

Partizionamento

Temur e il suo assistente Ulug'bek stanno preparando il loro trucco di magia per lo spettacolo televisivo *L'Uzbekistan è talentuoso*.

Il trucco consiste nella risoluzione di un **problema di partizionamento**: cioè suddividere un array in K gruppi disgiunti non vuoti tale che la somma degli elementi in ogni gruppo sia la stessa. In altre parole, ogni intero dell'array deve essere assegnato a esattamente uno di K gruppi e la somma degli elementi di ciascun gruppo deve essere la stessa. Per esempio, se l'array dato è $[2, 1, 6, 4, 5]$ e $K = 3$, i gruppi $[2, 4]$, $[1, 5]$ e $[6]$ sono una partizione valida, in cui ogni gruppo ha somma degli elementi uguale a 6.

Il trucco funziona come segue:

- Ulug'bek e Temur entrano in due cabine diverse e non possono comunicare l'uno con l'altro.
- La giuria fornisce a Ulug'bek l'intero K e un array A di N interi positivi $A[0], A[1], \dots, A[N-1]$, in cui ogni valore è compreso tra 1 e M inclusi.
- Ulug'bek sceglie al più $K - 1$ interi (non necessariamente distinti) e **li aggiunge** all'array A . Anche questi interi devono essere compresi tra 1 e M inclusi.
- La giuria **ordina in modo non decrescente** l'array così ottenuto e lo dà a Temur, assieme all'intero K .
- Temur deve quindi risolvere il problema per questo array esteso e ordinato.

Aiuta i due a far funzionare il trucco di magia! Si può dimostrare che sotto le assunzioni del problema, è sempre possibile risolvere il problema di partizionamento descritto.

Implementazione

Dovrai implementare due funzioni. La funzione che devi implementare per **Ulug'bek** è:

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

- A : l'array di N interi che la giuria fornisce a Ulug'bek.
- K : il numero di gruppi da creare. Nota che Ulug'bek può aggiungere fino a $K - 1$ interi all'array.
- M : il massimo valore consentito nell'array originale e per i valori aggiunti.
- La funzione viene chiamata esattamente una volta per caso di test.

La funzione deve restituire un array C contenente gli interi da aggiungere all'array originale. Sia S la lunghezza dell'array C , allora:

- S deve essere al più $K - 1$;
- gli elementi di S devono essere compresi tra 1 e M inclusi.

La funzione che devi implementare per **Temur** è:

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

- B : l'array di lunghezza $N + S$ contenente i numeri originariamente A e quelli aggiunti da Ulug'bek, riordinati in ordine non decrescente.
- K : il numero di gruppi da creare.
- La funzione viene chiamata esattamente una volta per caso di test.

La funzione deve restituire un array P che rappresenti la partizione di B in K gruppi:

- P deve essere lungo $N + S$;
- per ogni $0 \leq h < N + S$, $P[j]$ indica il gruppo cui appartiene $B[j]$;
- i gruppi devono essere numerati da 0 a $K - 1$, quindi $0 \leq P[j] < K$ per ogni $0 \leq j < N + S$;
- i gruppi non devono essere vuoti, cioè per ogni $0 \leq i < K$ ci deve essere almeno un $0 \leq j < N + S$ per cui $P[j] = i$.
- la somma degli elementi assegnati a ogni gruppo deve essere la stessa.

Nella valutazione effettiva, il tuo programma **verrà eseguito esattamente due volte**:

- Nella prima esecuzione `add_numbers` è chiamata esattamente una volta e il sistema di valutazione processa l'array restituito per calcolare B .
- Nella seconda esecuzione `find_partition` è chiamata esattamente una volta.

Assunzioni

- $3 \leq N \leq 100\,000$.
- $2 \leq K \leq 100\,000$.
- $K \leq N$.
- $1 \leq M \leq 10^9$.
- $1 \leq A[i] \leq M$ per ogni $0 \leq i < N$.

Subtask

Subtask	Punteggio	Assunzioni aggiuntive
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	Nessuna assunzione aggiuntiva.

Esempio

Considera la seguente chiamata:

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

Qui $A = [8, 2, 9, 6, 1, 5, 5]$, vogliamo partizionare i numeri in $K = 3$ gruppi e Ulug'bek può aggiungere numeri tra 1 e $M = 9$.

La funzione potrebbe restituire $[5, 4]$, a indicare che vanno aggiunti $S = 2$ nuovi interi: 5 e 4. Nota che questi valori sono validi siccome sono compresi tra 1 e $M = 9$.

L'array esteso è quindi ordinato e fornito a Temur dalla seguente chiamata:

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

Possiamo dividere i numeri nei gruppi: $[1, 5, 9]$, $[2, 5, 8]$, e $[4, 5, 6]$. Notiamo che la somma degli elementi di ogni gruppo è 15. Per fornire la soluzione, la funzione deve restituire l'array $[0, 1, 2, 0, 1, 2, 2, 1, 0]$.

Grader di esempio

Formato di input

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

Formato di output:

Dopo la chiamata ad `add_numbers` il grader di esempio:

- calcola l'array ordinato B ;
- stampa gli array C e B nel seguente formato, **seguiti da una linea vuota**.

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

Dopo la chiamata a `find_partition`, il grader stampa:

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

Dove L è la lunghezza dell'array P restituito da `find_partition`.