

Felbontás (Partition)

Temur és asszisztense, Ulug'bek egy bűvésztűrköt készítenek elő az *Uzbekistan Got Talent* műsorhoz. A trükk lényege, hogy Temur megoldja a következő **partíciós problémát**: egy adott pozitív egész számokból álló halmazt particionálunk K nem üres csoportra úgy, hogy az egyes csoportok elemeinek összege azonos legyen. Más szóval, a halmaz minden elemét a K csoport egyikéhez kell rendelni, és az egyes csoportokon belüli összegnek egyenlőnek kell lennie. Például, ha az adott gyűjtemény $[2, 1, 6, 4, 5]$ és $K = 3$, akkor egy érvényes partíció $[2, 4]$, $[1, 5]$ és $[6]$ lenne. Ebben az esetben az egyes csoportok elemeinek összege 6.

A trükköt a következőképpen hajtják végre:

- Ulug'bek és Temur külön fürkékbe lépnek, és nem tudnak kommunikálni egymással.
- Egy zsűritag megadja Ulug'beknek az A tömböt, amely N pozitív egész számból áll: $A[0], A[1], \dots, A[N-1]$, ahol minden érték 1 és M között van. A zsűritag megadja Ulug'beknek a K értéket is.
- Ulug'bek választ legfeljebb $K - 1$ (nem feltétlenül különböző) egész számot, hogy **új elemként** hozzáadja őket az A tömbhöz. Minden hozzáadott egész számnak 1 és M között kell lennie, az intervallum két szélét is beleértve.
- A zsűri hozzáfűzi az Ulug'bek által kiválasztott egész számokat az eredeti tömbhöz. Ezt a kibővített tömböt ezután **nemcsökkenő sorrendbe rendezi**, és Temurnak adja a K értékkel együtt.
- Temurnak ezután meg kell oldania a particionálási problémát erre a kibővített, rendezett tömbre.

A feladatod, hogy kidolgozz és megvalósíts egy stratégiát Temur és Ulug'bek számára. Bizonyítható, hogy az adott feltételek mellett létezik számukra olyan stratégia, amely lehetővé teszi a particionálási probléma sikeres megoldását, függetlenül a zsűri által megadott A tömbtől.

Megvalósítás

Két függvényt kell implementálnod.

Az **Ulug'bek** esetében a következő függvényt kell implementálnod:

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

- A : egy N hosszúságú tömb, amely az Ulug'beknek adott eredeti tömböt tartalmazza.

- K : a később létrehozandó csoportok száma; vegyük észre, hogy Ulug'bek akár $K - 1$ egész számot is hozzáadhat a tömbhöz.
- M : az eredeti és újonnan hozzáadott egész számok megengedett maximális értéke.
- Ez a függvény minden tesztesetben pontosan egyszer kerül meghívásra.

A függvénynek egy C tömböt kell visszaadnia, amely azokat az egész számokat tartalmazza, amelyeket Ulug'bek hozzá akar adni az eredeti tömbhöz. Jelölje S a C tömb hosszát.

- S legfeljebb $K - 1$ lehet.
- C minden elemének 1 és M között kell lennie, az intervallum végpontjait is beleértve.

Temur esetében a következő függvényt kell implementálnod:

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

- B : egy $N + S$ hosszúságú tömb, amely tartalmazza mind az A tömbből származó eredeti egész számokat, mind az Ulug'bek által hozzáadott egész számokat. B tömb elemei nemcsökkenő sorrendbe vannak rendezve.
- K : a létrehozandó csoportok száma.
- Ez a függvény minden tesztesetben pontosan egyszer kerül meghívásra.

Az eljárásnak egy P tömböt kell visszaadnia, amely a B halmaz K diszjunkt csoportra való felosztását tartalmazza.

- P hosszának $N + S$ -nek kell lennie.
- Minden olyan j esetén, amelyre $0 \leq j < N + S$, $P[j]$ jelöli azt a csoportot, amelyhez $B[j]$ tartozik.
- A csoportokat 0-tól $K - 1$ -ig kell számozni, azaz minden $0 \leq j < N + S$ esetén $0 \leq P[j] < K$.
- Minden i értékhez 0 és $K - 1$ között kell lennie legalább egy olyan j elemnek ($0 \leq j < N + S$), amelyre $P[j] = i$.
- A K csoporthoz rendelt elemek összegének azonosnak kell lennie.

A tényleges értékelés során a fenti függvényeket meghívó program **pontosan kétszer** fut le.

- A program első futtatásakor:
 - `add_numbers` függvényt pontosan egyszer hívjuk meg.
 - A visszaadott tömböt az értékelő rendszer feldolgozza a B tömb kiszámításához.
- A program második futtatása során:
 - `find_partition` függvényt pontosan egyszer hívjuk meg.

Korlátok

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

- $K \leq N$
- $1 \leq M \leq 10^9$
- $1 \leq A[i] \leq M$ minden olyan i esetén, amelyre $0 \leq i < N$.

Részfeladatok

Részfeladat	Pontszám	További megkötések
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	Nincsenek további megkötések.

Példa

Tekintsük a következő hívást:

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

Ebben a példában $A = [8, 2, 9, 6, 1, 5, 5]$. A számokat $K = 3$ csoportra szeretnénk osztani. Ulug'bek 1 és $M = 9$ közötti számokat tud hozzáadni.

A függvény visszaadhatja az $[5, 4]$ értéket, ami azt jelenti, hogy Ulug'bek úgy dönt, hogy $S = 2$ új egész számot ad hozzá, melyek: 5 és 4. Ez helyes, mivel mindkét elem 1 és $M = 9$ között van.

A kibővített tömböt ezután rendezzük és a következő hívással átadjuk Temurnak:

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

A tömb egész számait a következő három csoportra oszthatjuk: $[1, 5, 9]$, $[2, 5, 8]$ és $[4, 5, 6]$. Ezen csoportok összege 15. A partíció leírásához a függvénynek a $[0, 1, 2, 0, 1, 2, 2, 1, 0]$ tömböt kell visszaadnia.

Mintaértékelő

Bemeneti formátum:

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

Kimeneti formátum:

Az `add_numbers` függvény hívása után a mintaértékelő:

- Kiszámítja a B rendezett tömböt.
- Kiírja C és B tömböket a következő formátumban, **utána egy üres sorral**:

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

A `find_partition` függvény hívása után az értékelő a következő kimenetet adja:

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

Itt L a `find_partition` által visszaadott P tömb hossza.