

Partition

Temur și asistentul său, Ulug'bek, pregătesc un număr de magie pentru emisiunea *Uzbekistan Got Talent*. Trucul se bazează pe rezolvarea de către Temur a următoarei **probleme de partiționare**: partiționarea unei colecții date de numere întregi pozitive în K grupuri nenule, astfel încât suma elementelor din fiecare grup să fie aceeași. Cu alte cuvinte, fiecare număr întreg din mulțime trebuie atribuit exact unuia dintre cele K grupuri, iar suma din fiecare grup trebuie să fie egală. De exemplu, dacă mulțimea dată este $[2, 1, 6, 4, 5]$ și $K = 3$, o partiție validă ar fi $[2, 4]$, $[1, 5]$ și $[6]$. În acest caz, suma elementelor din fiecare grup este exact 6.

Trucul se realizează după cum urmează:

- Ulug'bek și Temur intră în cabine separate și nu pot comunica între ei.
- Un membru al juriului îi dă lui Ulug'bek un vector A format din N numere întregi, $A[0], A[1], \dots, A[N-1]$, unde fiecare valoare se află între 1 și M inclusiv, împreună cu valoarea lui K .
- Ulug'bek alege cel mult $K - 1$ (nu neapărat distincte) numere întregi pentru a le **adăuga ca elemente noi** la vectorul A . Fiecare număr întreg adăugat trebuie, de asemenea, să se încadreze între 1 și M , inclusiv.
- Juriul adaugă numerele întregi alese de Ulug'bek la vectorul original. Acest vector extins este apoi **sortat în ordine nedescrescătoare** și este transmis lui Temur, împreună cu valoarea lui K .
- Temur trebuie apoi să rezolve problema de partiționare pentru acest vector extins și sortat.

Sarcina ta este să concepi și să implementezi o strategie pentru Temur și Ulug'bek. Se poate demonstra că, în condițiile date, există o strategie care le permite să rezolve cu succes problema partiționării, indiferent de vectorul A furnizat de juriu.

Detalii de implementare

Trebuie să implementați două funcții.

Funcția pe care trebuie să o implementați pentru **Ulug'bek** este:

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

- A : un tablou de lungime N care reprezintă tabloul inițial dat lui Temur.
- K : numărul țintă de grupuri.

- M : valoarea maximă permisă pentru fiecare număr întreg inițial și nou adăugat.
- Această funcție este apelată exact o singură dată pentru fiecare caz de test.

Funcția trebuie să returneze un vector C care conține numerele întregi pe care Ulug'bek dorește să le adauge la vectorul inițial. Fie S lungimea vectorului C .

- S trebuie să fie cel mult $K - 1$.
- Fiecare element al lui C trebuie să se încadreze între 1 și M , inclusiv.

Funcția pe care trebuie să o implementați pentru **Temur** este:

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

- B : un vector de lungime $N + S$ care conține atât numerele întregi originale din A , cât și numerele întregi adăugate de Ulug'bek. Elementele vectorului B sunt sortate în ordine nedescrescătoare.
- K : numărul țintă de grupuri.
- Această funcție este apelată exact o singură dată pentru fiecare caz de test.

Funcția trebuie să returneze un vector P care reprezintă partiționarea lui B în K grupuri disjuncte.

- Lungimea lui P trebuie să fie $N + S$.
- Pentru fiecare j astfel încât $0 \leq j < N + S$, $P[j]$ indică grupul căruia îi aparține $B[j]$.
- Grupurile trebuie numerotate de la 0 la $K - 1$, ceea ce înseamnă că $0 \leq P[j] < K$ trebuie să fie valabil pentru fiecare $0 \leq j < N + S$.
- Pentru fiecare i cuprins între 0 și $K - 1$ inclusiv, trebuie să existe cel puțin un element j ($0 \leq j < N + S$) astfel încât $P[j] = i$.
- Suma elementelor atribuite fiecăruia dintre grupurile K trebuie să fie identică.

În evaluarea propriu-zisă, un program care apelează funcțiile de mai sus este rulat **exact de două ori**.

- În timpul primei rulări a programului:
 - Se apelează `add_number` exact o dată.
 - Vectorul returnat este procesat de grader pentru a calcula vectorul B .
- În timpul celei de-a doua rulări a programului:
 - Se apelează `find_partition` exact o dată.

Restricții

- $3 \leq N \leq 100\,000$
- $2 \leq K \leq 100\,000$
- $K \leq N$
- $1 \leq M \leq 10^9$
- $1 \leq A[i] \leq M$ pentru fiecare i astfel încât $0 \leq i < N$.

Subtask-uri

Subtask	Punctaj	Restricții suplimentare
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	Fără restricții suplimentare.

Exemplu

Considerăm următorul apel:

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

În acest exemplu, $A = [8, 2, 9, 6, 1, 5, 5]$. Vrem să împărțim numerele în $K = 3$ grupuri. Ulug'bek poate adăuga numere între 1 și $M = 9$.

Funcția poate returna $[5, 4]$, ceea ce înseamnă că Ulug'bek decide să adauge $S = 2$ numere întregi noi: 5 și 4. Acestea sunt valide, deoarece ambele elemente se află între 1 și $M = 9$.

Vectorul extins este apoi sortat și transmis lui Temur prin următorul apel:

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

Am putea împărți numerele întregi din acest vector în următoarele trei grupuri: $[1, 5, 9]$, $[2, 5, 8]$ și $[4, 5, 6]$. Observați că suma fiecăruia dintre aceste grupuri este egală cu 15. Pentru a raporta această partiție, funcția trebuie să returneze vectorul $[0, 1, 2, 0, 1, 2, 2, 1, 0]$.

Grader local

Format de intrare:

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

Format de ieșire:

După finalizarea apelului către `add_numbers`, graderul local:

- Calculează vectorul sortat B .
- Afișează vectorii C și B în următorul format, **urmat de o linie goală**:

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

După finalizarea apelului către `find_partition`, graderul local afișează:

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

Aici, L este lungimea vectorului P returnat de `find_partition`.