

Partition

Temur va uning yordamchisi Ulug'bek *Uzbekiston Got Talent* ko'rsatuvi uchun sehrli fokus tayyorlamoqda. Fokus Temurning quyidagi **guruhlash muammosi** ni hal qilishiga asoslanadi: berilgan musbat butun sonlar to'plamini K ta bo'sh bo'lmagan guruhlariga shunday bo'lish kerakki, har bir guruhdagi elementlar yig'indisi bir xil bo'lsin. Boshqacha qilib aytganda, to'plamdagi har bir butun son K guruhning aniq bittasiga birlashtirilishi va har bir guruhdagi yig'indi teng bo'lishi kerak. Masalan, agar berilgan to'plam $[2, 1, 6, 4, 5]$ va $K = 3$ bo'lsa, to'g'ri bo'linish $[2, 4]$, $[1, 5]$ va $[6]$ bo'ladi. Bu holda har bir guruhdagi elementlar yig'indisi aniq 6 ga teng bo'ladi.

Fokus quyidagicha bajariladi:

- Ulug'bek va Temur alohida kabinalarga kiradilar va bir-birlari bilan muloqot qila olmaydilar.
- Hakamlar Ulug'bekka N musbat butun sonlardan iborat A massivini beradi, $A[0], A[1], \dots, A[N-1]$, bu yerda har bir qiymat 1 va M oralig'ida bo'ladi. Hakamlar Ulug'bekka K qiymatini ham beradi.
- Ulug'bek A massiviga **yangi elementlar sifatida** eng ko'pi bilan $K - 1$ ta butun sonni (har xil bo'lishi shart emas) qo'shadi. Qo'shilgan har bir butun son ham 1 va M oralig'ida (inklusiv) bo'lishi kerak.
- Hakam Ulug'bek tanlagan butun sonlarni asl massivga qo'shadi. Ushbu kengaytirilgan massiv keyin **kamaymaydigan tartibda saralanadi** va Temurga K qiymati bilan birga beriladi.
- Temur keyin ushbu kengaytirilgan, saralangan massiv uchun guruhlash muammosini hal qilishi kerak.

Sizning vazifangiz Temur va Ulug'bek uchun strategiya ishlab chiqish va uni amalga oshirish. Strategiya doimo mavjud. Ya'ni, berilgan cheklovlar ostida, hakamlar tomonidan taqdim etilgan A massivdan qat'i nazar, ular uchun bo'lish muammosini muvaffaqiyatli hal qilish imkonini beruvchi strategiya mavjudligini isbotlash mumkin.

Implementation Details

Siz ikkita protsedurani shakllantirishingiz kerak.

Ulug'bek uchun ishlab chiqishingiz kerak bo'lgan protsedura quyidagicha:

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

- A : Ulug'bekka berilgan asl massivni ifodalovchi, uzunligi N bo'lgan massiv.
- K : bo'linishi kerak bo'lgan guruhlar soni; Ulug'bek massivga $K - 1$ ta gacha butun sonlarni qo'shishi mumkinligini unutmang.
- M : har bir asl va yangi qo'shilgan butun son uchun ruxsat etilgan maksimal qiymat.
- Ushbu protsedura har bir test uchun aniq bir marta chaqiriladi.

Protsedura C massivini qaytarishi kerak, unda Ulug'bek asl massivga qo'shmoqchi bo'lgan butun sonlar bo'ladi. S ni C massivining uzunligi deb belgilaymiz.

- S eng ko'pida $K - 1$ bo'lishi kerak.
- C ning har bir elementi 1 va M oralig'ida bo'lishi kerak.

Temur uchun ishlab chiqishingiz kerak bo'lgan protsedura quyidagicha:

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

- B : uzunligi $N + S$ bo'lgan massiv bo'lib, unda A dagi asl butun sonlar va Ulug'bek qo'shgan butun sonlar mavjud. B massivining elementlari kamaymaydigan tartibda saralangan.
- K : bo'linishi kerak bo'lgan guruhlar soni.
- Ushbu protsedura har bir test uchun aniq bir marta chaqiriladi.

Protsedura B ning K ta guruhlariga bo'linishini ifodalovchi P massivini qaytarishi kerak.

- P ning uzunligi $N + S$ bo'lishi kerak.
- Har bir j ($0 \leq j < N + S$) uchun, $P[j]$ bu $B[j]$ tegishli bo'lgan guruhni bildiradi.
- Guruhlar 0 dan $K - 1$ gacha raqamlanishi kerak, ya'ni har bir $0 \leq j < N + S$ uchun $0 \leq P[j] < K$ bo'lishi kerak.
- 0 va $K - 1$ oralig'idagi har bir i uchun, $P[j] = i$ bo'lgan kamida bitta j ($0 \leq j < N + S$) elementi bo'lishi kerak.
- K ta guruhning har biriga biriktirilgan elementlarning yig'indisi bir xil bo'lishi kerak.

Tizimdagi grader da yuqoridagi protseduralarni chaqiradigan dastur **aynan ikki marta** ishga tushiriladi.

- Dasturning birinchi ishga tushirilishida:
 - `add_numbers` aynan bir marta chaqiriladi.
 - Qaytarilgan massiv B massivini hisoblash uchun baholash tizimi tomonidan qayta ishlanadi.
- Dasturning ikkinchi ishga tushirilishida:
 - `find_partition` aynan bir marta chaqiriladi.

Constraints

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

- $K \leq N$
- $1 \leq M \leq 10^9$
- Har bir i ($0 \leq i < N$) uchun $1 \leq A[i] \leq M$.

Subtasks

Subtask	Ball	Qo'shimcha Cheklovlar
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	Qo'shimcha cheklovlarsiz

Example

Quyidagi chaqiruvni ko'raylik:

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

Ushbu misolda $A = [8, 2, 9, 6, 1, 5, 5]$. Biz raqamlarni $K = 3$ ta guruhga bo'lishni xohlaymiz. Ulug'bek 1 va $M = 9$ oralig'idagi sonlarni qo'sha oladi.

Ushbu protsedura $[5, 4]$ qaytarishi mumkin, ya'ni Ulug'bek $S = 2$ ta yangi butun sonni qo'shishga qaror qiladi: 5 va 4. Bu sonlar yaroqli, chunki ikkala element ham 1 va $M = 9$ orasida.

Kengaytirilgan massiv keyin saralanadi va quyidagi chaqiruv bilan Temurga uzatiladi:

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

Ushbu massivdagi butun sonlarni quyidagi uchta guruhga bo'lishimiz mumkin: $[1, 5, 9]$, $[2, 5, 8]$ va $[4, 5, 6]$. E'tibor bering, ushbu guruhlarining har biri yig'indisi 15 ga teng. Ushbu guruhlashni berish uchun, protsedura $[0, 1, 2, 0, 1, 2, 2, 1, 0]$ massivini qaytarishi kerak.

Sample Grader

Input format:

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

Output format:

`add_numbers` ga chaqiruv tugagandan so'ng, sizdagi grader:

- Saralangan B massivini hisoblaydi.
- C va B massivlarini quyidagi formatda, **oxirida bo'sh satr** bilan chop etadi:

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

`find_partition` funksiyasiga chaqiruv tugagandan so'ng, grader quyidagi natijani beradi:

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

Bu yerda L - bu `find_partition` tomonidan qaytarilgan P massivining uzunligi.