☆ Custom-Sorted Array

In an array, we can swap the elements at any two indices in a single operation called a *move*. For example, if our array is $a = [17, 4, 8]$, we can swap $a[0] = 17$ and $a[2] = 8$ to get $a = [8, 4, 17]$ in a single move. We want to custom-sort an array such that all of the *even* elements are at the beginning of the array and all of the *odd* elements are at the end of the array.

For example, if our array is $[6, 3, 4, 5]$, then the following four arrays are valid custom-sorted arrays:

- $a = [6, 4, 3, 5]$
- $a = [4, 6, 3, 5]$
- $a = [6, 4, 5, 3]$
- $a = [4, 6, 5, 3]$

Function Description

Complete the function *moves* in the editor below. The function must return the minimum number of moves it takes to sort an array of integers with all even elements at earlier indexes than any odd element.

moves has the following parameter(s):

a[a[0],...a[n-1]]: an array of positive integers

Note: The order of the elements within even or odd does not matter.

Constraints

- $2 \le n \le 10^5$
- $1 \le a[i] \le 10^9$, where $0 \le i < n$.
- It is guaranteed that array *a* contains at least one *even* and one *odd* element.

▸ **Input Format for Custom Testing**

▾ **Sample Case 0**

▶ Input Format for Custom Testing

▼ Sample Case 0

Sample Input 0

```
4
13
10
21
20
```

Sample Output 0

```
1
```

Explanation 0

Given *a = [13, 10, 21, 20]*, we can swap *a[0]* and *a[3]* to get the custom-sorted array *a = [20, 10, 21, 13]* in 1 move.

▼ Sample Case 1

Sample Input 1

```
5
8
5
11
4
6
```

Sample Output 1

```
2
```

Jenny is a big sports fan, especially uberball.

Rules of uberball are following :

- The match is played by 2 teams
- During one round of the game, a team scores a point, and thus increases its score by 1.
- Both team starts with 0 points each.

The game ends when

- One of the teams gets 25 points and another team has < 24 points (strictly less than 24).
- If the score ties at 24:24, the teams continue to play until the absolute difference between the scores is 2.

After the game has ended, the final score is given in the format X:Y, which means the first team has scored X points and the second has scored Y points, can you find the number of different sequences of getting points by teams that leads to this final score?

Complete the function uberball in your editor. It has 2 parameters:
An integer X.
An integer Y.

It must return the number of different possible sequences of getting those points. As the answer could be very large, return the value of result % (10^9 + 7).

Input Format
The locked stub code in your editor reads the following input from stdin and passes it to your function:
The first line contains a single integer X.
The next line contains a single integer Y.

Constraints
$0 \leq X,Y \leq 10^9$

Output Format
The locked code in the editor prints the return value of the function.
Your function must return the number of different possible sequences of getting those points. As the answer could be very large, return the value of result % (10^9 + 7).

It must return the number of different possible sequences of getting those points. As the answer could be very large, return the value of result % (10^9 + 7).

Input Format
The locked stub code in your editor reads the following input from stdin and passes it to your function:
The first line contains a single integer X.
The next line contains a single integer Y.

Constraints
$0 \le X,Y \le 10^9$

Output Format
The locked code in the editor prints the return value of the function.
Your function must return the number of different possible sequences of getting those points. As the answer could be very large, return the value of result % (10^9 + 7).

Sample Input 1
3
25

Sample Output 1
2925

Explanation 1
There are 2925 different sequences to reach the score (3,25).

Sample Input 2
24
17

Sample Output 2
0

Uber Shuttle Problem

An imaginary city Z has N(<= 20) junctions and M(<= N*(N-1)/2) roads connecting these junctions. There are N-2 passengers in total, one in each junction from 2,3,...,N-1. An uber shuttle currently at junction 1 in the city needs to pick up the N - 2 people by going to those junctions and finally reach junction N. You are given the time taken to traverse each of the M roads and the pair of junctions each of them connect. All roads are bidirectional. You can use the same road multiple times (if required). You can also pickup the people in any order you want. You need to find the minimum time needed to finally reach junction N after picking up everyone. (Your intermediate route can also pass through N. It doesn't affect the answer). Output this minimum time. If it is not possible to either reach the destination or pick up someone, output -1.

Constraints

1 <= N <= 20

0 <= M <= N*(N-1)/2

1 <= time taken to traverse each road <= 10^5

Nodes are labelled from 1 to N.

Input graph does not contain multi edges, i.e. between any 2 pairs of cities there exists at most one edge.

Input format

First line contains 2 integers N,M as defined in the problem statement.

Next M lines contains 3 integers u v w each representing the 2 junctions the road connects and the time taken to traverse that road respectively.

Output format

Output 1 integer representing the minimum time required to pick up everyone and reach the destination.

Output -1 if the task cannot be completed.

Sample input

5 5

1 3 20

2 4 100

2 3 40

2 5 10

4 5 15

Sample output

100

Explanation

In this case, it is optimal to pick up 3,2,4 in that order and going from 2 to 4 through 5. The best route is 1->3->2->5->4->5 and the total time taken on this route is 100.

1 3 20
2 4 100
2 3 40
2 5 10
4 5 15

Sample output

100

Explanation

In this case, it is optimal to pick up 3,2,4 in that order and going from 2 to 4 through 5. The best route is 1->3->2->5->4->5 and the total time taken on this route is 100.
You can verify that this is the optimal solution and all other routes take longer time to complete.

YOUR ANSWER

We recommend you take a quick tour of our editor before you proceed. The timer will pause up to 90 seconds for the tour. Start tour

Original code | C++14

```
#include <map>
#include <set>
#include <list>
#include <cmath>
#include <ctime>
#include <deque>
#include <queue>
#include <stack>
#include <string>
#include <bitset>
#include <cstdio>
#include <limits>
#include <vector>
#include <climits>
```