

Ayırma

Temur ve asistanı Ulug'bek, *Özbekistan'ın Yetenekleri* programı için bir sihirbazlık numarası hazırlıyorlar. Bu numara, Temur'un aşağıdaki **ayırma problemini** çözmesi üzerine kurulu: Verilen pozitif tamsayılar kümesini, her gruptaki elemanların toplamı eşit olacak şekilde K adet boş olmayan gruba ayırın. Başka bir deyişle, kümedeki her tamsayı, K gruptan tam olarak birine atanmalı ve her grup içindeki tamsayıların toplamı eşit olmalıdır. Örneğin, verilen küme $[2, 1, 6, 4, 5]$ ve $K = 3$ ise, geçerli bir ayırma şu şekilde olur: $[2, 4]$, $[1, 5]$ ve $[6]$. Bu durumda, her gruptaki elemanların toplamı tam olarak 6'dır.

Sihirbazlık numarası şu şekilde gerçekleştirilir:

- Ulug'bek ve Temur ayrı kabinlere girerler ve birbirleriyle iletişim kuramazlar.
- Bir jüri üyesi, Ulug'bek'e N pozitif tamsayıdan oluşan A dizisini verir; $A[0], A[1], \dots, A[N - 1]$, burada her değer 1 ile M arasında (bu değerler dahil) yer alır. Jüri üyesi ayrıca Ulug'bek'e K değerini de verir.
- Ulug'bek, A dizisine **yeni elemanlar olarak eklemek** üzere en fazla $K - 1$ (birbirlerinden farklı olması gerekmeyen) tamsayı seçer. Eklenen her tamsayı da 1 ile M aralığında (bu değerler dahil) olmalıdır.
- Jüri, Ulug'bek tarafından seçilen tamsayıları orijinal diziye ekler. Bu genişletilmiş dizi daha sonra **azalmayan sırayla sıralanır** ve K değeri ile birlikte Temur'a verilir.
- Temur daha sonra bu genişletilmiş, sıralanmış dizi için ayırma problemini çözmelidir.

Göreviniz, Temur ve Ulug'bek için bir strateji geliştirmek ve uygulamaktır. Verilen kısıtlamalar altında, jüri tarafından sağlanan A dizisinden bağımsız olarak, bu ikilinin ayırma problemini başarıyla çözmelerini sağlayan bir stratejinin var olduğu kanıtlanabilir.

Kodlama Detayları

İki prosedür kodlamalısınız.

Ulug'bek için kodlamanız gereken prosedür şudur:

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

- A : Ulug'bek'e verilen orijinal diziye temsil eden, uzunluğu N olan bir dizi.
- K : hedeflenen grup sayısı; Ulug'bek'in diziye en fazla $K - 1$ tane tamsayı ekleyebileceğini unutmayın.

- M : her bir orijinal ve yeni eklenen tamsayı için izin verilen maksimum değer.
- Bu prosedür, her test durumu için tam olarak bir kez çağrılır.

Prosedür, Ulug'bek'in orijinal diziye eklemek istediği tamsayıları içeren C dizisini döndürmelidir. S , C dizisinin uzunluğunu temsil etsin.

- S , en fazla $K - 1$ olmalıdır.
- C dizisinin her elemanı, 1 ile M aralığında (bu değerler dahil) olmalıdır.

Temur için kodlamanız gereken prosedür şudur:

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

- B : $N + S$ uzunluğunda, hem A adresindeki orijinal tamsayıları hem de Ulug'bek'in eklediği tamsayıları içeren bir dizi. B dizisinin elemanları azalmayan sırayla sıralanmıştır.
- K : hedef grup sayısı.
- Bu prosedür, her test durumu için tam olarak bir kez çağrılır.

Prosedür, B dizisinin K adet birbirinden ayrı gruba ayrılmasını temsil eden P dizisini döndürmelidir.

- P dizisinin uzunluğu $N + S$ olmalıdır.
- $0 \leq j < N + S$ olan her j için, $P[j]$, $B[j]$ 'nin ait olduğu grubu belirtir.
- Gruplar, 0'dan $K - 1$ 'e kadar numaralandırılmalıdır; yani, her $0 \leq j < N + S$ için $0 \leq P[j] < K$ şartı geçerli olmalıdır.
- 0 ile $K - 1$ arasındaki her bir i için, $P[j] = i$ koşulunu sağlayan en az bir j ($0 \leq j < N + S$) elemanı bulunmalıdır.
- K grup için her birine atanan elemanların toplamı aynı olmalıdır.

Kodunuz değerlendirilirken, yukarıdaki prosedürleri çağırarak bir program **tam olarak iki kez** çalıştırılır.

- Programın ilk çalıştırılması sırasında:
 - `add_numbers` tam olarak bir kez çağrılır.
 - Döndürülen dizi, B dizisini hesaplamak üzere değerlendirme sistemi tarafından işlenir.
- Programın ikinci çalıştırılması sırasında:
 - `find_partition` tam olarak bir kez çağrılır.

Kısıtlamalar

- $3 \leq N \leq 100\,000$
- $2 \leq K \leq 100\,000$
- $K \leq N$
- $1 \leq M \leq 10^9$
- $1 \leq A[i] \leq M$, her i için $0 \leq i < N$ olacak şekilde.

Alt Görevler

Alt Görev	Puan	Ek Kısıtlamalar
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	Ek kısıtlama yoktur.

Örnek

Aşağıdaki çağrıyı ele alalım:

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

Bu örnekte $A = [8, 2, 9, 6, 1, 5, 5]$. Sayıları $K = 3$ gruba ayırmak istiyoruz. Ulug'bek, 1 ile $M = 9$ arasındaki sayıları ekleyebilir.

Prosedür, $[5, 4]$ sonucunu döndürebilir; bu, Ulug'bek'in $S = 2$ yeni tamsayı eklemeye karar verdiği anlamına gelir: 5 ve 4. Her iki eleman da 1 ile $M = 9$ arasında olduğu için bunlar geçerlidir.

Genişletilmiş dizi daha sonra sıralanır ve aşağıdaki çağrı ile Temur'a aktarılır:

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

Bu dizideki tamsayıları şu üç gruba ayırabiliriz: $[1, 5, 9]$, $[2, 5, 8]$ ve $[4, 5, 6]$. Bu grupların her birinin toplamının 15'ye eşit olduğuna dikkat edin. Bu ayırmayı bildirmek için, prosedür $[0, 1, 2, 0, 1, 2, 2, 1, 0]$ dizisini döndürmelidir.

Örnek Değerlendirici

Girdi biçimi:

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

Çıktı biçimi:

`add_numbers` çağrısı tamamlandıktan sonra, örnek değerlendirici:

- Sıralanmış B dizisini hesaplar.
- C ve B dizilerini aşağıdaki biçimde yazdırır ve sonrasında **boş bir satırla takip eder**:

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

`find_partition` çağrısı tamamlandıktan sonra, değerlendirici şunları çıktılar:

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

Burada, L , `find_partition` tarafından döndürülen P dizisinin uzunluğudur.