

Partition

Temur en zijn assistent, Ulug'bek, bereiden een goocheltruc voor om deel te nemen aan de show *Oezbekistan Got Talent*. De truc gaat er om dat Temur het volgende **partitieprobleem** oplost: partitioneer een gegeven lijst van positieve getallen in K niet-lege groepen zodat de som van de elementen in elke groep gelijk is. Met andere woorden, elk getal van de lijst moet toegewezen worden aan precies één van de K groepen, en de som binnen elke groep moet gelijk zijn. Bijvoorbeeld, als de gegeven lijst $[2, 1, 6, 4, 5]$ is, en $K = 3$, dan zouden $[2, 4]$, $[1, 5]$ en $[6]$ een geldige partitie vormen. Hier is de som van de elementen in elke groep exact 6.

De truc wordt als volgt uitgevoerd:

- Ulug'bek en Temur gaan aparte hokjes in en kunnen niet met elkaar communiceren.
- Een jurylid geeft Ulug'bek een lijst A met N positieve getallen, $A[0], A[1], \dots, A[N-1]$, waar elke waarde tussen 1 en M ligt (inclusief). Het jurylid geeft Ulug'bek ook de waarde K .
- Ulug'bek kiest hoogstens $K - 1$ (niet noodzakelijk verschillende) getallen om **als nieuwe elementen toe te voegen** aan de lijst A . Elke toegevoegd getal moet ook weer tussen 1 en M liggen (inclusief)
- De jury voegt de door Ulug'bek gekozen getallen toe aan de originele lijst. Deze uitgebreide lijst wordt dan **in niet-dalende volgorde gesorteerd** en aan Temur gegeven, samen met de waarde K .
- Temur moet vervolgens het partitieprobleem oplossen voor deze uitgebreide, gesorteerde lijst.

Jouw taak is om een strategie voor Temur en Ulug'bek te ontwerpen en te implementeren. Men kan bewijzen dat, onder de gegeven beperkingen, er een strategie bestaat die hen toelaat het partitieprobleem op te lossen, onafhankelijk van de lijst A die door de jury gegeven wordt.

Implementatiedetails

Je moet twee procedures implementeren.

De procedure die je voor **Ulug'bek** moet implementeren is:

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

- A : Een lijst van lengte N die de originele lijst voorstelt die aan Ulug'bek wordt gegeven.

- K : Het aantal groepen; merk op dat Ulug'bek maximaal $K - 1$ getallen mag toevoegen aan de lijst.
- M : De maximale waarde voor elk origineel en nieuw toegevoegd getal.
- Deze procedure wordt precies één keer aangeroepen voor elk testgeval.

De procedure moet een lijst C teruggeven die de getallen bevat die Ulug'bek wil toevoegen aan de originele lijst. Laat S de lengte van de lijst C voorstellen.

- S mag maximaal $K - 1$ zijn.
- Elk element van C moet inclusief tussen 1 en M liggen.

De procedure die je moet implementeren voor **Temur** is:

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

- B : Een lijst van lengte $N + S$ die zowel de originele getallen van A als de extra getallen die Ulug'bek toevoegde bevat. De elementen van de lijst B zijn niet-dalend gesorteerd.
- K : Het aantal groepen.
- Deze procedure wordt precies één keer aangeroepen voor elk testgeval.

De procedure moet een lijst P teruggeven die de partitie van B in K disjuncte groepen voorstelt.

- De lengte van P moet $N + S$ zijn.
- Voor elke j zodat $0 \leq j < N + S$, geeft $P[j]$ aan bij welke groep $B[j]$ hoort.
- De groepen moeten genummerd worden van 0 tot en met $K - 1$, dus $0 \leq P[j] < K$ moet gelden voor alle $0 \leq j < N + S$.
- Voor elke i inclusief tussen 0 en $K - 1$ moet er minstens één element j ($0 \leq j < N + S$) zijn zodat $P[j] = i$.
- De som van de elementen toegekend aan elk van de K groepen moet identiek zijn.

In de werkelijke grader wordt een programma dat de bovenstaande procedures aanroept **exact twee keer** uitgevoerd.

- In de eerste uitvoering van het programma:
 - Wordt `add_numbers` exact eenmaal aangeroepen.
 - Wordt de teruggegeven array gebruikt door het grading systeem om de lijst B te berekenen.
- In de tweede uitvoering van het programma:
 - Wordt `find_partition` exact eenmaal aangeroepen.

Beperkingen

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

- $1 \leq M \leq 10^9$
- $1 \leq A[i] \leq M$ voor alle i zodat $0 \leq i < N$.

Subtaken

Subtaak	Score	Extra beperkingen
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	Geen extra beperkingen.

Voorbeeld

Beschouw de volgende aanroep:

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

In dit voorbeeld is $A = [8, 2, 9, 6, 1, 5, 5]$. We willen de getallen partitioneren in $K = 3$ groepen. Ulug'bek kan getallen toevoegen tussen 1 en $M = 9$.

De procedure kan $[5, 4]$ teruggeven, wat wil zeggen dat Ulug'bek besluit om $S = 2$ nieuwe getallen toe te voegen: 5 en 4. Deze getallen zijn geldig, gezien ze beide tussen 1 en $M = 9$ liggen.

De uitgebreide lijst wordt dan gesorteerd en aan Temur doorgegeven met de volgende aanroep:

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

We zouden de getallen van deze lijst kunnen verdelen over deze drie groepen: $[1, 5, 9]$, $[2, 5, 8]$, en $[4, 5, 6]$. Merk op dat de som van elk van deze groepen gelijk is aan 15. Om deze partitie te rapporteren moet de procedure de lijst $[0, 1, 2, 0, 1, 2, 2, 1, 0]$ teruggeven.

Voorbeeldgrader

Invoerformaat:

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

Uitvoerformaat:

Na de aanroep naar `add_numbers` zal de voorbeeldgrader:

- De gesorteerde lijst B berekenen.
- De lijsten C en B in het volgende formaat uitschrijven, **gevolgd door een lege lijn**:

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

Aan het eind van de aanroep naar `find_partition`, schrijft de grader:

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

hier is L de lengte van de lijst P teruggegeven door `find_partition`.