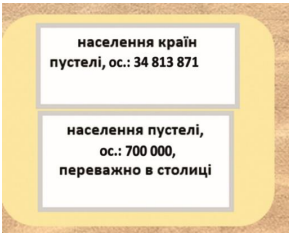


Рис. 8. Блок 5 інформаційної панелі

Блок 6 (рис. 9) містить два перемикачі:

1) при виборі перемикача «Рослини» з'являється малюнок із зображеннями і назвами деяких рослин вибраної зі списку пустелі;

2) при виборі перемикача «Тварини» з'являється малюнок із зображеннями і назвами деяких тварин, що мешкають у вибраній зі списку пустелі.

Рис. 9. Блок 6 інформаційної панелі

Учасник повинен створити інформаційну панель на аркуші «Dashboard» точно за зразком. Розміщення об'єктів, фон панелі, фон розташування фрагментів панелі, малюнки, написи, діаграми, керуючі елементи повинні відповідати зразку. Кількість додаткових аркушів не регламентується.

Наведемо приклад панелі для пустелі Руб-ель-Халі (рис. 10, 11).

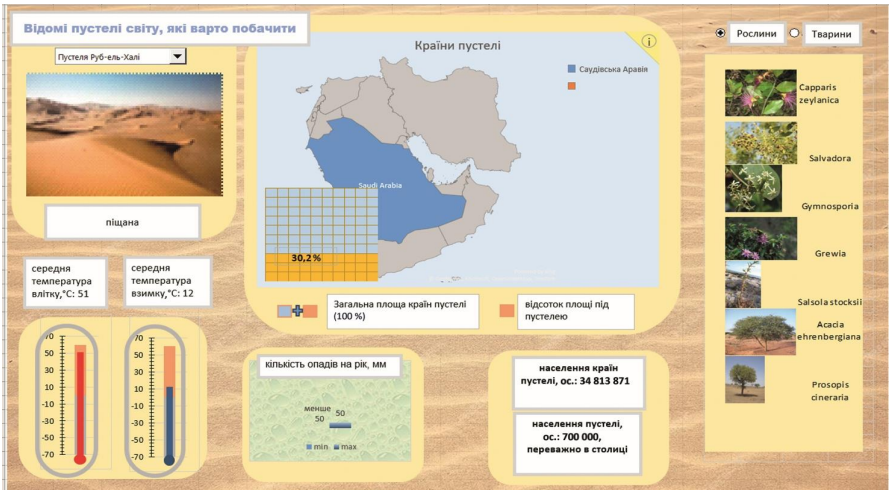


Рис. 10. Панель для пустелі Руб-ель-Халі

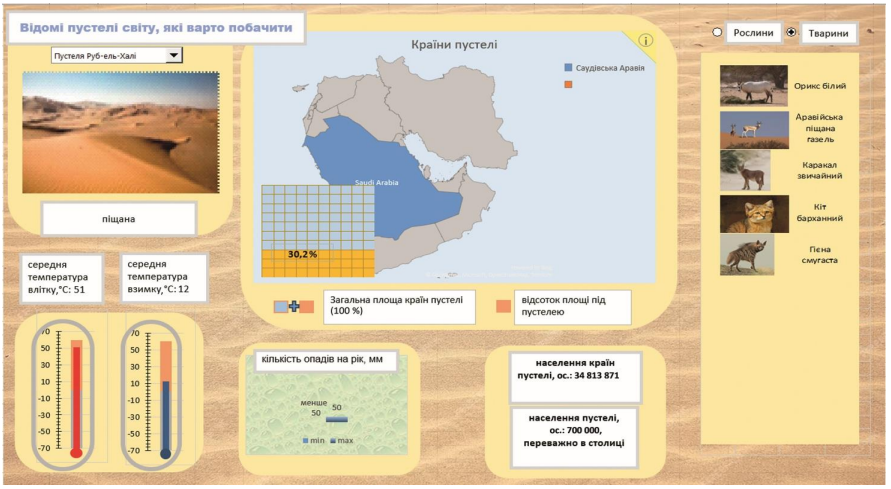


Рис. 11. Панель для пустелі Руб-ель-Халі

Задача «Deserts. Перегони»

Результати роботи учасники зберігають у файлі ***deserts_перегони.xlsx.***

Одним із популярних видів перегонів є перегони на верблюдах. Ці тварини тисячоліттями допомагають людині в суворих умовах пустельного клімату. Їхня витривалість давала людині шанс на подорожі довгими пустельними маршрутами з вантажами товарів. У сучасному світі їх давно замінили автомобілі та літаки, але збереглась і залишається популярною східна традиція перегонів на верблюдах.

Учасникам потрібно створити модель перегонів, у яких змагаються три вершники. Їм надано інструкцію (файл ***Інструкція перегони.docx***) та відеозразок (файл ***перегони.mp4***), неухильно дотримуючись яких потрібно розв'язати задачу.

Файл deserts_перегони.xlsx

Аркуш «Перегони»

Інструкція

У цьому завданні потрібно створити модель перегонів, у яких змагаються три вершники.

Аркуш «Перегони».

Початок гри. Учасникам надано зображення пустелі й вершників (на аркуші і в папці «Image»). Вершники відрізняються кольором головного убору та жилетки. Умовно називатимемо їх Синій, Червоний, Білий. На початку перегонів потрібно розташувати їх на старті (на одній лінії). Прапорець START вимкнено (рис. 1). Довжина перегонів – 8000 м.

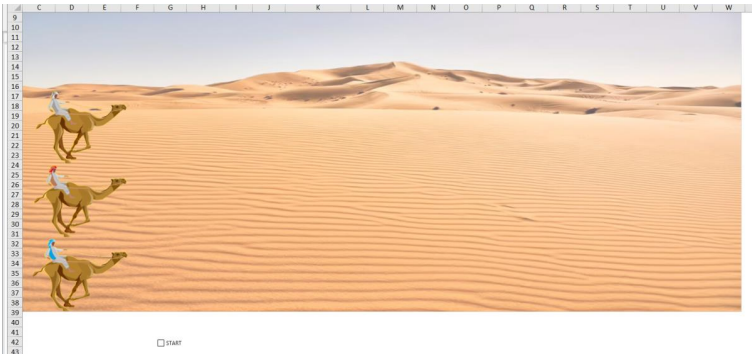


Рис. 1. Старт перегонів

Початок перегонів. Встановлюємо прапорець. Натискаємо F9. При кожному натисканні F9 кожен вершник переміщується на випадкову величину з інтервалу (50; 400). Наприклад, зроблено 16 натискань (рис. 2).

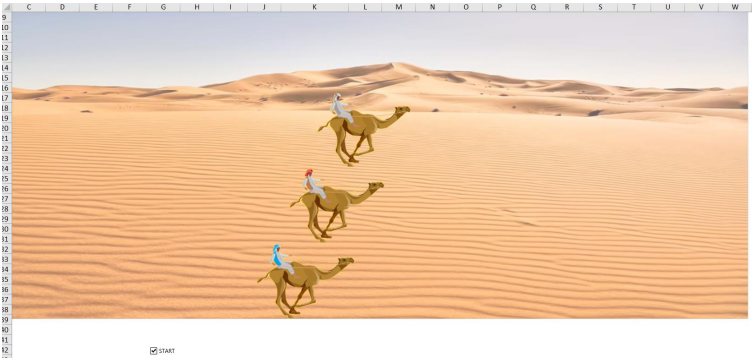


Рис. 2. Початок перегонів

Той вершник, чий верблюд першим досягне фінішу (пройде 8000 м), отримає в нагороду шаблю. Якщо кілька прийде одночасно, всі вони отримають шаблі.

Після досягнення фінішу хоча б одним учасником перегони припиняються (F9 не діє) (рис. 3).

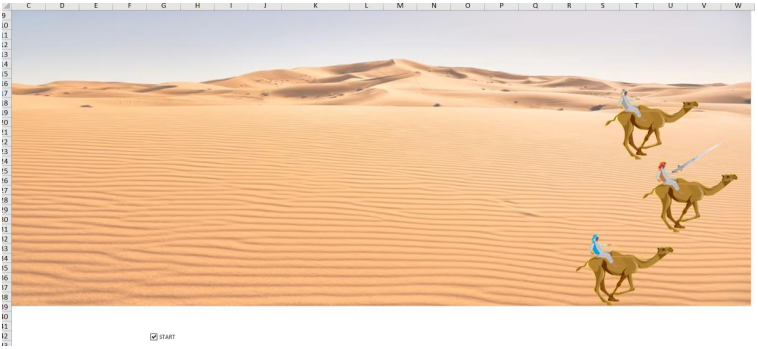


Рис. 3. Фініш перегонів

Завдання MS Word

Задача «Dune. Інтерактив»

Завдання виконується виключно засобами MS Word. Результати роботи учасники зберігають у файлах ***Дюна 2021.docx, Дюна 2024.docx*** *та* ***Екранізації.docx****. Додатково учасники мають здати і файли з описами персонажів* ***Владімір Харконнен.docx, Дункан Айдаго.docx, Звір Раббан.docx, Леді Джессіка.docx, Лето Атрід.docx, Пол Атрід.docx, Превелебна мати.docx, Чані.docx****.*

Пересічні люди сприймають текстовий процесор як пустелю з текстом і вставленими фото. Та якщо придивитись, то світ текстового процесора перетвориться на оазис з можливостями швидко підготувати макет книги чи журналу. І ще більше: зробити такий контент інтерактивним. Звичайно, для видання великих тиражів книг є спеціальні видавничі системи. Але можна зробити цікаву інтерактивну історію вдома чи в школі для використання в електронному вигляді, зберігаючи природу.

Пропонуємо учасникам олімпіади стати авторами такого маленького бестселера і виступити у ролі кіноглядачів. Учасники олімпіади точно за інструкцією (файл ***Інструкція Word.docx***) мають створити такі файли: ***Дюна 2021.docx***, ***Дюна 2024.docx*** та ***Екранізації.docx.***

Афіша до фільму «Дюна»

«Дюна» (англ. Dune) – американський фантастичний фільм 2021 року режисера Дені Вільньова, сценарій для якого написав він сам у співпраці з Джоном Спейтсом та Еріком Ротом. Це перша стрічка з двох у рамках запланованої адаптації однойменного роману Френка Герберта 1965 року.

Шляхетний дім Атрідів править дощовою планетою Каладан. Проте юний син герцога Лето, Пол, бачить сни про пустельну планету Арракіс та її жителів – фременів.

Впливову сім'ю Атрідів на чолі з герцогом Лето відправляють на Арракіс стежити за добуванням прянощів, найважливішої речовини у Всесвіті. Раніше ця планета належала войовничим Харконненам, які тероризували місцевий народ (фременів) та залишили після себе тільки ледве працюючу техніку. Вони не мають наміру залишати Арракіс – занадто багато грошей приносять місцеві піщані дюни…

Акторський склад:

Тімоті Шаламе – Пол Атрід, нащадок дому Атрідів;

Оскар Айзек – Лето Атрід, дворянин, нещодавно призначений намісником імператора на планеті Арракіс;

Ребекка Фергюсон – леді Джессіка, мати Пола, конкубіна герцога Лето;

Стеллан Скашгорд – Владімір Харконнен, барон дому Харконненів, ворог Лето, колишній намісник імператора на Арракісі;

Шарлотта Ремплінг – Превелебна мати Могіям, віщунка імператора;

Дейв Батиста – Звір Раббан, жорстокий племінник барона Харконнена;

Зендея – Чані, таємнича молода фременка;

Джейсон Момоа – Дункан Айдаго, зброяр дому Атрідів, вчитель фехтування Пола.

Афіша до фільму «Дюна: Частина друга»

«Дюна: Частина друга» (англ. Dune: Part Two) – епічний науково-фантастичний фільм 2024 року.

Пол – хлопець, який втратив всю свою сім'ю і все, що для нього було важливим, у війні із завойовниками Харконненами. Приєднавшись до пустельного народу, герой пізнає його таємниці та звичаї, намагаючись стати повноцінним членом групи.

Планета Арракіс нагороджує чоловіка великою честю осідлати хробака – тварину, яку місцеві шанують як бога. Тепер вигнанець стає лідером фременів. Вони стали його новою сім'єю, і лише корінні жителі планети, об'єднавшись під керівництвом молодого воїна, зможуть вигнати полчища окупантів.

Знайшовши кохання і прийнявши свою долю, справжній воїн вирушає у небезпечну та кровопролитну пригоду. Минуле і майбутнє переплетуться в одній реальності та зможуть допомогти подолати зло.

Акторський склад:

Тімоті Шаламе – Пол Муад'Діб, нащадок дому Атрідів;

Зендея – Чані, юна фременка, яка прагне свободи;

Ребекка Фергюсон – леді Джессіка, мати Пола, конкубіна покійного батька Пола;

Хав'єр Бардем – Стілґар, вождь фременського племені в Сітч-Таборі;

Флоренс П'ю – принцеса Ірулан, донька імператора;

Остін Батлер – Фейд-Раута Харконнен, молодший племінник Харконнена й очікуваний володар Арракіса;

Дейв Батиста – Звір Раббан, жорстокий племінник барона Харконнена.

Владімір Харконнен

Барон Владімір Харконнен виступає головним антагоністом «Дюни», котрому якраз протистоїть Лето Атрід. Голова дому Харконненів і правитель планети Г'єді Прайм став губернатором Арракіса ще до того, як планета була передана.

Дункан Айдаго

Зброяр дому Атрідів Дункан Айдаго також відіграє доволі значиму роль у становленні Пола – зокрема, він брав участь у його бойових тренуваннях. У трейлері можна побачити, що вони великі друзі. Привабливий герой користується популярністю у жінок, а також вважається майстерним бійцем. Також його як військового радника Лето цінують за дипломатичність, тому саме Дункан одним з перших вирушає на планету Арракіс, щоб налагодити відносини з місцевими жителями.

Звір Раббан

Звір Раббан – старший племінник барона Харконнена, він був губернатором Арракісу під час правління дядька. Жорстокий та не дуже проникливий силач протягом багатьох років знущався над фременами, які за садистські схильності злодія стали називати його Звіром. Барон поставив Звіра до влади, щоб той зробив фременів настільки нещасливими, що після зміщення Глоссу з посту та заміни на молодшого, більш здібного племінника Фейд-Рауту Владіміра Харконнена вважали б великодушним та мудрим правителем.

Леді Джессіка

Леді Джессіка ніколи не була дружиною Лето Атріда – її виховували як старшу наложницю герцога. Героїня повинна була чітко виконувати свої функції та народити доньок, щоб одну з них звести з Фейд-Раутою Харконненом – племінником Владіміра Харконнена. Але Джессіка та

Лето закохались одне в одного, героїня припинила дотримуватись вказівок та у пари народився хлопчик – Пол Атрід. До того ж у Джессіки було видіння, що саме син відіграє велику роль у якійсь великій справі.

Леді Джессіка є членом ордену Бене Гессерит, учасниці якого за допомогою містицизму та науки впливають на політичне становище. Вельми проникліва героїня вміє маніпулювати людьми, володіє бойовими навичками та гіпнозом.

Лето Атрід

Головою дому Атрідів виступає герцог Лето Атрід, батько Пола. Народ любить справедливого правителя та мудрого стратега з гострим розумом, але, як натякають у трейлері, зберігати владу йому стає все складніше. Через ворожнечу із домом Харконненів герой постійно зберігає пильність і намагається передбачати потенційні загрози з боку противника.

Пол Атрід

Головним героєм «Дюни» виступає обдарований молодий чоловік на ім'я Пол Атрід із дому Атрідів. Син герцога Лето Атріда та його конкубіни леді Джессіки. Пол виріс на планеті Каладан, де отримав аристократичне виховання. З дитинства він не тільки вивчав бойові мистецтва, а й активно розвивав свій розум і надприродні сили. Йому було призначено здійснити дещо велике заради майбутнього свого народу. А поки Пол Атрід розбиратиметься, у чому ж його доля, він також постарається зрозуміти суть своїх здібностей, які з віком стають все могутнішими. Підлітку 15 років, а вже така відповідальність!

Може здатись, що перед нами класичний «рятівник», який робить добро, але насправді все дещо інакше. Пол Атрід ще дуже юний і може припускатися помилок. Сам Тімоті Шаламе в інтерв'ю назвав персонажа антигероєм, який боїться величезної відповідальності та не дуже хоче мандрувати галактикою. Але хлопець все ж бере на себе відповідальність і стає лідером: «Він думає, що до свого повноліття набереться досвіду у батька, вивчить його спосіб управління, а потім стане молодим генералом. Можливо, десятиріччя потому».

Превелебна мати Могіям

Превелебна мати Гайя Могіям – наставниця леді Джессіки та важлива фігура у Бене Гессерит. За її наказом Джессіка повинна була народити Лето дівчаток, що допомогло б орденові посилити свій контроль. Але героїня не послухалася Превелебної матері. Утім, Могіям все одно цікавиться Полом Атрідом, який встиг проявити свій потенціал. Могіям вміє читати думки та проникати в розум.

Чані

Таємничу синьооку дівчину звати Чані. Вона фременка й живе на планеті Арракіс, владу над якою так прагнуть отримати Атріди та Харконнени. Колір очей Чані саме такий завдяки близькості до прянощів.

Пол Атрід одразу звертає увагу на героїню, оскільки бачив її у віщунському сні. Чані стає провідницею Пола до культури фременів. До речі, раніше Зендея (акторка, що зіграла Чані) зізналась, що її у фільмі не так вже й багато.

Завдання PowerPoint

Задача «Solar Tower Power Plant. Модель»

Завдання виконується виключно засобами MS PowerPoint. Результати роботи учасники зберігають у файлі ***Solar Tower Power Plant.pptx***.

Завдання учасників олімпіади: з наданих авторами файлів створити модель баштової сонячної електростанції суворо за зразком (файл ***Solar Tower Power Plant ЗРАЗОК.mp4***) та відповідно до інструкції (файл ***Інструкція Power Point.docx***).

Сучасний стан екології вимагає від людини припинення експлуатації теплових електростанцій через забруднення довкілля шкідливими викидами, відмови від побудови гідроелектростанцій через критичне порушення екосистеми річок. Натомість стоїть питання ширшого впровадження технології відновлювальної «зеленої енергії». Одним з найефективніших технологічних рішень є сонячні електростанції, які

економічно не залежать від наявності природних копалин і екологічно безпечні для довкілля. Логічно для розміщення таких електростанцій використовувати території з найбільшою кількістю сонячної енергії – пустелі. Баштова сонячна електростанція є однією з найефективніших електростанцій у співвідношенні ефективність / екологічність. Великою її перевагою перед електростанціями на сонячних панелях є можливість акумулювати надлишки енергії, тобто працювати і після заходу сонця.

В олімпіадному завданні розглядається типова ситуація розфокусування системи дзеркал. Для налаштування кутів віддзеркалення кожного із дзеркал системи використовується програмний комплекс управління електростанцією. Для розгляду переваг у обслуговуванні сонячної баштової електростанції створено демонстраційний проєкт. Учаснику пропонується доопрацювати модуль проєкту, наданий у файлі ***Solar Tower Power Plant.pptx***.

Інструкція

Сонячна електростанція – це інженерна споруда, що перетворює сонячну енергію (радіацію) на електричну енергію. Серед різновидів сонячних електростанцій особливу увагу привертають баштові електростанції.

Баштова сонячна електростанція складається з високої вежі, навколо якої розташовується геліополе з десятками тисяч модулів дзеркал (геліостатів). Кожен геліостат рухомий, за допомогою комп'ютера всі дзеркала обертаються за сонцем. Таким чином сонячні промені потрапляють до веж протягом усього світлового дня. Сфокусоване сонячне випромінювання нагріває в башті спеціальну рідину (в основному це сольові розчини). Через теплообмінник рідина нагріває воду, яка подається на турбіни для генерації електроенергії.

Великою перевагою геліотермальних електростанцій перед звичайними сонячними панелями є можливість акумулювати надлишки енергії, тобто працювати і після заходу сонця. Перегріта рідина накопичується в спеціальних об'ємних сховищах і вночі використовується для роботи турбін та виробництва електроенергії.

Учасникам пропонується:

1) проаналізувати зображення файлів ***Solar Tower Power Plant ЗРАЗОК.mp4*** та ***Solar Tower Power Plant.pptx*** і створити виключно

засобами вбудованого в PowerPoint графічного редактора об'єкти, яких не вистачає на слайді у файлі ***Solar Tower Power Plant.pptx***;

2) створити керовану анімацію процесу налаштування технологічного процесу.

При відтворенні анімації налаштування дзеркал дотримуватись законів відбивання світла: промінь, що падає, промінь відбитий і перпендикуляр до поверхні відбивання, проведений із точки падіння променя, лежать в одній площині, а кут відбивання дорівнює куту падіння. Величина потужності генерації електроенергії пропорційна кількості сонячної енергії, яка потрапляє на теплоприймач вежі. Візуалізується показами вимірювального приладу та швидкістю обертання ротора.

Ро́тор (англ. *rotor*; скорочення від *rotator*, пов'язаного з латин. *roto* – обертаюсь) – обертова частина машини (на відміну від нерухомої частини – статора). Зокрема, обертова частина електричної машини (такий ротор називають також якорем).

Рис. 1–2. Ашалімська електростанція (Ізраїль, пустеля Негев)

Задача «Solar Tower Power Plant. Алгоритм управління моделлю»

Результати роботи учасники зберігають у файлі ***Solar Tower Power Plant.pptx***.

Завдання учасників олімпіади: у створеній ними моделі баштової сонячної електростанції точно за зразком (файл ***Solar Tower Power Plant ЗРАЗОК.mp4***) та відповідно до інструкції (файл ***Інструкція Power Point.docx***) створити керовану анімацію регулювання

геліостатів і відповідної зміни розташування чи функціонування елементів зображення.

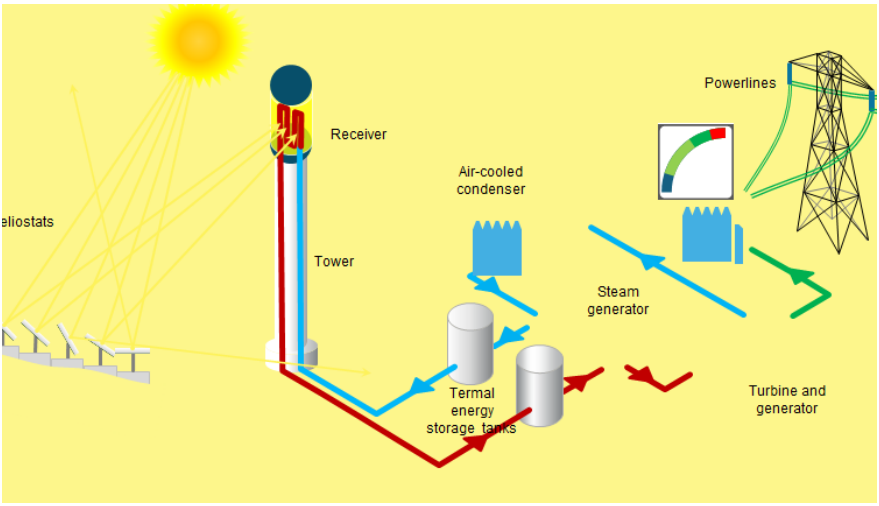


Рис. 1. Модель баштової сонячної електростанції

ІІ тур

До уваги учасників!

У розв'язку завдання дозволяється використовувати тільки файли з початковими даними, які розміщено у каталозі ***Для учасника***. ЗАБОРОНЕНО вставляти у ***файли-розв'язки*** зображення з ***файлів-зразків*** чи з ***файлів-інструкцій.*** Використання редактора VBA та макрорекордера ЗАБОРОНЕНО за винятком вбудованих макрокоманд MS Access.

Перевірка розв'язків учасників передбачає зміну вхідних даних та перевірку результатів обчислень зі зміненими даними.

Всі завдання необхідно виконати за 3 години, зберегти на робочому столі ПК у каталозі з назвою **ІІ_тур*_Прізвище_Ім'я_Область_Клас*** (наприклад, **ІІ_тур*_Петренко_Олесь_Волинська_11***) і передати копію цієї папки під підпис члену журі.

Під час олімпіади учасники мають право ставити запитання виключно у письмовому вигляді. Запитання повинно бути сформульоване таким чином, щоб на нього можна було відповісти однозначно ТАК чи НІ.

У випадку, якщо одна із цих відповідей може стати підказкою або відповідь міститься у тексті завдання, учасники отримують відповідь БЕЗ КОМЕНТАРІВ.

У роботі залишати відомості, які ідентифікують особу учасника(ці) (за винятком назви каталогу з файлами-розв'язками), ЗАБОРОНЕНО!!!

Під час виконання завдань олімпіади учасникам заборонено користуватись будь-якими цифровими обчислювальними чи комунікаційними пристроями, крім персонального комп'ютера, наданого оргкомітетом на робочому місці.

Попереднє співвідношення балів: Access, Excel, Excel відповідно 40:35:35.

Задача «Video Games»

Завдання виконується виключно засобами MS Access та MS Excel. Результати роботи учасники зберігають у файлі ***VideoGames_sales.accdb****.*

Відома гра «Dune» багато років очолювала рейтинги кращих ігор. Утім, час та індустрія створення ігор не стоять на місці.

Учасникам, неухильно дотримуючись інструкцій (файл ***Інструкція_VideoGames.docx***), необхідно створити базу даних, що міститиме інформацію про продажі відеоігор у світі.

Інструкція

Потрібно створити базу даних, що містить інформацію про продажі відеоігор у світі.

Вихідні дані містяться у файлі ***vg_sales.xlsx***, проєкт бази даних – у файлі ***VideoGames_sales.accdb***.

При виконанні завдання заборонено змінювати типи полів та додавати інші поля у таблицях БД, додавати інші таблиці у БД.

Необхідно:

1) створити ключові поля у таблицях БД;

2) створити зв'язки з типами зв'язків між таблицями БД;

3) імпортувати дані з файлу ***vg_sales.xlsx***;

4) створити запити, що знаходять:

– Query1: назви ігор, випущених заданим виробником (publisher) та заданого жанру (genre). Виробник та жанр задаються користувачем з клавіатури під час виконання запиту;

– Query2: назву гри (nameGames), платформу, на якій вона випущена (platform), та сумарну кількість продажів (sales) на всіх ринках (market). Назва гри або її частина задається користувачем з клавіатури під час виконання запиту;

– Query3: назви трьох кращих ігор за продажами на заданому ринку (market) у заданому році та платформу, на якій вони випускалися (platform), у порядку спадання продажів. Рік і ринок задаються користувачем з клавіатури під час виконання запиту;

– Query4: 10 % виробників (publisher), які випустили найбільшу кількість ігор на заданій платформі (platform), у порядку зменшення кількості. Назва платформи або її частина задається користувачем з клавіатури під час виконання запиту;

– Query5: назви п'яти платформ (platform) з найбільшою кількістю ігор та п'яти платформ з найменшою кількістю ігор у порядку спадання кількості ігор.

(Матеріали взято з https://www.kaggle.com/code/faisaljanjua0555/eda-video-games-sales/notebook)

Задача «Найбільші пустелі світу»

Завдання виконується виключно засобами MS Excel. Результати роботи учасники зберігають у файлі ***Найбільші пустелі.xlsx****.*

Пустелі займають більше однієї п'ятої площі поверхні Землі, і вони є на кожному континенті. Температура в пустелях становить від –60 в арктичних пустелях до +50 в піщаних.

Учасникам пропонується створити за допомогою засобів табличного процесора діаграму, яка відображатиме відношення площ найбільших пустель світу, обраних з наданого списку, відповідно до інструкції (файл ***Інструкція_Найбільші пустелі світу.docx***).

Інструкція

Місцевість, де випадає менше 25 см опадів на рік, вважається пустелею. У цих районах існує «дефіцит вологи», тобто вони можуть втрачати більше вологи через випаровування, ніж отримують від опадів. У найсухіших пустелях, таких як чилійська пустеля Атакама, випадає менше 2 мм опадів на рік. Таке середовище настільки суворе та потойбічне, що вчені навіть досліджували його, щоб знайти підказки про можливість життя на Марсі. З іншого боку, кожні кілька років надзвичайно дощовий період може привести до «суперцвітіння», коли навіть Атакама вкривається квітами.

Незважаючи на поширене уявлення про пустелі, як неймовірно жаркі місця, найбільшими пустелями є Арктична та Антарктична, де надзвичайно холодно більшу частину року. Найбільш гарячою пустелею є Сахара на півночі Африки, де температура вдень досягає 50 градусів за Цельсієм. Є гірські пустелі. Лише близько 20 % усіх пустель вкриті піском.

Вам пропонується створити за допомогою засобів табличного процесора діаграму, яка відображатиме співвідношення площ найбільших пустель світу, обраних з наданого списку.

Технічні умови:

– інтерфейс має відповідати наданим зразкам;

– таблиця має містити аркуші «Вибір», «Діаграма», «Розрахунки» та «Дані»;

– на аркуші «Вибір» здійснюється вибір необхідних об'єктів елементами керування;

– на аркуші «Діаграма» відображається діаграма;

– на аркуші «Розрахунки» виконуються необхідні розрахунки;

– на аркуші «Дані» розміщуються використовувані дані щодо найбільших пустель;

– перехід між аркушами «Вибір» та «Діаграма» здійснюється за допомогою кнопок;

– якщо кількість обраних об'єктів більша ніж 10, виводиться повідомлення «Занадто багато обрано!»;

– при виборі об'єкта / скасуванні вибору відбуваються відповідні зміни у списку об'єктів та на діаграмі;

– стовпчики діаграми мають показувати правильне відношення площ обраних пустель;

– кожен стовпчик має бути квадратом (або схожим на квадрат), стовпчики повинні мати різні кольори;

– діаграма повинна мати легенду, яка відображає відповідність кольору стовпчика назві об'єкта та містить його площу;

– необхідні для виконання дані містяться у файлі ***Найбільші пустелі світу.xlsx***;

– зразки зображень і відео містяться у папці «Зразки»;

– необхідні зображення містяться у папці «Зображення»;

– результат виконання завдання зберегти у файлі ***Найбільші пустелі.xlsx***.

Таблиця 1

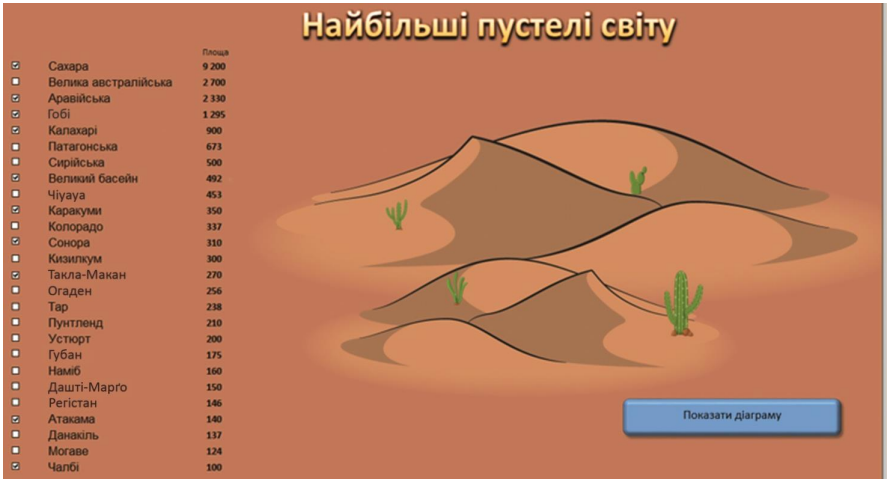
Вхідні дані

Назва пустелі	**Площа (тис. км²)**	**Площа (км²)**	**Розташування**	**Країни**
Сахара	9200	9 200 000	Східна, Північна та Західна Африка	Алжир, Чад, Єгипет, Еритрея, Лівія, Мавританія, Марокко, Нігер, Судан, Туніс, Західна Сахара
Велика австралійська	2700	2 700 000	Австралія	Австралія

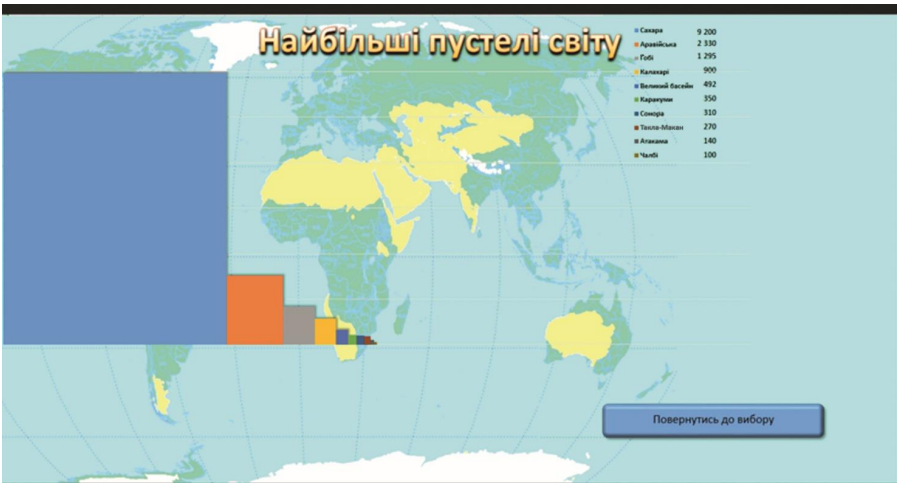
Аравійська	2330	2 330 000	Західна Азія	Ірак, Йорданія, Кувейт, Оман, Катар, Саудівська Аравія, Об'єднані Арабські Емірати, Ємен
Гобі	1295	1 295 000	Східна Азія	Китай, Монголія
Калахарі	900	900 000	Південна Африка	Ботсвана, Намібія, Південна Африка
Патагонська	673	673 000	Південна Америка	Аргентина
Сирійська	500	500 000	Західна Азія	Ірак, Йорданія, Саудівська Аравія, Сирія
Великий басейн	492	492 098	Північна Америка	США
Чіуауа	453	453 248	Північна Америка	Мексика, США
Каракуми	350	350 000	Центральна Азія	Туркменістан
Колорадо	337	337 000	Північна Америка	США
Сонора	310	310 000	Центральна та Північна Америка	Мексика, США
Кизилкум	300	300 000	Центральна Азія	Казахстан, Туркменістан і Узбекистан
Такла-Макан	270	270 000	Східна Азія	Китай
Огаден	256	256 000	Східна Африка	Ефіопія, Сомалі, Сомаліленд
Тар	238	238 000	Південна Азія	Індія та Пакистан
Пунтленд	210	210 000	Східна Африка	Сомалі
Устюрт	200	200 000	Центральна Азія	Казахстан, Туркменістан, Узбекистан

Губан	175	175 000	Східна Африка	Сомалі, Сомаліленд
Наміб	160	160 000	Середня та Південна Африка	Ангола, Намібія, Південна Африка
Дашті-Марго	150	150 000	Південна Азія	Афганістан
Регістан	146	146 000	Південна Азія	Афганістан
Атакама	140	140 000	Південна Америка	Чилі, Перу
Данакіль	137	137 000	Східна Африка	Джибуті, Еритрея, Ефіопія
Могаве	124	124 000	Північна Америка	США
Чалбі	100	100 000	Східна Африка	Кенія

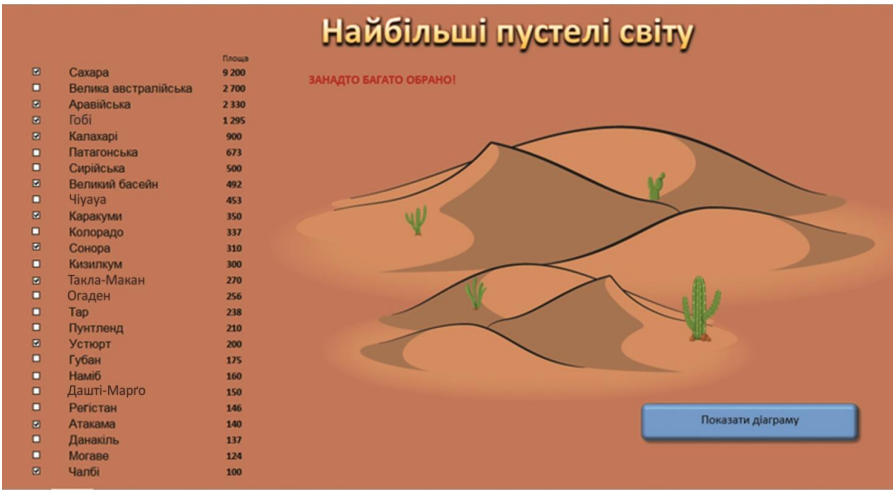
Зразок 1



Зразок 2



Зразок 3



Зразок 4

Завдання «Подорож Намібією»

Завдання виконується виключно засобами MS Excel. Результати роботи учасники зберігають у файлі ***Подороож.xlsx***.

Пропонуємо вам вирушити у подорож до незвичайної Намібії, основну територію якої складають пустелі та плоскогір'я. Пропонуємо відвідати поселення, пустелю, національні парки та інші пам'ятні місця.

У межах завдання необхідно відтворити систему, яка візуалізує подорож цією цікавою країною, суворо дотримуючись інструкції (файл ***Інструкція_Подороож.docx***).

Рис. 1. Плато Соссусфлей

Вхідні дані

На аркуші «Точки» у діапазоні A5:G19 вказані назви, координати та малюнки 15 точок, праворуч від точок розташовано мапу країни Намібія. Всі зображення, необхідні для розв'язання завдання, також можете знайти у папці «Фотографії».

На аркуші «Маршрути» у діапазоні C2:H11 розташовані 9 маршрутів, які складаються з 5 точок (початок, 3 проміжні точки та кінець).

Під час перевірки журі вхідні дані НЕ ЗМІНЮЄ.

Завдання

На аркуші «Подорож» необхідно реалізувати строго за прикладом наведену систему (приклад роботи показано у відео «Робота системи»).

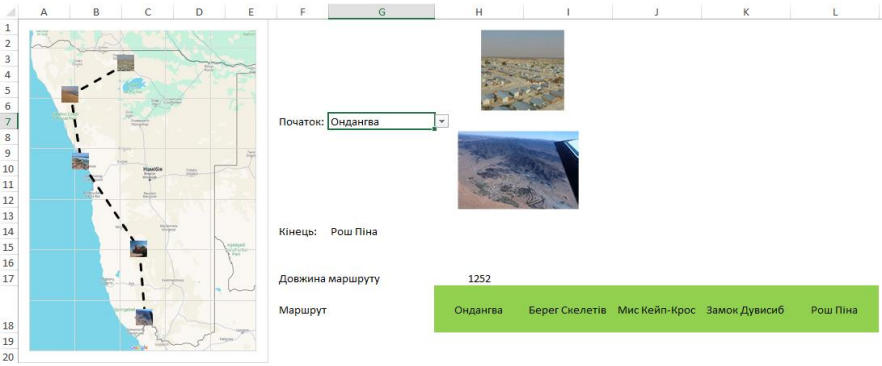


Рис. 2. Зразок аркуша «Подорож»

У комірках G7 та G14 вибрати відповідно точки початку та закінчення маршруту зі спливаючого списку.

Рис. 3. Зразок вибору точок початку та закінчення маршруту

Праворуч від надписів розташовані фотографії початку та кінця маршруту, які відповідають зображенню обраної точки.

Рис. 4. Зразок поєднання точки початку маршруту та її зображення

Рис. 5. Зразок поєднання точок початку й кінця маршруту
та їхніх зображень

У комірках H18:L18 відображається маршрут, кінець та початок якого відповідають обраним значенням, 3 проміжні точки відповідають обраному маршруту з аркуша «Маршрут» за початком та кінцем, гарантується, що існує єдиний маршрут для обраної пари.

Початок: Ондангва

Кінець: Ораньємунд

Довжина маршруту 1488

Маршрут	Ондангва	Наскельні малюнки	Національний парк Ватерберг	Каньйон Фіш-Рівер	Ораньємунд

Рис. 6. Зразок відображення маршруту відповідно до вибраних точок початку й кінця

У діапазоні A1:E2 розташована діаграма, на якій зображено мапу країни й обраний маршрут у вигляді зображень точок і пунктирної лінії пройденого маршруту (у порядку початок – точка 1 – точка 2 – точка 3 – кінець). Розташування точок відповідає реальному розташуванню на мапі (наприклад, міста Опуво та Ораньємунд).

Початок: Опуво

Кінець: Ораньємунд

Довжина маршруту

Маршрут

Рис. 7. Зразок зображення маршруту на мапі країни

При зміні початку або кінця маршрут на мапі змінюється на відповідний.

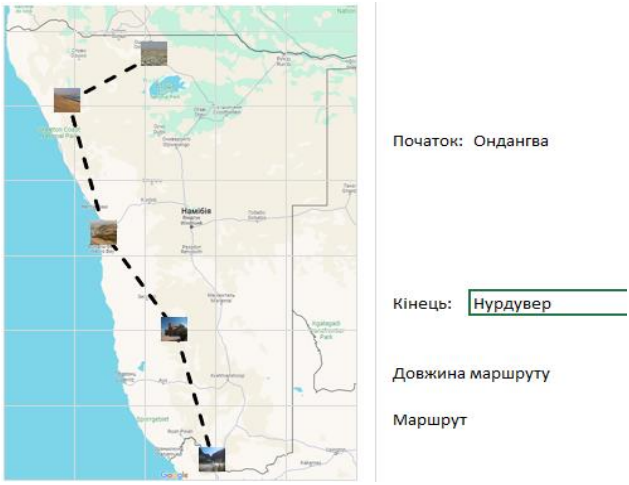


Рис. 8. Зразок зображення маршруту на мапі країни

У комірці H17 підраховується довжина маршруту.

Примітка. Відстань між точками A(x_1, y_1) та B(x_2, y_2) розраховується за формулою: $S_{AB} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

Але не всім цікаво відвідувати всі три пам'ятки під час подорожі. Тому система має надавати можливість вилучати проміжні точки з маршруту. При натисканні на одну з назв проміжних точок цей пункт вилучається з маршруту – відповідна комірка більше не виділяється зеленим.

Рис. 9. Зразок вилучення однієї точки з маршруту

Якщо точку маршруту вилучити, то вона зникає з діаграми. В довжині маршруту цю точку також не враховують.

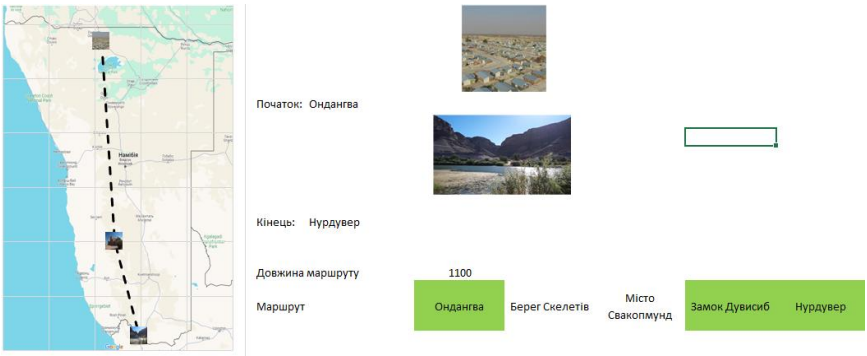


Рис. 10. Зразок вилучення двох точок з маршруту

Рис. 11. Зразок вилучення кількох точок з маршруту

Повторне натискання на точку повертає її до маршруту. Початок та кінець вилучити з маршруту не можна.

Примітка. Додавання / вилучення точки з маршруту можна реалізувати альтернативним способом, додавши над або під назвою кожної з трьох точок список із значеннями «так», «ні». Але реалізація вибору точок за допомогою списку *оцінюється в меншу кількість балів!*

Таблиця 1

Вхідні дані

№	Назва	Координати		Зображення
1	Опуво	216	-149	
2	Ондангва	430	-134	
3	Рунду	808	-136	
4	Національний парк Етоша	475	-201	
5	Берег Скелетів	180	-258	

6	Наскельні малюнки	269	-400	
7	Мис Кейп-Крос	228	-522	
8	Національний парк Ватерберг	566	-379	
9	Місто Свакопмунд	285	-611	
10	Каньйон Фіш-Рівер	591	-1115	
11	Місто-привид Колманскоп	355	-1015	

12	Замок Дувисиб	487	-870	
13	Нурдувер	594	-1216	
14	Ораньємунд	475	-1199	
15	Рош Піна	509	-1140	

Рис. 12. Мапа Намібії

(Матеріали взято з http://teg.com.ua/top-12-pamyatki-namibiyi/)

Список літератури

Інформатика

1. Використання Powtoon для створення анімаційного відео. URL: http://surl.li/erroi (дата звернення: 02.06.2024).
2. Геометрична руханка-релакс. URL: https://youtu.be/A80CAAp6l7M?si=Toi166ssnL0OMiLp (дата звернення: 01.07.2024).
3. Збірник задач та розв'язань із програмування / С. М. Бондаренко, В. В. Зуб, О. І. Коваленко та ін. ; за заг. ред. О. Є. Баранової, Ю. М. Літоша. Чернігів : ЧОІППО імені К. Д. Ушинського, 2016. Ч. 1. 40 с. URL: http://choippo.cn.sch.in.ua/Files/downloadcenter/%D0%A0%D0%BE%D0%B7%D0%B2%D1%8F%D0%B7%D0%BA%D0%B8%2010-12.2015_%D0%BE%D1%81%D1%82%D0%B0%D0%BD_0.pdf (дата звернення: 01.06.2024).
4. Збірник задач та розв'язків II етапу Всеукраїнської учнівської олімпіади з інформатики 2014–2015 навчального року / авт.-упоряд.: С. М. Бондаренко, В. В. Зуб, Ю. М. Літош. Чернігів, 2015. URL: https://drive.google.com/file/d/0BzXzAlavBkWxX29fVHpUemlNSEE/view?resourcekey=0-Ra-8_yXnTCY2t6gpx22rEg (дата звернення: 01.06.2024).
5. Кукурудз С. Ф., Процюк В. Р., Ваврик Т. О. Збірник задач з програмування : навч. посіб. Івано-Франківськ : Факел, 2005. 247 с.
6. Онлайн-інструмент для створення діаграм. URL: https://nulab.com/cacoo/ (дата звернення: 01.06.2024).
7. Петрова О. О., Солодовник Г. В. Алгоритмічні задачі та їх вирішення : навч. посіб. / Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. Харків : ХНУМГ ім. О. М. Бекетова, 2021. 105 с.
8. Сервіс PowToon – чудова програма для презентацій. URL: http://surl.li/errph (дата звернення: 01.07.2024).
9. Як комп'ютерні ігри впливають на дітей, підлітків та дорослих? URL: http://surl.li/errlj (дата звернення: 01.07.2024).
10. 12 інтерактивних онлайн-дошок для дистанційного навчання та спільної роботи. Освіта Нова. URL: https://osvitanova.com.ua/posts/4181-12-interaktyvnykh-onlain-doshok-dlia-dystantsiinoho-navchannia-ta-spilnoi-roboty (дата звернення: 01.07.2024).

Інформаційні технології

1. Анісімов А. В., Кулябко П. П. Інформаційні системи та бази даних : навч. посіб. для студ. факультету комп'ютерних наук та кібернетики. Київ, 2017. 110 с.

2. Антоненко В. М. Мамченко С. Д., Рогушина Ю. В. Сучасні інформаційні системи і технології: управління знаннями : навч. посіб. Ірпінь : Національний університет ДПС України, 2016. 212 с.
3. Костріков С. В., Сегіда К. Ю. Географічні інформаційні системи : навч.-метод. посіб. Харків, 2016. 82 с.
4. Морзе Н. В., Піх О. З. Інформаційні системи : навч. посіб. / за наук. ред. Н. В. Морзе. Івано-Франківськ : Лілея-НВ, 2015. 384 с.
5. Оцінка рівня цифрової грамотності за допомогою інструменту Цифрограм на порталі Дія. Освіта. URL: https://osvita.diia.gov.ua/digigram (дата звернення: 28.06.2024).
6. Соколов В. Ю. Інформаційні системи і технології : навч. посіб. Київ : ДУІКТ, 2010. 138 с.
7. Сучасні інформаційні системи і технології: конспект лекцій / В. Г. Іванов, С. М. Іванов, В. В. Карасюк та ін. ; за заг. ред. В. Г. Іванова, В. В. Карасюка. Харків : Нац. юрид. ун-т ім. Ярослава Мудрого, 2014. 347 с. URL: https://dspace.nlu.edu.ua/bitstream/123456789/6421/1/Konspekt_leksiy_95.pdf (дата звернення: 28.05.2024).
8. Штучний інтелект. Як він вплине на освіту. URL: https://nus.org.ua/articles/shtuchnyj-intelekt-yak-vin-vplyne-na-osvitu/ (дата звернення: 01.07.2024).
9. 11 технологій штучного інтелекту, які допоможуть зробити навчання ефективнішим. URL: https://osvitanova.com.ua/posts/5953-11-tekhnolohii-shtuchnoho-intelektu-iaki-dopomozhut-zrobyty-navchannia-efektyvnishym (дата звернення: 01.06.2024).

Відомості про авторів

ІНФОРМАТИКА

Горошко Юрій Васильович	завідувач кафедри інформатики та обчислювальної техніки Національного університету «Чернігівський колегіум» імені Т. Г. Шевченка, доктор педагогічних наук, професор;
Гриненко Валерій Владиславович	студент Харківського національного університету радіоелектроніки;
Денисов Костянтин Ігорович	студент Харківського національного університету радіоелектроніки;
Савчук Костянтин Ростиславович	студент Київського національного університету імені Тараса Шевченка;
Столітній Андрій Сергійович	студент Ягеллонського університету (м. Краків, Польща)

ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Бондік Ірина Георгіївна	учителька інформатики Дніпровського наукового ліцею інформаційних технологій Дніпровської міської ради, вчителька-методистка;
Варзар Євген Анатолійович	учитель інформатики Природничо-наукового ліцею № 145 Печерського району м. Києва;
Громко Григорій Юрійович	учитель інформатики Комунального закладу «Нечаївський ліцей ім. Ю. І. Яновського» Компаніївської селищної ради Кропивницького району Кіровоградської області;
Кузічев Микола Миколайович	консультант з інформатичної галузі Комунальної установи «Центр професійного розвитку “Освітня траєкторія”» Дніпровської міської ради;
Кушнір Віталій Анатолійович	учитель фізики, математики, інформатики Капустянської гімназії Новоушицької селищної ради Хмельницької області;
Ніколаєв Тарас Геннадійович	учитель Дніпровського ліцею № 100 «Лідер» Дніпровської міської ради, менеджер Центру розвитку талантів IT-компанії SoftServe
Шаравара Ольга Валеріївна	учителька Дніпровського ліцею № 100 «Лідер» Дніпровської міської ради

Навчальне видання

Завдання IV етапу
Всеукраїнських учнівських олімпіад
з навчальних предметів

2023/2024 навчальний рік

ІНФОРМАТИКА

ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Практикум

Редагування: *Н. І. Гетьман, І. В. Братащук*
Дизайн обкладинки *О. А. Чекановська*

У виданні з навчальною метою
використані матеріали, в тому числі ілюстративні,
що містяться у вільному доступі в мережі Інтернет

Формат 60×84 1/16. Папір офс. 80 г/м^2.
Друк цифровий. Ум. друк. арк. 5,58.
Наклад 300 прим.

Видавництво: Національний центр «Мала академія наук України»,
Кловський узвіз, буд. 8, м. Київ, 01021
Свідоцтво суб'єкта видавничої справи:
ДК № 6999 від 04.12.2019