Задача A. Фотографія

Автор: Андрій Холод
Роздробник: Павло Цікалишин
Розбір підготовано: Костянтин Денисов

Фотографія має розмір $n \times n$. Рамка товщини t додає смугу шириною t з кожного боку, тому зовнішній розмір усієї конструкції дорівнює $(n + 2t) \times (n + 2t)$. Площа рамки дорівнює різниці площ зовнішнього квадрата та фотографії.

Площа фотографії: $S_{\text{in}} = n^2$.
Площа зовнішнього квадрата: $S_{\text{out}} = (n + 2t)^2$.
Отже, площа рамки: $S = S_{\text{out}} - S_{\text{in}} = (n + 2t)^2 - n^2$.

Спростимо вираз: $(n + 2t)^2 - n^2 = n^2 + 4nt + 4t^2 - n^2 = 4t(n + t)$.

Тому відповідь: $S = 4t(n + t)$.

Складність. Час: $O(1)$, пам'ять: $O(1)$.

Зауваження. За $n \leqslant 10^9$ значення площ може бути порядку 10^{18}, тому слід використовувати 64-бітний цілий тип (наприклад, `long long`).

Задача B. Футбол

Автор: Андрій Холод
Роздробник: Павло Цікалишин
Розбір підготовано: Костянтин Денисов

Щоб забити гол, достатньо влучити в будь-яку точку воріт, тобто мінімізувати $|x - x_v| + |y - y_v|$ по всіх $y \in [y_1, y_2]$.

Частина $|x - x_v|$ не залежить від y, тому мінімізувати треба лише $|y - y_v|$ на відрізку $[y_1, y_2]$. Мінімальна відстань від точки y_v до відрізка $[y_1, y_2]$ дорівнює:

- 0, якщо $y_1 \leqslant y_v \leqslant y_2$,
- $y_1 - y_v$, якщо $y_v < y_1$,
- $y_v - y_2$, якщо $y_v > y_2$.

Це можна записати одним рядком як $d_y = \max(0,\ y_1 - y_v,\ y_v - y_2)$.

Тоді відповідь: $p_{\min} = |x - x_v| + d_y$.

Складність. Час: $O(1)$, пам'ять: $O(1)$.

Задача C. Пікнік

Автор: Андрій Холод
Роздробник: Павло Цікалишин
Розбір підготовано: Костянтин Денисов

Потрібно розбити масив $a_1, \ldots, a_n$ на $m = k + 1$ **послідовних** частин так, щоб сума в кожній частині була однакова, і використати всі шматки. Нехай $S = \sum_{i=1}^{n} a_i$. Тоді кожна частина має вагу $T = S/m$, отже обов'язково $m \mid S$.

Через те, що $n \leqslant 5000$, можна зробити дуже просту перевірку: перебрати всі можливі m від найбільшого до 1 і перевірити, чи можна нарізати на m частин сумою T жадібно зліва направо. Оскільки ми знаємо наперед суму в кожній частині, то далі перший відрізок визначається однозначно, за ним другий і так далі.

Перевірка для фіксованого m. Обчислюємо $T = S/m$ і йдемо по масиву, накопичуючи cur та підраховуючи кількість друзів cnt:

- $cur := cur + a_i$.
- якщо $cur = T$, то "закриваємо"частину: $cnt := cnt + 1$, $cur := 0$;
- якщо $cur > T$, то таке m неможливе (бо всі $a_i \geqslant 1$ і ми вже перебрали суму).

В кінці потрібно, щоб $cur = 0$ і $cnt = m$.

Складність. Перебираємо m не більше ніж n разів, кожна перевірка за $O(n)$, отже загалом $O(n^2)$. За $n \leqslant 5000$ це проходить. Пам'ять $O(n)$.

Задача D. Комфорт Василька

Автор: Андрій Холод
Роздробник: Павло Цікалишин
Розбір підготовано: Костянтин Денисов

Потрібно знайти найдовший підвідрізок масиву $a_1, \ldots, a_n$, для якого $\max(a_l, \ldots, a_r) - \min(a_l, \ldots, a_r) \leqslant k$. Це класична задача на два вказівники з підтримкою поточного мінімуму і максимуму в вікні.

Якщо фіксувати праву межу r і збільшувати l, то max не зростає, а min не спадає, отже різниця $\max - \min$ не збільшується. Тому можна рухати два вказівники: розширювати вікно вправо, а коли умова порушується — зсувати ліву межу, поки знову не стане добре.

Проблема: потрібно швидко знати min і max у поточному вікні $[l, r]$. Для цього підтримуємо `std::multiset` на елементах поточного вікна.

Алгоритм завжди підтримує інваріант: після внутрішнього циклу вікно $[l, r]$ є комфортним. Для кожного r ми мінімізуємо l (зсуваємо його лише поки треба), тому довжина $r - l + 1$ є максимальною серед усіх комфортних підвідрізків, що закінчуються в r. Максимум по всіх r дає відповідь задачі.

Складність. Кожен індекс додається в множину рівно один раз і видаляється не більше одного разу, тому загальний час $O(n \log n)$, пам'ять $O(n)$.

Задача E. Гра

Автор: Костянтин Денисов
Роздробник: Павло Цікалишин
Розбір підготовано: Костянтин Денисов

Очевидно, що на відрізку $[l, r]$ треба ставити якнайбільші числа, тобто $a_i = m$ для всіх $l \leqslant i \leqslant r$. Тоді цей відрізок має найбільший потенціал, і далі наша задача — зробити так, щоб жоден інший підвідрізок не міг з ним зрівнятися за сумою, та ще й серед усіх таких масивів максимізувати суму невід'ємних елементів.

Окремий випадок $l = r$. Тут максимальна сума має бути рівно m і досягатися **лише** на позиції l. Тому поза l не можна ставити m, інакше одиночний підвідрізок з цієї позиції теж матиме суму m. Отже поза l найкраще ставити $m - 1$. Але треба ще зіпсувати всі підвідрізки, які включають l і ще щось: для цього достатньо розставити $-m$ на кожній другій позиції, починаючи від сусідів $l - 1$ та $l + 1$. Тоді будь-яке розширення відрізка за межі $[l, l]$ неминуче підхопить хоча б один $-m$, і сума стане строго меншою за m.

Випадок $l < r$.

Поза відрізком $[l, r]$ оптимальна конфігурація має максимально багато m, але такі m доводиться ставити через один, чергуючи з $-m$; інакше з'являється занадто великий підвідрізок поза $[l, r]$ або підвідрізок, що "приклеюється"до $[l, r]$.

Тому для **префікса** довжини $s = l - 1$ оптимально ставити (зліва направо):

- якщо s **парна**, то $m, -m, m, -m, \ldots, m, -m, m - 1, -m$; доводиться ставити одну $m - 1$, щоб сума елементів була від'ємна, бо інакше буде декілька елементів з однаковою максимальною сумою.
- якщо s **непарна**, то $m, -m, m, -m, \ldots, m, -m, -m$.

Для **суфікса** довжини $t = n - r$ робимо аналогічну (дзеркальну) побудову за парністю t.

У будь-якому допустимому масиві поза $[l, r]$ не можна ставити багато великих додатних підряд: кожен додатний m змушений мати поруч достатню "компенсацію"$-m$, інакше з'являється підвідрізок з занадто великою сумою, який конкурує з $[l, r]$. Отже m можна ставити не частіше, ніж через один, тобто шаблон $m, -m, m, -m, \ldots$ дає максимальну кількість m. У парному випадку рівно один раз доводиться замінити m на $m - 1$, щоб уникнути рівності максимумів.

Складність. Побудова займає $O(n)$ часу і $O(1)$ додаткової пам'яті (окрім виводу).