Задача A. Тест

Відповідь дорівнює $b - c$.

Задача B. Прогулянка

Достатньо розглянути усі $3! = 6$ способів обрати шлях між магазинами. Отже, відповідь буде дорівнювати

$$min(|a|+|a-b|+|b-c|, |a|+|a-c|+|c-b|, |b|+|b-a|+|a-c|, |b|+|b-c|+|a-c|, |c|+|a-c|+|b-a|, |c|+|c-b|+|b-a|)$$

Але є інший, красивіший розв'язок.

Нехай $a \leqslant b \leqslant c$, тоді є наступні випадки:

1. $c \leqslant 0$ або $a \geqslant 0$ — відповідь буде дорівнювати $|a|$ або $|c|$ відповідно.
2. Інакше, одна точка лежить між двома іншими, і на шляху до іншої точки ми її відвідаємо. Тобто, відповідь — $\min(2 \cdot |c| + |a|, 2 \cdot |a| + |c|) = |a| + |c| + \min(|c|, |a|)$.

Задача C. OSU!

Створимо декілька змінних:

- $count\_miss$ — скільки промахів поспіль було до цього кроку
- $current\_score$ — рахунок поточної гри
- $count\_games$ — кількість ігор
- max_score — максимальний рахунок серед всіх ігор
- new_game — змінна буде вказувати, чи ми починаємо нову гру на наступний крок.

Тепер треба обробити рахунки, нехай ми зараз зчитали рахунок $score$, тоді:

- якщо $new\_game = 1$, то треба додати 1 до $count\_games$ і зробити $new\_game = 0$.
- якщо $score = 0$, то треба додати 1 до $count\_miss$.
- якщо $score = 100$ чи $score = 300$, треба додати $score$ до $current\_score$ та занулити $count\_miss$.
- якщо $count\_miss = 3$, то додаємо 1 до $count\_games$, робимо $max\_score = \max(max\_score, current\_score)$ та зануляємо $count\_miss$ та $current\_score$.

Також після зчитування всіх балів, треба вкінці оновити max_score та $count\_games$ враховуючи останню гру.

Задача D. Не час для ігор

Нехай ans_t — відповідь на запит з часом t. Порахуймо всі значення ans. Будемо йти від більшого t до меншого. В момент часу $t = n - 2$ маємо $ans_{n-2} = a_{n-1} + a_n$. Надалі можна виражати відповідь для часу t як максимум серед відповіді для $t + 1$ і значення пари елементів a_{t+1} та a_{t+2}, тобто, $ans_t = \max(ans_{t+1}, a_{t+1} + a_{t+2})$ для усіх $0 < t < n - 2$, а $ans_0 = \max(ans_1, a_1 + a_2, a_1 + a_n)$ треба розглянути як окремий випадок. Тепер будемо зчитувати запити та відповідати на них виводячи значення ans_t.

Асимптотика рішення — $\mathcal{O}(n + q)$.

Задача E. Мовні проблеми

Будемо перебирати відповідь за допомогою бінарного пошуку. Якщо ми можемо утворити пари використовуючи j волонтерів, то зможемо утворити якщо візьмемо $j+1$ волонтерів. Як можна перевірити, що можна утворити пари, щоб виконувались $a_i + p_i \geqslant x$, де p_i — рівень знань волонтера у парі з i-ю людиною? Відсортуймо масив a за зростанням, і зробимо масив B, в який покладемо перші j волонтерів з масиву b. Відсортуємо масив B по спаданню та візьмемо перші n елементів. Тепер залишилось перевірити, що виконується $a_i + B_i \geqslant x$ для усіх i від 1 до n.

Доведемо, що це завжди буде оптимально. Припустимо, що ми маємо 4 індекси $i \leqslant k < t \leqslant j$, і дві пари (a_i, B_j) та (a_t, B_k), для яких виконується $a_i + B_j \geqslant x$ та $a_t + B_k \geqslant x$. Доведемо, що також підходять пари (a_i, B_k) та (a_t, B_j). За умовою маємо, що $a_i \leqslant a_t$ та $B_k \geqslant B_j$, розпишемо нерівності:

$$a_i + B_k \geqslant a_i + B_j \geqslant x$$

$$a_t + B_j \geqslant a_i + B_j \geqslant x$$

В обох нерівностях ми замінили більший елемент на менший для більшої сторони та отримали нерівність, яка правда. Отже, існує оптимальне розбиття на пари, для яких не існує пар (a_i, B_j), (a_t, B_k) $(i \leqslant k < t \leqslant j)$, а отже, парування (a_i, B_i) є одним з оптимальних.

Таке рішення матиме асимптотику $\mathcal{O}((n+m) \cdot \log^2 m)$, що недостатньо швидко, але при гарних оптимізаціях мало змогу пройти усі тести.

Подумаємо, як можна прибрати зайвий $\log m$ з асимптотики. Він з'являється з сортування масиву B. Як отримувати масив B вже відсортованим? Створимо вектор, який буде зберігати пари (b_i, i). Посортуємо його по спаданню. Надалі ми зможемо створювати масив B ітеруючись по вектору і перевіряючи, чи належить цей елемент до тих, які ми розглядаємо, тобто $i \leqslant j$.

Це оптимізує рішення до асимптотики $\mathcal{O}((n+m) \cdot \log m)$, що достатньо, щоб не перевищити ліміт часу.

Задача F. Іспит

Рішення, яке працює для $n \leqslant 20$. Будемо перебирати елементи, які візьмемо у множину за 2^n за допомогою біт-масок (схожа ідея була у другій задачі першого туру III етапу Всеукраїнської олімпіади з програмування). Коли ми обрали множину, будемо знаходити її значення побітового АБО, мінімум та максимум — val, mn та mx відповідно. Якщо $\texttt{popcount}(\text{val}) \geqslant k$ будемо оновлювати відповідь: $ans = \min(ans, mx - mn)$.

Рішення, яке працює для $a_i < 1024$. Можна відштовхуватися від того, що існує 1024 різних значень a_i. Залишимо по одному входженню кожного значення в масив a (якщо воно було) та відсортуємо масив за зростанням. Побачимо, що на такому масиві нам вигідно обирати його підвідрізкі, а не підпослідовності. Припустимо, що ми обрали два елементи з мінімальним та максимальним значеннями — a_l та a_r відповідно. Тобто, його ціна вже є фіксованою. Якщо ми будемо брати елементи, які не будуть збільшувати максимум, чи зменшувати мінімум, його ціна не зміниться. Очевидно, що $\texttt{popcount}(a|b) \geqslant \max(\texttt{popcount}(a), \texttt{popcount}(b))$, де $|$ позначає побітове АБО. А отже, можна брати усі елементи з індексами $l \leqslant i \leqslant r$.

Будемо перебирати усі підвідрізки та оновлювати відповідь за $\mathcal{O}(m^2)$, де m — розмір масиву після видалення повторюваних елементів.

Повне рішення використовує ідею з попереднього рішення. Відсортуємо масив за зростанням та будемо розглядати його підвідрізки. Можна помітити, що якщо не підходив відрізок $(l; r)$, то відрізок $(l+1; r)$ теж не буде підходить. Це натякає на два вказівники, але як рахувати значення побітового АБО на відрізку? З означення побітового АБО, i-й біт буде дорівнювати 1, якщо він дорівнює 1 хоча б в одному числі, серед яких береться побітове АБО. Можемо зберігати для кожного біта скільки разів він зустрічається на відрізку, а значення побітового АБО можна відновити з цих значень — якщо цей біт зустрічався на відрізку, то цей біт буде й у значенні побітового АБО. Зсуваємо праву границю вправо, доки відрізок не підходить, оновлюємо відповідь і видаляємо найлівіший елемент з відрізка.

Отже, асимптотика — $\mathcal{O}(n \log A)$, де A — максимальне значення a_i.

Автор усіх задач: Андрій Столітній.