

Partition

Temur och hans assistent, Ulug'bek, är på Lolos födelsedag och ska uppträda ett magiskt trick. Tricket går ut på att Temur ska lösa följande **partitionsproblem**: dela upp en given mängd positiva heltal i K icke-tomma grupper så att summan av elementen i varje grupp är densamma. Med andra ord måste varje heltal från mängden tilldelas exakt en av grupperna K , och summan inom varje grupp måste vara lika stor. Om den givna mängden till exempel är $[2, 1, 6, 4, 5]$ och $K = 3$, skulle en giltig uppdelning vara $[2, 4]$, $[1, 5]$ och $[6]$. I detta fall är summan av elementen i varje grupp exakt 6.

Tricket utförs på följande sätt:

- Ulug'bek och Temur går in i separata bås och kan inte kommunicera med varandra.
- Lolo ger Ulug'bek en vektor A bestående av N positiva heltal, $A[0]$, $A[1]$, ..., $A[N - 1]$, där varje värde ligger mellan 1 och M , båda värdena inkluderade. Lolo ger också Ulug'bek värdet på K .
- Ulug'bek väljer högst $K - 1$ (inte nödvändigtvis olika) heltal som ska **läggas till som nya element** i arrayen A . Varje tillagt heltal måste också ligga mellan 1 och M , båda värdena inkluderade.
- Lolo lägger till de heltal som Ulug'bek valt till den ursprungliga arrayen. Denna utökade array **sorteras sedan i icke-minskande ordning** och överlämnas till Temur, tillsammans med värdet på K .
- Temur måste sedan lösa partitionsproblemet för denna utökade, sorterade array.

Lolo älskar magiska trick som handlar om matematik, och önskar att få se hur Temur och Ulug'bek. Din uppgift är att uppfylla Lolos önskan, och utforma samt implementera en strategi för Temur och Ulug'bek. Det kan bevisas att under de givna villkoren, finns det en strategi som gör det möjligt för dem att lösa partitionsproblemet, oavsett vilken lista A som Lolo tillhandahåller.

Implementationsdetaljer

Du skall implementera två funktioner

Den funktion du ska implementera för **Ulug'bek** är:

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

- A : en array med längden N som representerar den ursprungliga arrayen som Ulug'bek får.

- K : det önskade antalet grupper; notera att Ulug'bek kan lägga till upp till $K - 1$ heltal till arrayen.
- M : det högsta tillåtna värdet för varje ursprungligt och nyligen tillagt heltal.
- Denna procedur anropas exakt en gång för varje testfall.

Funktionen ska returnera en array C som innehåller de heltal som Ulug'bek vill lägga till i den ursprungliga arrayen. Låt S beteckna längden på arrayen C .

- S får vara högst $K - 1$.
- Varje element i C måste ligga mellan 1 och M , inklusivt.

Funktionen som du ska implementera för **Temur** är:

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

- B : en array med längden $N + S$ som innehåller både de ursprungliga heltal från A och de heltal som Ulug'bek lade till. Elementen i arrayen B är sorterade i icke-minskande ordning.
- K : det önskade antalet grupper.
- Denna funktion anropas exakt en gång för varje testfall.

Funktionen ska returnera en array P som representerar uppdelningen av B i K disjunkta grupper.

- Längden på P ska vara $N + S$.
- För varje j så att $0 \leq j < N + S$, anger $P[j]$ den grupp som $B[j]$ tillhör.
- Grupperna ska numreras från 0 till $K - 1$, vilket innebär att $0 \leq P[j] < K$ måste gälla för varje $0 \leq j < N + S$.
- För varje i mellan 0 och $K - 1$ (båda inkluderade) måste det finnas minst ett element j ($0 \leq j < N + S$) sådant att $P[j] = i$.
- Summan av elementen som tilldelats var och en av grupperna K måste vara identisk.

Vid den faktiska bedömningen körs ett program som anropar ovanstående procedurer **exakt två gånger**.

- Under programmets första körning:
 - `add_numbers` anropas exakt en gång.
 - Den returnerade arrayen bearbetas av gradern för att beräkna arrayen B .
- Under programmets andra körning:
 - `find_partition` anropas exakt en gång.

Begränsningar

- $3 \leq N \leq 100\,000$
- $2 \leq K \leq 100\,000$
- $K \leq N$
- $1 \leq M \leq 10^9$
- $1 \leq A[i] \leq M$ för varje i så att $0 \leq i < N$.

Deluppgifter

Deluppgift	Poäng	Ytterligare begränsningar
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	Inga ytterligare begränsningar.

Exempel

Betrakta följande anrop:

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

I det här exemplet är $A = [8, 2, 9, 6, 1, 5, 5]$. Vi vill dela upp talen i $K = 3$ grupper. Ulug'bek kan lägga till tal mellan 1 och $M = 9$.

Proceduren kan returnera $[5, 4]$, vilket innebär att Ulug'bek beslutar sig för att lägga till $S = 2$ nya heltal: 5 och 4. Dessa är giltiga eftersom båda elementen ligger mellan 1 och $M = 9$.

Den utökade arrayen sorteras sedan och skickas vidare till Temur med följande anrop:

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

Vi kan dela upp heltalet i denna array i följande tre grupper: $[1, 5, 9]$, $[2, 5, 8]$ och $[4, 5, 6]$. Observera att summan av var och en av dessa grupper är lika med 15. För att rapportera denna uppdelning ska funktionen returnera arrayen $[0, 1, 2, 0, 1, 2, 2, 1, 0]$.

Exempelgrader

Inputformat:

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

Outputformat:

När anropet till `add_numbers` har slutförts gör exempelgradern följande:

- Beräknar den sorterade arrayen B .
- Skriver ut arrayerna C och B i följande format, **följt av en tom rad**:

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

När anropet till `find_partition` har slutförts visar betygsättaren följande:

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

Här är L längden på arrayen P som returneras av `find_partition`.