КОНТРОЛНО СЪСТЕЗАНИЕ
НА РАЗШИРЕНИЯ НАЦИОНАЛЕН ОТБОР
София, 5 – 6 май 2026 г.
Група С, 7 – 8 клас

Задача СК12. СУМИ (Анализ на решението)

⏳ 1 сек. 💾 256 MB

Нека с `pos[x]` означим позицията на монетата с номинал x. Ще търсим тези позиции и накрая ще върнем редицата, зададена от `ans[pos[x]] = x`. Позициите са индексирани от 0, а номиналите са числата от 1 до N.

Първа подзадача

Тук $N = 3$. Ако попитаме за една монета, отговорът е 2 точно когато нейният номинал е 1; във всички други случаи е 1. Така намираме позицията на единицата. След това към нея добавяме една от останалите монети. За номинал 2 първата непостижима сума е 4 и получаваме 0, а за номинал 3 получаваме 2. Третата монета се определя еднозначно.

Брой въпроси: 5 в дадената имплементация.
Имплементация: `sums_kiril_6p.cpp`

Междинни разсъждения

За следващите подзадачи е полезно да разберем как се намира най-малката непостижима сума. Нека с вече разгледаните монети можем да получим всички суми от 0 до S. Ако следващата по големина монета е $c \leq S + 1$, с нейното добавяне можем да получим всички суми до $S + c$. Ако $c > S + 1$, сумата $S + 1$ остава непостижима и по-големите монети няма да помогнат.

Ще наричаме помощно множество такъв набор от вече разпознати монети, чиито суми покриват целия интервал от 0 до общата им сума S. Да добавим към него една неизвестна монета с номинал x. Ако $x \leq S + 1$, първата непостижима сума става $S + x + 1$. В противен случай тя остава $S + 1$. Разбира се, ако получената стойност е над N, функцията връща 0.

Втора подзадача

Тук $N = 70$. Един подход е да намираме номиналите последователно, като проверяваме всяка все още неизвестна позиция. За първите 10 монети използваме помощното множество $\{1, \ldots, x - 1\}$. Ако общата му сума е S, позицията на x се разпознава по отговора $S + x + 1$. За $x \leq 10$ този отговор е най-много 56, така че не се заменя с 0.

Не можем да продължим по същия начин за всички номинали, защото сумата на помощните монети бързо ще стане твърде голяма. Вместо това за търсения номинал x ще избираме помощно множество с обща сума точно $x-1$. Тогава без монетата x отговорът е x. Ако добавим нея, отговорът се променя. Ако добавим по-голяма монета, той остава x. Понеже по-малките номинали вече са разпознати, това е достатъчно, за да намерим позицията на x.

За построяването на помощното множество нека $t = x - 1$ и изберем максимално k, за което $\frac{k(k+1)}{2} \leq t$. Функцията `get_sum(t)` връща

$$\left\{1, 2, \ldots, k-1,\ t - \frac{k(k-1)}{2}\right\}.$$

Сумата е точно t, а последният номинал е между k и $2k$, така че няма повторения. За $k \geq 5$ имаме $2k \leq 1 + \frac{k(k-1)}{2}$ и следователно няма пропуск в постижимите суми. При $k = 4$ конструкцията работи за $t = 10, 11, 12, 13$. Единственото изключение в използвания диапазон е $t = 14$: вместо $\{1, 2, 3, 8\}$ използваме $\{1, 2, 4, 7\}$. Именно това обяснява специалния случай в авторовата функция.

Така можем да проверяваме позициите една по една. За $N = 70$ дори най-много $\frac{N(N+1)}{2}$ въпроса за едно извикване са достатъчни: за петте подтеста това са най-много 12425 въпроса, под прага за пълни точки.

Постигната сложност: $O(N^2)$ въпроса и $O(N)$ памет.
Имплементация: `sums_kiril_34p.cpp`

Намаляване на броя въпроси

Нека вече знаем всички номинали, по-малки от x, и имаме помощно множество със сума $x - 1$. Можем да добавим не една неизвестна позиция, а произволно множество от такива позиции. Ако сред тях няма монета x, първата непостижима сума остава x. Ако я има, тя става по-голяма от x. Получаването на 0 също означава, че сумата е станала по-голяма, затова правилната проверка е `query(q) != x`.

Нареждаме неизвестните позиции в списък и питаме за негов префикс, заедно с помощните монети. Проверяваното условие става вярно точно когато префиксът достигне позицията на x. Следователно можем да използваме двоично търсене. Функцията `get_pos` намира тази позиция с $O(\log N)$ въпроса. Ако остава само една неизвестна позиция, тя се връща без заявка.

Остава да уточним как се намират първите 11 монети, за които още не прилагаме `get_sum`. Авторовото решение използва последователно помощните множества $\emptyset$, $\{1\}$, $\{1, 2\}$, $\{1, 2, 3\}$ и $\{1, 2, 3, 4\}$. Първите им непостижими суми са съответно 1, 2, 4, 7 и 11. Те позволяват да разпознаем последователно групите

$$\{1\}, \qquad \{2\}, \qquad \{3, 4\}, \qquad \{5, 6, 7\}, \qquad \{8, 9, 10, 11\}.$$

Тук двоичното търсене намира първата неизвестна позиция с номинал, не по-голям от текущата първа непостижима сума M. В рамките на една група монетите може да бъдат открити в произволен ред. Затова след намирането на позицията питаме още веднъж само за нея и помощното множество. Ако отговорът е r, номиналът е $r - M$. Имаме $r \leq 22 < 70$, така че този отговор не е скрит като 0. След приключване на всяка група всички монети, нужни за следващото помощно множество, вече са известни.

След първите 11 монети намираме $12, 13, \ldots, N$ по ред чрез `get_sum` и двоично търсене. Вече знаем кой номинал търсим и допълнителният въпрос за разпознаването му не е нужен.

Постигната сложност: $O(N \log N)$ въпроса и $O(N)$ памет.
Имплементация: `sums_kiril_61p.cpp`

Трета подзадача

При $N = 1000$ можем да намалим още броя въпроси. Първо намираме монетите от 1 до 31 по вече описания начин. Общата им сума е

$$1 + 2 + \cdots + 31 = 496.$$

Към тези монети добавяме по една неизвестна позиция. Ако нейният номинал е $x \leq 497$, първата непостижима сума е $497 + x \leq 994 < 1000$. Така един въпрос едновременно ни казва, че монетата е в този диапазон, и определя точния ѝ номинал. Ако $x > 497$, отговорът остава 497 и засега оставяме позицията неизвестна.

След едно обхождане на всички останали позиции сме разпознали монетите от 1 до 497. За номиналите от 498 до 1000 отново използваме помощно множество със сума $x-1$ и двоично търсене. Така голяма част от монетите са намерени с по един въпрос вместо с логаритмичен брой.

Изборът на 31 е съществен. При монетите от 1 до 32 общата сума би била 528 и за някои добавени номинали първата непостижима сума би надхвърлила 1000. Тогава няколко различни номинала биха дали отговор 0 и нямаше да можем да ги разпознаем чрез изваждане.

Пълната авторова програма разглежда отделно $N = 3$, използва последователните двоични търсения за $N = 70$ и добавя груповото разпознаване за $N = 1000$. При всяко извикване на `solve` масивите за известните позиции се нулират.

Постигната сложност: $O(N \log N)$ въпроса и $O(N)$ памет, с по-малък брой въпроси благодарение на груповата стъпка.
Имплементация за $N = 1000$: `sums_kiril_55p.cpp`.
Обединено решение: `author.cpp`

Посочените сложности броят въпросите, а не цялото време за изпълнение. Един въпрос с K позиции струва $O(K \log K)$ време в грейдъра; в конкретната реализация се създават и вектори с префикси. Консервативна горна граница за общото време на двоичния подход е $O(N^2 \log^2 N)$.

Накрая трябва да внимаваме и с оценяването: прагът 26000 въпроса е общ за петте подтеста, а не за едно извикване на `solve`. Всички позиции в една заявка трябва да са различни. Това е изпълнено, защото помощните монети са вече разпознати, а останалата част на заявката се взема само от неизвестните позиции.

Автор: Кирил Зулямски