

分割 (Partition)

Temur と彼の助手 Ulug'bek は、番組 *Uzbekistan Got Talent* に向けて手品を準備している。この手品は、Temur が次の**分割問題**を解くことを中心に展開される：与えられた正の整数の列を K 個の空でないグループに分割し、各グループの要素の和がすべて等しくなるようにする。すなわち、数列に含まれるそれぞれの整数は K 個のグループのうちちょうど 1 つに割り当てられる必要があり、また、各グループ内の要素の和は等しくなければならない。例えば、与えられた数列が $[2, 1, 6, 4, 5]$ であり $K = 3$ である場合、有効な分割としては $[2, 4]$, $[1, 5]$, $[6]$ が挙げられる。このとき各グループの要素の和はちょうど 6 になる。

この手品は次のように行われる：

- Ulug'bek と Temur は別々のブースに入り、互いに連絡を取ることはできない。
- 審査員が Ulug'bek に N 個の正整数 $A[0], A[1], \dots, A[N-1]$ からなる配列 A と値 K を与える。 A のそれぞれの値は 1 以上 M 以下である。
- Ulug'bek は配列 A に **新しい要素として追加** する、最大で $K - 1$ 個の整数（必ずしも互いに異なる必要はない）を選ぶ。追加される整数も 1 以上 M 以下でなければならない。
- 審査員は Ulug'bek が選んだ整数を元の配列の末尾に追加する。この拡張された配列は、その後**昇順にソート**され、 K の値とともに Temur に渡される。
- Temur はこの拡張されソートされた配列に対して分割問題を解かなければならない。

あなたの課題は、Temur と Ulug'bek のための戦略を考案し、実装することである。ここで、与えられた制約の下で、審査員によって提供される配列 A によらず、分割問題を正しく解くことのできるような戦略が必ず存在することが証明できる。

実装の詳細

あなたは 2 種類の関数を実装する必要がある。

Ulug'bek のために実装すべき関数は以下の通りである：

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

- A : Ulug'bek に渡される元の配列を表す長さ N の配列。
- K : 目標とするグループ数。また、Ulug'bek は配列に $K - 1$ 個まで値を追加できる。
- M : 元の配列の整数および新たに追加される整数に対して許容される最大値。
- この関数は各テストケースに対してちょうど 1 回呼び出される。

この関数は Ulug'bek が元の配列に追加したい整数を格納した配列 C を返す必要がある。 S を配列 C の長さとする。

- S は $K - 1$ 以下でなければならない。
- C のそれぞれの要素は 1 以上 M 以下でなければならない。

Temur のために実装すべき手続きは以下の通りである：

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

- B : 長さ $N + S$ の配列で、 A からの元の整数と Ulug'bek が追加した整数の両方を含む。配列 B の要素は昇順にソートされている。
- K : 目標となるグループ数。
- この関数は各テストケースに対してちょうど 1 回呼び出される。

この関数は B を K 個のグループに分割した結果を表す配列 P を返す必要がある。

- P の長さは $N + S$ である必要がある。
- $0 \leq j < N + S$ を満たす各 j について、 $P[j]$ は $B[j]$ が属するグループを示す。
- グループには 0 から $K - 1$ までの番号が付けられる必要がある。すなわち、各 $0 \leq j < N + S$ について $0 \leq P[j] < K$ が成立しなければならない。
- 0 以上 $K - 1$ 以下の各 i について、 $P[j] = i$ を満たす j ($0 \leq j < N + S$) が少なくとも 1 つ存在しなければならない。
- K 個のグループそれぞれに割り当てられた要素の合計はすべて等しい必要がある。

実際の採点では、上記の手続きを呼び出すプログラムがちょうど 2 回実行される。

- プログラムの最初の実行時：
 - `add_numbers` がちょうど 1 回呼び出される。
 - 返された配列が採点システムによって処理され、配列 B が計算される。
- プログラムの 2 回目 の実行時：
 - `find_partition` がちょうど 1 回呼び出される。

制約

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

小課題

小課題	得点	追加の制約
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	追加の制約はない.

例

次の呼び出しを考える：

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

この例では、 $A = [8, 2, 9, 6, 1, 5, 5]$ である。これらの数を $K = 3$ つのグループに分割したい。Ulug'bek は、1 以上 $M = 9$ 以下の数を追加できる。

この関数は $[5, 4]$ を返すことができる。これは、Ulug'bek が $S = 2$ つの新しい整数を追加することを意味する。すなわち、5 および 4 を追加する。これらの要素はいずれも 1 以上 $M = 9$ 以下であるため、有効である。

その後、拡張された配列はソートされ、次の呼び出しで Temur に渡される：

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

この配列の整数を $[1, 5, 9]$ 、 $[2, 5, 8]$ 、および $[4, 5, 6]$ の 3 つのグループに分割できる。これらの各グループの和はすべて 15 となる。この分割結果を報告するために、この関数は配列 $[0, 1, 2, 0, 1, 2, 2, 1, 0]$ を返す必要がある。

採点プログラムのサンプル

入力形式：

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

出力形式：

`add_numbers` の呼び出しが完了した後、採点プログラムのサンプルは以下の処理を行う：

- ソート済みの配列 B を計算する。
- 配列 C および B を以下の形式で出力し、**その後に空行を 1 行出力する**：

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

`find_partition` の呼び出しが完了した後、採点プログラムのサンプルは以下を出力する：

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

ここで、 L は `find_partition` によって返される配列 P の長さである。