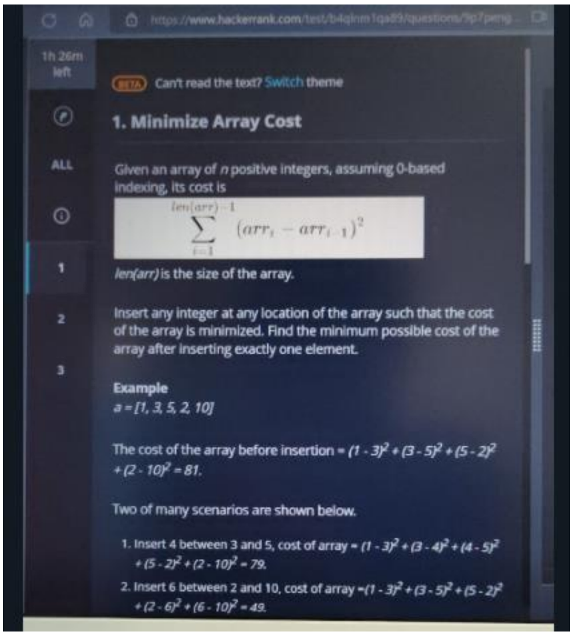
Atlassian Questions

1)



Can't read the text? Switch theme

1. Minimize Array Cost

Given an array of *n* positive integers, assuming 0-based indexing, its cost is

$$\sum_{i=1}^{len(arr)-1} (arr_i - arr_{i-1})^2$$

len(arr) is the size of the array.

Insert any integer at any location of the array such that the cost of the array is minimized. Find the minimum possible cost of the array after inserting exactly one element.

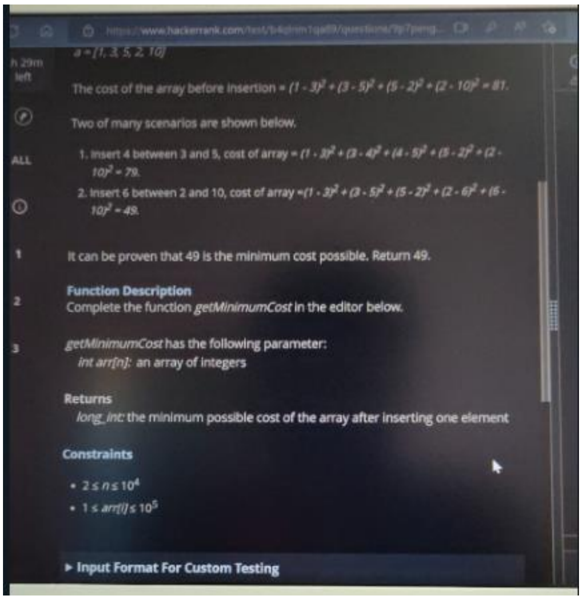
Example

a = [1, 3, 5, 2, 10]

The cost of the array before insertion = $(1 - 3)^2 + (3 - 5)^2 + (5 - 2)^2 + (2 - 10)^2 = 81$.

Two of many scenarios are shown below.

1. Insert 4 between 3 and 5, cost of array = $(1 - 3)^2 + (3 - 4)^2 + (4 - 5)^2 + (5 - 2)^2 + (2 - 10)^2 = 79$.
2. Insert 6 between 2 and 10, cost of array = $(1 - 3)^2 + (3 - 5)^2 + (5 - 2)^2 + (2 - 6)^2 + (6 - 10)^2 = 49$.



a = [1, 3, 5, 2, 10]

The cost of the array before insertion = $(1 - 3)^2 + (3 - 5)^2 + (5 - 2)^2 + (2 - 10)^2 = 81$.

Two of many scenarios are shown below.

1. Insert 4 between 3 and 5, cost of array = $(1 - 3)^2 + (3 - 4)^2 + (4 - 5)^2 + (5 - 2)^2 + (2 - 10)^2 = 79$.
2. Insert 6 between 2 and 10, cost of array = $(1 - 3)^2 + (3 - 5)^2 + (5 - 2)^2 + (2 - 6)^2 + (6 - 10)^2 = 49$.

It can be proven that 49 is the minimum cost possible. Return 49.

Function Description

Complete the function *getMinimumCost* in the editor below.

getMinimumCost has the following parameter:

int arr[n]: an array of integers

Returns

long_int: the minimum possible cost of the array after inserting one element

Constraints

- $2 \le n \le 10^4$
- $1 \le arr[i] \le 10^5$

▸ Input Format For Custom Testing

2)

BETA Can't read the text? Switch theme

2. Organize Encyclopedias

Cole volunteers as a shelver at the school library, where the encyclopedias need to be re-organized. After re-organizing all the encyclopedias, Cole realizes that some of the encyclopedias are shelved in the wrong order. To fix the mistake, Cole decides to remove all the encyclopedias from the *n* by *m* shelf. While reshelving the encyclopedias, Cole realizes that the most efficient way to reshelve the encyclopedias is to select one encyclopedia, remove it, and remove all the other encyclopedias in the same row and column that the author wrote of the encyclopedia Cole chose. Cole wants to determine the minimum number of encyclopedias that need to be selected to remove all the encyclopedias from the shelf.

Note: The encyclopedias shelf section contains *k* authors. Each author is represented by an integer from 1 to *k*.

Example
Let there be $k = 3$ authors. Let the following matrix represent the encyclopedias shelf:

```
2 2 1
1 1 1
2 3 3
```

Cole can choose the encyclopedia at position (2,3) in the first operation and remove it. Then, the matrix looks like this:

Example
Let there be $k = 3$ authors. Let the following matrix represent the encyclopedias shelf:

```
2 2 1
1 1 1
2 3 3
```

Cole can choose the encyclopedia at position (2,3) in the first operation and remove it. Then, the matrix looks like this:

```
2 2 x
x x x
2 3 3
```

Cole can choose the encyclopedia at position (1,1) in the second operation and remove it. Then, the matrix looks like this:

```
x x x
x x x
x 3 3
```

Cole can choose the encyclopedia at position (3,3) in the third operation and remove it. Then, the matrix looks like this:

```
x x x
x x x
x x x
```

Therefore, the minimum number of encyclopedias to select from the shelf is 3.

Function Description
Complete the function *findMinimumOperations* in the editor below.

```
x x x
x x x
x x x
```

Therefore, the minimum number of encyclopedias to select from the shelf is 3.

Function Description
Complete the function *findMinimumOperations* in the editor below.

findMinimumOperations has the following parameter(s):
bookshelf: a 2-D integer array of dimension *n* by *m*, representing the encyclopedia books
k: the authors of the book have IDs between 1 and *k*

Returns
int: an integer denoting the minimum number of operations to remove all the encyclopedias off the bookshelf.

Constraints

- $1 \le n \le 100$
- $1 \le m \le 100$
- $1 \le k \le 50$

▸ **Input Format For Custom Testing**

▾ **Sample Case 0**

Sample Input For Custom Testing

3)

3. Constructing a Network

There are a number of computers ready to be connected to some network which will be viewed as a graph of nodes. Each computer has a particular signal strength equal to its node number. Each second, two computers get connected to each other. A group of computers connected to one another (directly or indirectly) forms a network. The strength of the network is defined as the maximum signal strength of all the computers present in the network. A single computer forms a network with strength equal to the signal strength of the computer itself. Also, it is possible that the computers present in the same network get connected to each other. After each second, determine the sum of the strengths of all networks.

Example

c_nodes = 5
c_edges = 4
c_from = [1, 2, 1, 4]
c_to = [2, 3, 3, 5]

In the i^{th} second, the computers numbered *c_from[i]* and *c_to[i]* get connected.

Initial state:

- Network 1 = {1}, Strength = 1
- Network 2 = {2}, Strength = 2
- Network 3 = {3}, Strength = 3

In the i^{th} second, the computers numbered *c_from[i]* and *c_to[i]* get connected.

Initial state:

- Network 1 = {1}, Strength = 1
- Network 2 = {2}, Strength = 2
- Network 3 = {3}, Strength = 3
- Network 4 = {4}, Strength = 4
- Network 5 = {5}, Strength = 5

At the 0^{th} second, computers *1* and *2* get connected. *sum = 2 + 3 + 4 + 5 = 14*

- Network 1 = {1, 2}, Strength = 2
- Network 2 = {3}, Strength = 3
- Network 3 = {4}, Strength = 4
- Network 4 = {5}, Strength = 5

At the 1^{st} second, computers *2* and *3* get connected. *sum = 3 + 4 + 5 = 12*

- Network 1 = {1, 2, 3}, Strength = 3
- Network 3 = {4}, Strength = 4
- Network 4 = {5}, Strength = 5

At the 2^{nd} second, computers *1* and *3* get connected. *sum = 3 + 4 + 5 = 12*

- Network 1 = {1, 2, 3}, Strength = 3
- Network 3 = {4}, Strength = 4
- Network 4 = {5}, Strength = 5

At the 3^{rd} second, computers *4* and *5* get connected. *sum = 3 + 5 = 8*

- Network 1 = {1, 2, 3}, Strength = 3

At the 3^{rd} second, computers *4* and *5* get connected. *sum = 3 + 5 = 8*

- Network 1 = {1, 2, 3}, Strength = 3
- Network 3 = {4, 5}, Strength = 5

The sums are returned in an array, *[14, 12, 12, 8]*.

Function Description

Complete the function *networkSums* in the editor below.

networkSums has the following parameter(s):
  int *c_nodes*: the number of nodes
  int *c_edges*: the number of connections added during a period of *c_edges* seconds
  int *c_from[c_edges]*: each node *c_from[i]* is an end of the i^{th} edge
  int *c_to[c_edges]*: each node *c_to[i]* is an end of the i^{th} edge

Returns:
  int[]: the sums of the strengths of all networks after each second

Constraints

- $1 \le c\_nodes \le 10^5$
- $1 \le c\_edges \le 3 * 10^5$
- $1 \le c\_from \le c\_nodes$
- $1 \le c\_to \le c\_nodes$

▸ **Input Format For Custom Testing**