

Rozděl a flexi

Adam a Hugo připravují kouzelnický trik pro show *Uzbekistán má talent*. Trik spočívá v tom, že Adam vyřeší následující **problém rozdělení**: danou multimnožinu kladných celých čísel rozdělí na K neprázdných skupin tak, aby součet prvků v každé skupině byl stejný. Jinými slovy, každé celé číslo z multimnožiny musí být přiřazeno přesně jedné z K skupin a součet v každé skupině musí být stejný. Například pokud je daná multimnožina $[2, 1, 6, 4, 5]$ a $K = 3$, platné rozdělení by bylo $[2, 4]$, $[1, 5]$ a $[6]$. V tomto případě je součet prvků v každé skupině přesně 6.

Trik se provádí následovně:

- Hugo a Adam vstupují do oddělených kabin a nemohou spolu komunikovat.
- Člen poroty dá Hugovi pole A obsahující N kladných celých čísel, $A[0], A[1], \dots, A[N - 1]$, kde každá hodnota je mezi 1 a M včetně. Člen poroty dá Hugovi také hodnotu K .
- Hugo vybere maximálně $K - 1$ (ne nutně různých) celých čísel, která **přidá jako nové prvky** do pole A . Každé přidané celé číslo musí být také v rozmezí 1 až M včetně.
- Porota připojí celá čísla vybraná Hugem k původnímu poli. Toto rozšířené pole je poté **seřazeno v neklesajícím pořadí** a předáno Adamovi spolu s hodnotou K .
- Adam pak musí vyřešit problém rozdělení tohoto rozšířeného, seřazeného pole.

Vaším úkolem je vymyslet a realizovat strategii pro Adama a Huga. Lze dokázat, že za daných omezení pro ně existuje strategie, která jim umožňuje úspěšně vyřešit problém rozdělení, a to bez ohledu na pole A poskytnuté porotou.

Implementační detaily

Měli byste implementovat dvě funkce.

Funkce, kterou byste měli implementovat pro **Huga**, je následující:

```
std::vector<int> add_numbers(std::vector<int> A, int K, int M)
```

- A : pole délky N představující původní pole předané Hugovi.
- K : cílový počet skupin; poznamenejme, že Hugo může do pole přidat až $K - 1$ celých čísel.
- M : maximální povolená hodnota pro každé původní i nově přidané celé číslo.
- Tato funkce je pro každý test volána právě jednou.

Funkce by měla vrátit pole C obsahující celá čísla, která chce Hugo přidat do původního pole. Nechť S označuje délku pole C .

- S musí být nejvýše $K - 1$.
- Každý prvek pole C musí ležet v intervalu od 1 do M včetně.

Funkce, kterou byste měli implementovat pro **Adama**, je:

```
std::vector<int> find_partition(std::vector<int> B, int K)
```

- B : pole o délce $N + S$ obsahující jak původní celá čísla z A , tak čísla, která Hugo přidal. Prvky pole B jsou seřazeny v neklesajícím pořadí.
- K : požadovaný počet skupin.
- Tato funkce je v každém testu volána přesně jednou.

Funkce by měla vrátit pole P , které představuje rozdělení B do K vzájemně disjunktních skupin.

- Délka P by měla být $N + S$.
- $P[j]$ označuje skupinu, do které patří $B[j]$ (pro každé j takové, že $0 \leq j < N + S$).
- Skupiny by měly být očíslovány od 0 do $K - 1$, což znamená, že musí platit $0 \leq P[j] < K$ (pro každé $0 \leq j < N + S$).
- Pro každou skupinu i (od 0 do $K - 1$ včetně) musí existovat alespoň jeden prvek j ($0 \leq j < N + S$) takový, že $P[j] = i$.
- Součet prvků přiřazených ke každé z K skupin musí být stejný.

Při testování v CMS se program, který volá výše uvedené funkce, spustí **právě dvakrát**.

- Během prvního spuštění programu:
 - Je právě jednou volána funkce `add_numbers`.
 - Vracené pole je zpracováno CMSkem, které spočte pole B .
- Během druhého spuštění programu:
 - Je právě jednou volána funkce `find_partition`.

Omezení

- $3 \leq N \leq 100\,000$
- $2 \leq K \leq 100\,000$
- $K \leq N$
- $1 \leq M \leq 10^9$
- $1 \leq A[i] \leq M$ pro každé i takové, že $0 \leq i < N$.

Podúlohy

Podúloha	Body	Omezení
1	5	$N = 3$
2	4	$M = 1$
3	7	$M \leq 2$
4	10	$N \leq 10$
5	7	$K = 2$
6	12	$K \leq 3$
7	17	$K \leq 10$
8	20	$K \leq 100$
9	18	Žádná další omezení.

Příklad

Uvažme následující volání:

```
add_numbers([8, 2, 9, 6, 1, 5, 5], 3, 9)
```

V tomto příkladu je $A = [8, 2, 9, 6, 1, 5, 5]$. Chceme čísla rozdělit do $K = 3$ skupin. Hugo může přidávat čísla v rozmezí 1 až $M = 9$.

Funkce může vrátit pole $[5, 4]$, které znamená, že se Hugo rozhodl přidat $S = 2$ nová čísla: 5 a 4. Tato čísla jsou platná, protože obě leží v rozmezí 1 a $M = 9$.

Rozšířené pole se poté seřadí a předá Adamovi následujícím voláním:

```
find_partition([1, 2, 4, 5, 5, 5, 6, 8, 9], 3)
```

Čísla z tohoto pole může rozdělit do těchto tří skupin: $[1, 5, 9]$, $[2, 5, 8]$ a $[4, 5, 6]$. Všimněte si, že součet každé z těchto skupin se rovná 15. K odevzdání tohoto rozdělení by měla funkce vrátit pole $[0, 1, 2, 0, 1, 2, 2, 1, 0]$.

Ukázkový grader

Formát vstupu:

```
N K M  
A[0] A[1] ... A[N-1]
```

Formát výstupu:

Po dokončení volání funkce `add_numbers` ukázkový grader:

- Vypočítá seřazené pole B .
- Vypíše pole C a B v následujícím formátu, **následované prázdným řádkem**:

```
S  
C[0] C[1] ... C[S-1]  
B[0] B[1] ... B[N+S-1]
```

Po dokončení volání funkce `find_partition` vypíše ukázkový grader následující výstup:

```
L  
P[0] P[1] ... P[L-1]
```

Zde L je délka pole P , které vrací funkce `find_partition`.