

Indeling

Temur og hans assistent, Ulug'bek, er i gang med at forberede et tryllesummer for *Uzbekistan Har Talent*. Tryllesummeret går ud på at Temur skal løse det følgende **inddelingsproblem**. Givet en samling af heltal ønskes det at dele samlingen op i K disjunkte grupper så summen af tallene i hver gruppe er ens. For eksempel, givet samlingen $[2, 1, 6, 4, 5]$ og $K = 3$, så er en gyldig inddeling $[2, 4]$, $[1, 5]$ og $[6]$. I dette tilfælde er summen af tallene i hver gruppe præcis 6.

Tryllesummeret bliver udført på følgende vis:

- Ulug'bek og Temur går ind i forskellige rum så de ikke kan kommunikere med hinanden.
- Et medlem af jurien giver Ulug'bek en liste A med N positive heltal, $A[0], A[1], \dots, A[N-1]$, hvor hvert tal har en værdi mellem 1 og M , inklusivt. Jurien giver også Ulug'bek værdien K .
- Ulug'bek vælger op til $K - 1$ (ikke nødvendigvis forskellige) tal og **tilføjer dem som nye tal** til listen A . Hvert nyt tal skal være tal mellem 1 og M , inklusivt.
- Jurien tilføjer tallene valgt af Ulug'bek til den oprindelige liste. Denne liste er derefter **sorteret i svagt stigende rækkefølge** og bliver derefter givet til Temur sammen med værdien K .
- Temur skal derefter løse inddelingsproblemet for denne udvidet og derefter sorteret liste.

Din opgave er at udtænke og implementere en strategi for både Temur og Ulug'bek.

Implementerings detaljer

Du skal implementere to funktioner.

Funktionen der fortæller hvad **Ulug'bek** skal gøre er:

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

- A : en liste af længde N der udgør den originale liste givet til Ulug'bek.
- K : antallet af grupper du skal danne. Du skal være opmærksom på at Ulug'bek må tilføje op til $K - 1$ tal til listen.
- M : den maksimale værdi tilladt for hvert af de oprindelige og nye tal.
- Denne funktion bliver kaldt præcis en gang for hver test case.

Denne funktion skal returnere en liste C der indeholder de tal Ulug'bek ønsker at tilføje til den oprindelige list. Lad S være længden af listen C .

- S må maksimalt være $K - 1$
- Hvert tal i C skal være mellem 1 og M inklusivt.

Funktionen der fortæller hvad **Temur** skal gøre er:

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

- B : en liste af længde $N + S$ der indeholder både de oprindelige tal fra A og tallene som Ulug'bek tilføjede. Tallene i liste B er sorteret i svagt stigende rækkefølge.
- K : antallet af grupper du skal danne.
- Denne funktion bliver kaldt præcis én gang per test case.

Funktionen skal returnere en liste P der udgør en inddeling af B i K disjunkte grupper.

- Længden af P skal være $N + S$.
- For hvert j hvor $0 \leq j < N + S$, så fortæller $P[j]$ hvilken gruppe $B[j]$ tilhører.
- Grupperne skal nummereres fra 0 til $K - 1$, det vil sige at $0 \leq P[j] < K$ skal holde for alle $0 \leq j < N + S$.
- For hvert i mellem 0 og $K - 1$ inklusivt, skal der være mindst et tal j ($0 \leq j < N + S$) sådan at $P[j] = i$.
- Summen af tallene, der er tildelt til hver af grupperne skal være identiske.

I gradingen bliver et program, der kalder de ovenstående funktioner, kørt **præcis to gange**

- I første gennemkørsel af programmet:
 - `add_numbers` bliver kaldt præcis én gang.
 - Den returnerede liste bliver gennemarbejdet af grading systemet til at beregne listen B .
- I anden gennemkørsel af programmet:
 - `find_partition` bliver kaldt præcis én gang.

Begrænsninger

- $3 \leq N \leq 100\,000$
- $2 \leq K \leq 100\,000$
- $K \leq N$
- $1 \leq M \leq 10^9$
- $1 \leq A[i] \leq M$ for alle i hvor $0 \leq i < N$.

Delopgaver

Delopgave	Point	Yderlige begrænsninger
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	Ingen yderligere begrænsninger.

Eksempel

Betragt følgende kald:

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

I dette eksempel har vi $A = [8, 2, 9, 6, 1, 5, 5]$. Vi ønsker at inddele tallene i $K = 3$ grupper. Ulug'bek kan tilføje tal mellem 1 og $M = 9$.

Funktionen kan returnere $[5, 4]$, hvilket betyder at Ulug'bek beslutter sig for at tilføje $S = 2$ nye tal: 5 og 4. Disse er begge gyldige siden at begge tal er mellem 1 og $M = 9$.

Den udvidede liste efter at den er sorteret gives til Temur med det følgende kald:

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

Vi kan vælge at opdele tallene fra denne liste ind i disse tre grupper $[1, 5, 9]$, $[2, 5, 8]$ og $[4, 5, 6]$. Vær opmærksom på at summen i hver af disse grupper af tal er lig med 15. For at rapportere denne inddeling skal funktionen returnerer listen $[0, 1, 2, 0, 1, 2, 2, 1, 0]$.

Eksempel Grader

Input format:

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

Output format:

Efter kaldene til `add_numbers` er færdige gør eksempel graderen følgende:

- Beregner den sorterede liste B .
- Skriver listen C og B ud i det følgende format, **efterfulgt af en tom linje**:

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

Efter kaldene til `add_numbers` er færdige giver graderen følgende output:

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

Her er L længden af listen P returneret af `find_partition`.