Задача A. Борги

Початково в Аліси a гривень. Коли Петрик віддасть їй свій борг у b гривень, то в Аліси стане $a+b$ гривень. Коли Світлана віддасть Алісі свій борг у c гривень, то в Аліси стане $a+b+c$ гривень. Останньою дією Аліса віддасть Ромчику борг у d гривень, після чого її баланс стане рівним $a+b+c-d$ гривень.

Відповідь на задачу: $a+b+c-d$

Задача B. Середнє арифметичне

За формулою середнього арифметичного $c = \frac{a+b}{2}$, тоді домноживши обидві частини рівняння на два отримаємо $2c = a+b$. Віднявши від обох частин рівняння a отримаємо $2c-a = b$, з чого й випливає відповідь на задачу.

Відповідь: $2c-a$

Задача C. Петрик та екзамен

З умови задачі випливає, що Петрик успішно вирішив $a-x$ легких задач з a легких задач та $b-y$ складних задач з b складних задач. Нехай прості задачі коштують по c балів, тоді складні задачі коштують по $2c$ балів. Порахувавши бали Петрика, отримаємо $(a-x)\cdot c + (b-y)\cdot 2c$ балів, а максимальна кількість балів, яку можливо отримати дорівнює $a\cdot c + b\cdot 2c$. Розрахуємо співвідношення кількості балів, що отримав Петрик до максимальної кількості балів:

$\frac{(a-x)\cdot c+(b-y)\cdot 2c}{a\cdot c+b\cdot 2c}\cdot 100\% = \frac{(a-x)+(b-y)\cdot 2}{a+b\cdot 2}\cdot 100\%$

Після цього завдяки оператору розгалуження у випадку, якщо цей вираз є меншим за 51%, то слід вивести «NO», інакше «YES»

Примітка: під час реалізації задач краще уникати операцій з нецілими числами через втрату точності. Так, наприклад, у цій задачі можна помітити що умову складання іспиту можна переписати у наступному вигляді:

$100\cdot((a-x)+(b-y)\cdot 2) \geqslant 51\cdot(a+b\cdot 2)$

Задача D. Транспортна дилема

Позначимо за T_b, T_m та T_t загальний час, який займе дорога автобусом, метро та таксі відповідно. Тоді $T_b = d_b + t_b + w_b$, $T_m = d_m + t_m + w_m$, $T_t = d_t + t_t$. Позначимо за m мінімум серед T_b, T_m та T_t, тобто $m = min\{T_b, T_m, T_t\}$. Тоді шлях i буде оптимальним по часу, якщо $T_i = m$. Застосуємо наступний алгоритм:

1. Якщо $T_b = m$, то слід їхати автобусом, бо по часу це найоптимальніший варіант та найдешевший.

2. Інакше перевіримо чи $T_m = m$. Якщо так, то оберемо метро як відповідь. Оскільки минулий пункт не був виконаний, то $T_b > m$, а значить дешевшого оптимального транспорту за метро немає.

3. Інакше оберемо таксі, бо оскільки два минулі пункти не були виконані, то $T_b > m$, $T_m > m$, а значить дешевшого оптимального транспорту не існує.

Задача E. Скрутні часи

Вирішимо задачу за допомогою симуляції розкладу цього місяця, підраховуючи кількість зроблених задач кожного дня. Для кращого розуміння нижче наведений код. Позначимо поточний день за d, початково він рівний 1. Тоді у циклі поки d не перевищує кількість днів у місяці будемо підраховувати відповідь. Врахуємо перші a днів, розглянувши послідовно кожен день: якщо d кратне n, то це день хаотичного відпочинку, бо кожен n-й день є таким, відповідно у цей день ніякої роботи зроблено не буде. Інакше цей день є робочим. У такому випадку перевіримо, чи він є днем подвійної ефективності завдяки аналогічній перевірці на кратність числа d числу m, якщо воно кратне, то збільшимо відповідь на 2, інакше тільки на 1, бо день був звичайним. Оскільки були оброблені a днів, то наступні b днів є гарантовано вихідними. Тоді просто пропустимо їх збільшивши d на b, бо ніякої роботи у цей період не буде.

```
int res = 0;
int d = 1;
while (d <= k) {
    for (int i = 0; i < a && d <= k; i++) {
        if (d % n == 0) {
            res += 0;
        } else if (d % m == 0) {
            res += 2;
        } else {
            res += 1;
        }
        d++;
    }
    d += b;
}
```

Складність рішення: $\mathcal{O}(k)$

Задача F. До чого ЗНО доводить...

Навчимось рахувати кількість задач, яку розв'яже учасник, бо, знаючи це, можна однозначно визначити, який буде рахунок та хто буде переможцем. Розглянемо n наборів задач по a_i задач у кожному наборі та максимальну ефективність e. Помітимо, що якщо під час вирішення якогось набору він не був повністю розв'язаний, а ефективність стала нульовою, то наступні набори вже не будуть розв'язані ні на одну задачу. Таким чином задача зводиться до пошуку першого набору, який не можна розв'язати та підрахунку задач, що вийшло розв'язати.

Порахуємо скільки максимально можливо розв'язати задач з одного набору при даному e. У першу годину розв'язування набору буде розв'язано e задач, у другу – $e-1$, у третю – $e-2$... і так можна розв'язувати поки e не стане рівне 0. Кількість розв'язаних задач дорівнює сумі цієї послідовності, а оскільки це є арифметичною прогресією, то за формулою її суми отримаємо, що максимальна кількість задач з одного набору, яку можливо розв'язати рівна $\frac{e(e+1)}{2}$.

З цього отримали рішення: можемо послідовно йти по наборах задач поки кількість задач у кожному $\leqslant \frac{e(e+1)}{2}$ та додавати кількість задач з цих наборів до лічильника розв'язаних задач. Якщо кількість задач у поточному наборі більша за $\frac{e(e+1)}{2}$, то це буде останній набір з якого вийде розв'язати останні $\frac{e(e+1)}{2}$ задач.

Складність рішення: $\mathcal{O}(n+m)$

Задача G. Дорога в нове життя

У цієї задачі існує доволі багато рішень з використанням різних методів та відповідно різною асимптотичною складністю. Ми опишемо розв'язок, який використовує найбільшу кількість корисних в олімпіадах підходів.

Першою дією, оскільки заправки у вхідних даних подаються у довільній послідовності, то відсортуємо їх за незростанням x_i.

Розглянемо цю задачу за умови фіксованого розміру бака та позначимо його значення s. Навчимось рахувати мінімальну кількість грошей необхідну для поїздки при заданому s. Для цього скористаємось методом динамічного програмування позначивши за $dp[i]$ мінімальну кількість грошей необхідну, щоб доїхати до i-тої заправки. Оскільки перша заправка знаходиться у точці старту, то $dp[0] = 0$. Помітимо, що щоб дорога до заправки i була оптимальною по витратах, автівка має доїхати до заправки без залишку бензину у баці. Тоді якщо минула заправка, де автівка заправилась, мала номер $j < i$, то $dp[i] = dp[j] + w * (x_i - x_j) * c_j$. Проте для таких j підходять тільки такі j, що $w * (x_i - x_j) \leqslant s$, бо інакше розміру бака не вистачило б щоб доїхати до заправки i. Тоді можемо послідовно порахувати значення $dp[i]$. Для легкого отримання відповіді перед обчисленням додамо ще одну заправку з номером $n + 1$ для якої $x_{n+1} = d$. Тоді мінімальна кількість грошей необхідна для поїздки рівна $dp[n + 1]$.

Оскільки при збільшенні розміру бака ціна поїздки буде не зростати, то порахуємо мінімальну ціну поїздки поклавши $s = \infty$ та позначимо цю ціну за p. Тоді за допомогою бінарного пошуку по s знайдемо таке мінімальне s, що ціна поїздки з баком розміру s рівна p. У результаті це мінімальне s і буде відповіддю на задачу.

Складність рішення: $\mathcal{O}(n^2 logn)$

Задача H. Шахова проблема

Шахову дошку $8 \cdot 8$ можна зберігати та розглядати як двовимірний масив символів, назвемо його b. Для зручності реалізації нумерацію масиву можна взяти стандартну для задач на таблиці: від 0 до 7 зліва направо та від 0 до 7 знизу вгору. Тоді перед виводом відповіді слід конвертувати такий формат координат у формат окреслений в задачі.

Створимо функцію $isKingInCheck$, що приймає параметром масив b та повертає 1 якщо король перебуває під шахом, а 0 інакше. Першим чином у цій функції знайдемо координати клітинки короля та назвемо їх $(king_i, king_j)$. Для написання цієї функції можна створити додатковий двовимірний масив $isUnderAttack$ такий, що $isUnderAttack[i][j] = 1$, якщо клітинка (i, j) перебуває під атакою якоїсь чорної фігури, інакше ж $isUnderAttack[i][j] = 0$. Початково масив $isUnderAttack$ слід ініціалізувати нулями. Далі під час лінійного проходу по всім елементам b будемо оновлювати $isUnderAttack$ враховуючи фігуру що була розглянута. Тобто, наприклад, якщо під час цього лінійного проходу при поточних i та j відомо що $b[i][j] =$'p', то змінимо значення $isUnderAttack[i-1][j-1]$ та $isUnderAttack[i-1][j+1]$ на 1, якщо ці клітинки існують. Зрештою як результат функції можна повернути значення $isUnderAttack[king_i][king_j]$.

Ми навчились обраховувати функцію $isKingInCheck$ для довільної конфігурації дошки, тоді обрахуємо її для усіх можливих переміщень короля(таких положень максимум 8 + перше початкове). Якщо в усіх конфігураціях король знаходиться під шахом, то значить поточна ситуація є матом. Інакше, якщо в усіх конфігураціях, що є наслідками ходу короля, король знаходиться під шахом, то ця ситуація є патом. Якщо і перша, і друга умова описані вище не виконуються, то маємо справу зі звичайним положенням.

Автор усіх задач: Данило Тимошенко.