

Partition

Ο Temur και ο βοηθός του, ο Ulug'bek, ετοιμάζουν ένα μαγικό κόλπο για την εκπομπή *Uzbekistan Got Talent*. Το κόλπο σχετίζεται με την επίλυση του ακόλουθου **προβλήματος διαμέρισης** από τον Temur:

Να διαμερίσει (partition) μια δεδομένη συλλογή θετικών ακεραίων σε K μη κενές ομάδες έτσι ώστε το άθροισμα των στοιχείων σε κάθε ομάδα να είναι το ίδιο. Με άλλα λόγια, κάθε ακέραιος αριθμός από τη συλλογή πρέπει να ανήκει σε ακριβώς μία από τις K ομάδες και το άθροισμα των στοιχείων της κάθε ομάδας πρέπει να είναι ίσο.

Για παράδειγμα, εάν η δεδομένη συλλογή είναι $[2, 1, 6, 4, 5]$ και $K = 3$, μια έγκυρη διαμέριση θα είναι $[2, 4]$, $[1, 5]$ και $[6]$. Σε αυτήν την περίπτωση, το άθροισμα των στοιχείων σε κάθε ομάδα είναι ακριβώς 6.

Το κόλπο εκτελείται ως εξής:

- Ο Ulug'bek και ο Temur μπαίνουν σε ξεχωριστά δωμάτια και δεν μπορούν να επικοινωνήσουν μεταξύ τους.
- Η κριτική επιτροπή δίνει στον Ulug'bek έναν πίνακα A με N θετικούς ακεραίους αριθμούς, $A[0], A[1], \dots, A[N-1]$, όπου κάθε τιμή είναι μεταξύ 1 και M συμπεριλαμβανομένων. Η κριτική επιτροπή επίσης δίνει στον Ulug'bek την τιμή K .
- Ο Ulug'bek επιλέγει το πολύ $K-1$ (όχι απαραίτητα διαφορετικούς) ακεραίους αριθμούς για να **προσθέσει ως νέα στοιχεία** στον πίνακα A . Κάθε ακέραιος αριθμός που προστίθεται πρέπει επίσης να είναι μεταξύ 1 και M συμπεριλαμβανομένων.
- Ένα μέλος της κριτικής επιτροπής προσθέτει τους ακεραίους αριθμούς που επέλεξε ο Ulug'bek στον αρχικό πίνακα. Αυτός ο εκτεταμένος (extended) πίνακας στη συνέχεια **ταξινομείται σε αύξουσα σειρά** και δίνεται στον Temur, μαζί με την τιμή του K .
- Ο Temur πρέπει στη συνέχεια να λύσει το πρόβλημα διαμέρισης για αυτόν τον εκτεταμένο (extended) ταξινομημένο πίνακα.

Η αποστολή σου είναι να σχεδιάσεις και να εφαρμόσεις μια στρατηγική για τον Temur και μία για τον Ulug'bek. Μπορεί να αποδειχθεί ότι, υπό τους περιορισμούς που δίνονται, υπάρχει στρατηγική που τους επιτρέπει να λύσουν το πρόβλημα διαμέρισης, άσχετα από τον πίνακα A που δίνεται από την κριτική επιτροπή.

Λεπτομέρειες Υλοποίησης

Θα πρέπει να υλοποιήσετε δύο διαδικασίες.

Η συνάρτηση που πρέπει να υλοποιήσεις για το **Ulug'bek** είναι:

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

- A : ένας πίνακας μήκους N που αντιπροσωπεύει τον αρχικό πίνακα που δόθηκε στον Ulug'bek.
- K : ο ζητούμενος αριθμός ομάδων. Να σημειωθεί ότι ο Ulug'bek μπορεί να προσθέσει μέχρι $K - 1$ ακέραιους στον πίνακα.
- M : η μέγιστη επιτρεπόμενη τιμή για κάθε αρχικό και νέο ακέραιο αριθμό.
- Αυτή η συνάρτηση καλείται ακριβώς μία φορά για κάθε περίπτωση δοκιμής.

Η συνάρτηση θα πρέπει να επιστρέψει έναν πίνακα C που περιέχει τους ακέραιους αριθμούς που ο Ulug'bek θέλει να προσθέσει στον αρχικό πίνακα. Έστω S το μήκος του πίνακα C .

- S πρέπει να είναι το πολύ $K - 1$.
- Κάθε στοιχείο του C πρέπει να κυμαίνεται μεταξύ 1 και M συμπεριλαμβανομένων.

Η συνάρτηση που πρέπει να υλοποιήσεις για τον **Temur** είναι:

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

- B : ένας πίνακας μήκους $N + S$ που περιέχει τόσο τους αρχικούς ακέραιους από τον A όσο και τους ακέραιους που πρόσθεσε ο Ulug'bek. Τα στοιχεία του πίνακα B ταξινομούνται σε αύξουσα σειρά.
- K : ο ζητούμενος αριθμός ομάδων.
- Αυτή η συνάρτηση καλείται ακριβώς μία φορά για κάθε περίπτωση ελέγχου.

Η συνάρτηση θα πρέπει να επιστρέψει έναν πίνακα P που αντιπροσωπεύει την διαμέριση του B σε K ασύνδετες (disjoint) ομάδες.

- Το μήκος του P θα πρέπει να είναι $N + S$.
- Για κάθε j τέτοιο ώστε $0 \leq j < N + S$, $P[j]$ δηλώνει την ομάδα στην οποία ανήκει $B[j]$.
- Οι ομάδες θα πρέπει να αριθμούνται από 0 έως $K - 1$, που σημαίνει ότι $0 \leq P[j] < K$ πρέπει να ισχύει για κάθε $0 \leq j < N + S$.
- Για κάθε i μεταξύ 0 και $K - 1$ συμπεριλαμβανομένων, πρέπει να υπάρχει τουλάχιστον ένα στοιχείο j ($0 \leq j < N + S$) τέτοιο ώστε $P[j] = i$.
- Το άθροισμα των στοιχείων που έχουν τοποθετηθεί σε κάθε μία από τις ομάδες K πρέπει να είναι ίδιο.

Κατά την πραγματική βαθμολόγηση, ένα πρόγραμμα που καλεί τις παραπάνω διαδικασίες εκτελείται **ακριβώς δύο φορές**.

- Κατά την πρώτη εκτέλεση του προγράμματος:
 - Η συνάρτηση `add_numbers` καλείται ακριβώς μία φορά.
 - Ο επιστρεφόμενος πίνακας επεξεργάζεται από το σύστημα βαθμολόγησης για τον υπολογισμό του πίνακα B .
- Κατά τη δεύτερη εκτέλεση του προγράμματος:
 - Η συνάρτηση `find_partition` καλείται ακριβώς μία φορά.

Περιορισμοί

- $3 \leq N \leq 100\,000$
- $2 \leq K \leq 100\,000$
- $K \leq N$
- $1 \leq M \leq 10^9$
- $1 \leq A[i] \leq M$ για κάθε i τέτοιο ώστε $0 \leq i < N$.

Υποπροβλήματα

Υποπρόβλημα	Βαθμοί	Πρόσθετοι περιορισμοί
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	Χωρίς πρόσθετους περιορισμούς

Παράδειγμα

Υποθέτουμε την επόμενη κλήση:

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

Σε αυτό το παράδειγμα $A = [8, 2, 9, 6, 1, 5, 5]$. Θέλουμε να χωρίσουμε τους αριθμούς σε ομάδες $K = 3$. Ο Ulug'bek μπορεί να προσθέσει αριθμούς μεταξύ 1 και $M = 9$.

Η διαδικασία μπορεί να επιστρέψει $[5, 4]$, που σημαίνει ότι ο Ulug'bek αποφασίζει να προσθέσει $S = 2$ νέους ακέραιους αριθμούς: 5 και 4 . Αυτά ισχύουν καθώς και τα δύο στοιχεία είναι μεταξύ 1 και $M = 9$.

Ο εκτεταμένος πίνακας στη συνέχεια ταξινομείται και διαβιβάζεται στον Temur με την ακόλουθη κλήση:

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

Μπορούμε να διαιρέσουμε τους ακέραιους αριθμούς από αυτόν τον πίνακα σε αυτές τις τρεις ομάδες: $[1, 5, 9]$, $[2, 5, 8]$, και $[4, 5, 6]$. Σημειώστε ότι το άθροισμα καθεμίας από αυτές τις ομάδες είναι ίσο με 15 . Για να αναφερθεί αυτή η διαμέριση, η διαδικασία θα πρέπει να επιστρέψει τον πίνακα $[0, 1, 2, 0, 1, 2, 2, 1, 0]$.

Δειγματικός Βαθμολογητής

Μορφή εισόδου:

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

Μορφή εξόδου:

Αφού ολοκληρωθεί η κλήση της συνάρτησης `add_numbers` , ο δειγματικός βαθμολογητής:

- Υπολογίζει τον ταξινομημένο πίνακα B .
- Τυπώνει τους πίνακες C και B στην ακόλουθη μορφή, **ακολουθούμενους από μια κενή γραμμή**:

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

Αφού ολοκληρωθεί η κλήση του `find_partition`, ο βαθμολογητής εμφανίζει την εξής έξοδο:

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

Εδώ, L είναι το μήκος του πίνακα P που επιστρέφεται από `find_partition` .