

Partition

Temur e o seu assistente, Ulug'bek, estão a preparar um truque de magia para o programa de televisão *Uzbekistan Got Talent*. O truque consiste em Temur resolver o seguinte **problema de partição**: particionar uma dada colecção de inteiros positivos em K grupos não vazios de modo a que a soma dos elementos de cada grupo seja a mesma. Por outras palavras, cada número inteiro da colecção deve ser atribuído a exactamente um dos K grupos, e a soma dentro de cada grupo deve ser igual. Por exemplo, se a colecção dada for $[2, 1, 6, 4, 5]$ e $K = 3$, uma partição válida seria $[2, 4]$, $[1, 5]$ e $[6]$. Neste caso, a soma dos elementos de cada grupo é exactamente 6.

O truque é realizado da seguinte forma:

- Ulug'bek e Temur entram em cabines separadas e não conseguem comunicar um com o outro.
- Um membro do júri entrega a Ulug'bek um array A de N inteiros positivos, $A[0], A[1], \dots, A[N - 1]$, em que cada valor está compreendido entre 1 e M inclusive. O membro do júri também entrega a Ulug'bek o valor de K .
- Ulug'bek escolhe no máximo $K - 1$ inteiros (não necessariamente distintos) para **adicionar como novos elementos** ao array A . Cada número inteiro adicionado deve estar entre 1 e M inclusive.
- O júri acrescenta os números inteiros escolhidos por Ulug'bek ao array original. Este array expandido é então **ordenado por ordem não decrescente** e entregue a Temur, juntamente com o valor de K .
- Temur deve então resolver o problema de particionamento para este array estendido e ordenado.

A tua tarefa é elaborar e implementar uma estratégia para Temur e Ulug'bek. Pode provar-se que, sob as restrições dadas, existe uma estratégia que lhes permite resolver com sucesso o problema da partição, independentemente do array A fornecido pelo júri.

Detalhes da Implementação

Deves implementar duas funções.

A função que deves implementar para o **Ulug'bek** é:

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

- A : um array de tamanho N que representa o array original fornecido a Ulug'bek.
- K : o número alvo de grupos; nota que o Ulug'bek pode adicionar até $K - 1$ números inteiros ao array.
- M : o valor máximo permitido para cada número inteiro original e recém-adicionado.
- Esta função é chamada exactamente uma vez para cada caso de teste.

A função deve devolver um array C contendo os números inteiros que Ulug'bek pretende adicionar ao array original. Seja S o tamanho do array C .

- S deve ser no máximo $K - 1$.
- Cada elemento de C deve estar compreendido entre 1 e M inclusive.

A função que deves implementar para o **Temur** é:

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

- B : um array de tamanho $N + S$ contendo tanto os inteiros originais de A como os inteiros adicionados por Ulug'bek. Os elementos do array B estão ordenados por ordem não decrescente.
- K : o número alvo de grupos.
- Esta função é chamada exactamente uma vez para cada caso de teste.

A função deve devolver um array P que representa a partição de B em K grupos disjuntos.

- O comprimento de P deve ser $N + S$.
- Para cada j tal que $0 \leq j < N + S$, $P[j]$ significa o grupo a que pertence $B[j]$.
- Os grupos devem ser numerados de 0 a $K - 1$, o que significa que $0 \leq P[j] < K$ deve ser válido para cada $0 \leq j < N + S$.
- Para cada i entre 0 e $K - 1$, inclusive, deve existir pelo menos um elemento j ($0 \leq j < N + S$) tal que $P[j] = i$.
- A soma dos elementos atribuídos a cada um dos grupos K deve ser idêntica.

Na avaliação propriamente dita, um programa que chame as funções acima é executado **exactamente duas vezes**.

- Durante a primeira execução do programa:
 - `add_numbers` é chamada exactamente uma vez.
 - O array devolvido é processado pelo sistema de avaliação para calcular o array B .
- Durante a segunda execução do programa:
 - `find_partition` é chamada exactamente uma vez.

Restrições

- $3 \leq N \leq 100\,000$
- $2 \leq K \leq 100\,000$

- $K \leq N$
- $1 \leq M \leq 10^9$
- $1 \leq A[i] \leq M$ para cada i tal que $0 \leq i < N$.

Subtarefas

Subtarefa	Pontos	Restrições Adicionais
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	Nenhuma restrição adicional.

Exemplo

Considera a seguinte chamada:

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

Neste exemplo $A = [8, 2, 9, 6, 1, 5, 5]$. Queremos dividir os números em $K = 3$ grupos. Ulug'bek consegue adicionar números entre 1 e $M = 9$.

A função pode devolver $[5, 4]$, o que significa que Ulug'bek decide adicionar $S = 2$ novos inteiros: 5 e 4. Estes são válidos, pois ambos os elementos estão compreendidos entre 1 e $M = 9$.

O array expandido é então ordenado e passado para Temur com a seguinte chamada:

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

Podemos dividir os inteiros deste array nestes três grupos: $[1, 5, 9]$, $[2, 5, 8]$, e $[4, 5, 6]$. Nota que a soma de cada um destes grupos é igual a 15. Para reportar esta partição, a função deve devolver o array $[0, 1, 2, 0, 1, 2, 2, 1, 0]$.

Avaliador de Exemplo

Formato do Input:

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

Formato do Output:

Após a chamada a `add_numbers` estar concluída, o avaliador de exemplo:

- Calcula o array ordenado B .
- Imprime os arrays C e B no seguinte formato, **seguidos de uma linha em branco**:

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

Após a chamada a `find_partition` estar concluída, o avaliador apresenta o seguinte output:

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

Aqui, L é o tamanho da array P devolvido por `find_partition`.