

Partición

Temur y su asistente, Ulug'bek, están preparando un truco de magia para el programa *Uzbekistan Got Talent*. El truco consiste en que Temur resuelva el siguiente **problema de partición**: dividir una colección dada de enteros positivos en K grupos no vacíos de tal manera que la suma de los elementos en cada grupo sea la misma.

En otras palabras, cada número entero de la colección debe asignarse a exactamente uno de los K grupos, y la suma dentro de cada grupo debe ser igual. Por ejemplo, si la colección dada es $[2, 1, 6, 4, 5]$ y $K = 3$, una partición válida sería $[2, 4]$, $[1, 5]$ y $[6]$. En este caso, la suma de los elementos en cada grupo es exactamente 6.

El truco se realiza de la siguiente manera:

- Ulug'bek y Temur entran en cabinas separadas y no pueden comunicarse entre sí.
- Un miembro del jurado le da a Ulug'bek un arreglo A de N enteros positivos, $A[0], A[1], \dots, A[N-1]$, donde cada valor está entre 1 y M inclusive. El miembro del jurado también le da a Ulug'bek el valor de K .
- Ulug'bek elige como máximo $K - 1$ enteros (no necesariamente distintos) para **agregar como nuevos elementos** al arreglo A . Cada entero agregado también debe estar entre 1 y M inclusive.
- El jurado agrega los enteros elegidos por Ulug'bek al arreglo original. Este arreglo extendido luego se **ordena en orden no decreciente** y se le entrega a Temur, junto con el valor de K .
- Temur debe entonces resolver el problema de partición para este arreglo extendido y ordenado.

Tu tarea consiste en diseñar e implementar una estrategia para Temur y Ulug'bek. Se puede demostrar que, bajo las restricciones dadas, existe una estrategia que les permite resolver con éxito el problema de partición, independientemente del arreglo A proporcionado por el jurado.

Detalles de implementación

Debes implementar dos procedimientos.

El procedimiento que debes implementar para **Ulug'bek** es:

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

- A : un arreglo de longitud N que representa el arreglo original dado a Ulug'bek.

- K : el número objetivo de grupos; tenga en cuenta que Ulug'bek puede agregar hasta $K - 1$ enteros al arreglo.
- M : el valor máximo permitido para cada entero original y recién agregado.
- Este procedimiento se llama exactamente una vez para cada caso de prueba.

El procedimiento debe devolver un arreglo C que contenga los números enteros que Ulug'bek quiere agregar al arreglo original. Sea S la longitud del arreglo C .

- S debe ser como máximo $K - 1$.
- Cada elemento de C debe estar entre 1 y M inclusive.

El procedimiento que debe implementar para **Temur** es:

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

- B : un arreglo de longitud $N + S$ que contiene tanto los enteros originales de A como los enteros que Ulug'bek agregó. Los elementos del arreglo B están ordenados en orden no decreciente.
- K : el número objetivo de grupos.
- Este procedimiento se llama exactamente una vez para cada caso de prueba.

El procedimiento debe devolver un arreglo P que represente la partición de B en K grupos disjuntos.

- La longitud de P debe ser $N + S$.
- Para cada j tal que $0 \leq j < N + S$, $P[j]$ significa el grupo al que pertenece $B[j]$.
- Los grupos deben numerarse del 0 al $K - 1$, lo que significa que $0 \leq P[j] < K$ debe cumplirse para cada $0 \leq j < N + S$.
- Para cada i entre 0 y $K - 1$ inclusive, debe haber al menos un elemento j ($0 \leq j < N + S$) tal que $P[j] = i$.
- La suma de los elementos asignados a cada uno de los grupos K debe ser idéntica.

En la ejecución propiamente dicha, un programa que llama a los procedimientos anteriores se ejecuta **exactamente dos veces**.

- Durante la primera ejecución del programa:
 - `add_numbers` se llama exactamente una vez.
 - El array devuelto es procesado por el sistema de calificación para calcular el array B .
- Durante la segunda ejecución del programa:
 - `find_partition` se llama exactamente una vez.

Restricciones

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

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

Subtareas

Subtarea	Puntuación	Restricciones adicionales
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	Sin restricciones adicionales.

Ejemplo

Considere la siguiente llamada:

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

En este ejemplo $A = [8, 2, 9, 6, 1, 5, 5]$. Queremos dividir los números en $K = 3$ grupos. Ulug'bek puede sumar números entre 1 y $M = 9$.

El procedimiento puede devolver $[5, 4]$, lo que significa que Ulug'bek decide agregar $S = 2$ nuevos enteros: 5 y 4. Estas son válidas ya que ambos elementos están entre 1 y $M = 9$.

A continuación, el array extendido se ordena y se pasa a Temur con la siguiente llamada:

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

Podríamos dividir los enteros de este arreglo en estos tres grupos: $[1, 5, 9]$, $[2, 5, 8]$, y $[4, 5, 6]$. Tenga en cuenta que la suma de cada uno de estos grupos es igual a 15. Para informar sobre esta partición, el procedimiento debe devolver el arreglo $[0, 1, 2, 0, 1, 2, 2, 1, 0]$.

Grader de ejemplo

Formato de entrada:

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

Formato de salida:

Una vez que finaliza la llamada a `add_numbers`, el grader de ejemplo:

- Calcula el arreglo ordenado B .
- Imprime los arreglos C y B en el siguiente formato, **seguido de una línea vacía**:

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

Una vez que finaliza la llamada a `find_partition`, el evaluador muestra lo siguiente:

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

Aquí, L es la longitud del arreglo P devuelto por `find_partition`.