

# AMAZON OA 整理

---

## 1. getkConsistency score

---

删掉最多K个元素得到元素相同的subarray，这个subarray最长的可能长度。思路：  
hashmap<int,vector> key是stockprice， value是出现的index。然后对于每个元素做sliding window，  
找到最长的可能。

<https://www.1point3acres.com/bbs/thread-917161-1-1.html>



6m left

## 2. Code Question 2



ALL



1

2

A team of financial analysts at Amazon has designed a stock indicator to determine the consistency of Amazon's stock in delivering returns daily. More formally, the percentage return (rounded off to nearest integer) delivered by the stock each day over the last  $n$  days is considered. This is given as an array of integers, *stock\_prices*. The stock's *k-consistency score* is calculated using the following operations:

- In a single operation, omit a particular day's return from the *stock\_prices* array to get have one less element, then rejoin the parts of the array. This can be done at most  $k$  times.
- The maximum number of contiguous days during which the daily return was the same is defined as the *k-consistency score* for Amazon's stock. Note that the return may be positive or negative.

As part of the team, you have been assigned to determine the *k-consistency score* for the stock. You are given an array *stock\_prices* of size  $n$  representing the daily percentage return delivered by Amazon stock and a parameter  $k$ .

Determine the *k-consistency score*.

### Example:

Consider the percentage return delivered by Amazon's Stock in the last 8 days is represented as *stock\_prices* = [1, -2, 1, 1, 3, 2, 1, -2] and  $k = 3$ .



### Function Description

Complete the function *getkConsistency* in the editor below.

*getkConsistency* has the following parameters:

- int stock\_prices[n]*: the percentage return for Amazon stock over the last  $n$  days
- int k*: the maximum number of days that can be removed

### Returns

*int*: the maximum possible *k-consistency score*

### Constraints

- $1 \leq n \leq 3 \cdot 10^5$
- $-10^9 \leq \text{stock\_prices}[i] \leq 10^9$
- $0 \leq k \leq 3 \cdot 10^5$

▶ Input Format For Custom Testing

### ▼ Sample Case 0

#### Sample Input For Custom Testing

```
STDIN      FUNCTION
-----  -----
6      →  n = 6
1      →  stock_prices = [1, 1, 2, 1,
2, 1]
1
2
1
2
1
```



@一亩三分地

#### Function Description

Complete the function `getkConsistency` in the editor below.

`getkConsistency` has the following parameters:

`int stock_prices[n]`: the percentage return for Amazon stock over the last  $n$  days

`int k`: the maximum number of days that can be removed

#### Returns

`int`: the maximum possible  $k$ -consistency score

#### Constraints

- $1 \leq n \leq 3 \cdot 10^5$
- $-10^9 \leq stock\_prices[i] \leq 10^9$
- $0 \leq k \leq 3 \cdot 10^5$

### ► Input Format For Custom Testing

### ▼ Sample Case 0

#### Sample Input For Custom Testing

```
STDIN      FUNCTION
-----  -----
6      →  n = 6
1      →  stock_prices = [1, 1, 2, 1,
2, 1]
1
2
1
2
1
2
1
3      →  k = 3
```



@一亩三分地

#### Sample Output

### Explanation

After omitting the integers at the 3<sup>rd</sup> and the 5<sup>th</sup> positions, the array is [1, 1, 1, 1]. It is better not to do the remaining allowable operation.

#### ▼ Sample Case 1

##### Sample Input For Custom Testing

STDIN	FUNCTION
8	→ n = 8
7	→ stock_prices = [7, 5, 7, 7,
1, 1, 7, 7]	
5	
7	
7	
1	
1	
7	
7	
3	→ k = 3

##### Sample Output

5

### Explanation

After omitting the integers at the 2<sup>nd</sup>, 5<sup>th</sup>, and the 6<sup>th</sup> positions, the array is [7, 7, 7, 7, 7].



@一亩三分地

```

int main() {
    //vector<int> stockPrice={1,1,2,1,2,1};
    vector<int> stockPrice={7,5,7,7,1,1,7,7};
    int k=3;
    int result=0;
    unordered_map<int,vector<int>> m;
    for(int i=0;i<stockPrice.size();i++)
        m[stockPrice[i]].push_back(i);
    for(auto tmp:m){
        int currentLength = 0;
        int n=tmp.second.size();
        if(n==1) continue;
        int deleted=tmp.second[1]-tmp.second[0]-1;
        int left=0;int right=1;
        while(left<n && right<n){
            currentLength = max(right-left+1,currentLength);
            while(deleted>k){
                left++;
                deleted -= tmp.second[left]-tmp.second[left-1]-1;
            }
            right++;
            if(right<n) deleted += tmp.second[right]-tmp.second[right-1]-1;
        }
        result = max(result,currentLength);
    }
}

```

```
    cout<<result;
    return 0;
}
```

## 2. LC 973: 找最近的餐馆

---

3m 8s left

BETA Can't read the text? [Switch theme](#)

## 1. Code Question 1

ALL

Amazon's Alexa team is working on optimizing the customer experience for scenarios where customers ask generic questions. One example of a generic question is "What are good vegetarian restaurants nearby?" In this example, Alexa would then search for a list of vegetarian restaurants in the city and select the nearest X vegetarian restaurants relative to the customer's location. Given an array representing the locations of N vegetarian restaurants in the city, implement an algorithm to find the nearest X vegetarian restaurants to the customer's location.

1

### Input

2

The input to the function/method consists of two arguments:

3

*allLocations*, a list of elements where each element consists of a pair of integers representing the x and y coordinates of the vegetarian restaurant in the city; *numRestaurants*, an integer representing the number of nearby vegetarian restaurants that would be returned to the customer (X).

### Output

Return a list of elements where each element of the list represents the x and y integer coordinates of the nearest recommended vegetarian restaurant relative to customer's location. If there is one tie, use the location with the closest X coordinate. If no location is possible, return a list with an empty location - not just

Amazon Software Development HackerRank Question - Code Qu +

hackerrank.com/test/7rpe0tqbr00/questions/9ijji3tqop4

54s left

1

### Output

Return a list of elements where each element of the list represents the x and y integer coordinates of the nearest recommended vegetarian restaurant relative to customer's location. If there is one tie, use the location with the closest X coordinate. If no location is possible, return a list with an empty location - not just an empty list.

2

### Constraints

*numRestaurants*  $\leq \text{size}(\text{allLocations})$

3

### Note

The customer begins at the location [0, 0].

The distance from the customer's current location to a recommended vegetarian restaurant location (x, y) is the square root of  $x^2 + y^2$ .

If there are ties then return any of the locations as long as you satisfy returning exactly X nearby vegetarian restaurants.

The returned output can be in any order.

4

### Example

Input:

*allLocations* = [[1, 2], [3, 4], [1, -1]]  
*numRestaurants* = 2

Output:

Language

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

Output:  
[[1, -1], [1, 2]]

Explanation:

The distance of the customer's current location from  
location [1, 2] is square root(5) = 2.236



Brute Force:

```
class Solution {  
public:  
    vector<vector<int>> kClosest(vector<vector<int>>& points, int k) {  
        sort(points.begin(), points.end(), [] (const auto & p1, const auto & p2) {  
            return p1[0]*p1[0]+p2[0]*p2[0]+p2[1]*p2[1]-p1[1]*p1[1];});  
        vector<vector<int>> result;  
        for(int i=0;i<k;i++)  
            result.push_back(points[i]);  
        return result;  
    }  
};
```

快排:

```
class Solution {  
public:  
    vector<vector<int>> kClosest(vector<vector<int>>& points, int k) {  
        quickSort(points, 0, points.size() - 1);  
        return vector<vector<int>>(points.begin(), points.begin() + k);  
    }  
private:  
    void quickSort(vector<vector<int>>& points, int low, int high){  
        if(low >= high) return;  
        int pivotIndex = partition(points, low, high);  
        quickSort(points, low, pivotIndex-1);  
        quickSort(points, pivotIndex+1, high);  
    }  
    int partition(vector<vector<int>>& points, int low, int high){  
        auto pivot = distance(points[low]);  
        int i=low+1;  
        int j=high;  
        while(true){  
            while(i<high && distance(points[i])<pivot) i++;  
            while(j>low && distance(points[j]) > pivot) j--;  
            if(i>=j)  
                break;  
            swap(points[i],points[j]);  
            i++;  
            j--;  
        }  
        swap(points[low],points[j]);  
        return j;  
    }  
    long distance(vector<int>& p){  
        return p[0]*p[0]+p[1]*p[1];  
    }  
};
```

### 3. combo 问题

---

先排序。

把所有整数相加得到最大值。

那接下来就是 减去正数 或者加上负数

如果我们把负数都变成整数，那就统一变成了减去正数。

然后问题就变成了有一个array，求前k个最小的subarray的和，这题用priority\_queue做，里面装 pair<long,int>前者为subarray的和，后者为当前subarray的最后一个元素。

subarray move到下一个元素时要考虑两种情况：包含当前元素和不包含当前元素。

<https://leetcode.com/discuss/interview-question/1895396/amazon-sde-new-grad-qa-k-most-popular-combos>

Amazon Software Development X HackerRank Question - Code Q

hackerrank.com/test/7rpe0tqbr00/questions/3633mg39g6

2m 45s left BETA Can't read the text? [Switch theme](#)

### 3. Code Question 2

**ALL** Amazon Shopping has  $n$  items in inventory. Each item has a rating that may be negative. The team wants to work on an algorithm that will suggest combinations of these items that customers might buy, or, *combos* for short. A combo is defined as a subset of the given  $n$  items. The total popularity of a combo is the sum of the popularities of the individual items in the combo. Design an algorithm that can find the  $k$  combos with the highest popularities. Two combos are considered different if they have a different subset of items. Return an array of  $k$  integers where the  $i^{th}$  denotes the popularity of  $i^{th}$  best combo. Combos should be returned arranged best to worst.

**1**

**2**

**3**

**4**

**Note:** You can have an empty subset as a combo as well. The popularity for such a subset is 0.

**Example**

$n = 3$   
 $popularity = [3, 5, -2]$   
 $k = 3$

All possible popularities of combos are 0, 3, 5, -2, 8, 3, 1, 6. The best three combos have popularities 8, 6, and 5. The answer is [8, 6, 5].

**Function Description**

int[]: the number of best combos to return

**Returns**

**Int/k:** the popularities of the best  $k$  combos, in decreasing order

**Constraints**

- $1 \leq n \leq 10^5$
- $-10^9 \leq popularity[i] \leq 10^9$
- $1 \leq k \leq \min(2000, 2^n)$

**Input Format For Custom Testing**

**Sample Case 0**

**Sample Input For Custom Testing**

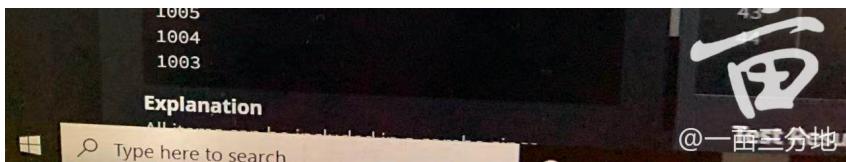
STDIN	Function
4	<code>popularity[] size n = 4</code>
1	<code>popularity = [1, 2, 3, 1000]</code>
2	
3	
1000	
4	<code>k = 4</code>

**Sample Output**

1006  
1005

Language

14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44



```
int main() {
    //string str="aabcdea";
    //string str="alabama";
    vector<int> popularity={3,5,-2};
    int k=3;
    long maxPopularity=0; //add all positive popularity
    for(auto& p:popularity){
        if(p>0)
            maxPopularity += p;
        else
            p=abs(p); //convert negative to positive
    }
    sort(popularity.begin(),popularity.end());
    priority_queue<pair<int,int>,vector<pair<int,int>>,greater<pair<int,int>>>
    pq;
    //min heap storing the sum of a subarray, top is the minimum;
    //for each pair, the 1st element is the sum of a subarry, the 2nd is the
    index of the last element in current subarry
    int n=popularity.size();
    pq.push(pair<int,int>(popularity[0],0));
    vector<long> subarraySum;
    while(!pq.empty() && subarraySum.size()<k-1) {
        auto tmp = pq.top();
        pq.pop();
        int currentMinSum=tmp.first;
        int currentEndIndex=tmp.second;
        subarraySum.push_back(currentMinSum);
        if(currentEndIndex+1<n) { //add the subarry end with next element
            pq.push(pair<int,int>(currentMinSum-
popularity[currentEndIndex]+popularity[currentEndIndex+1],currentEndIndex+1));
            //next
            subarray including current element
            pq.push(pair<int,int>(currentMinSum-
popularity[currentEndIndex]+popularity[currentEndIndex+1],currentEndIndex+1));
        }
        //next subarray without current element
    }
    vector<long> result;
    result.push_back(maxPopularity);
    for(int i:subarraySum)
        result.push_back(maxPopularity-i);
    for(auto i:result)
        cout<<i<<" ";
    return 0;
}
```

## 4 cell 问题

<https://www.1point3acres.com/bbs/thread-918351-1-1.html>

Amazon Web Services (AWS) has several processors for executing processes scheduled on its servers. There are n processes to be executed, where the ith process takes execution [i] amount of time to execute.

Two processes are cohesive if and only if their original execution times are equal.

When a process with execution time and simultaneously reduces the execution time of all its cohesive processes to  $\text{ceil}(\text{execution}[i]) / 2$ .

Given the execution time of n processes, find the total amount of time the processor takes to execute all the processes if you execute the processes in the given order, i.e. from left to right.

#### Notes

The `ceil()` function returns the smallest integer that is bigger or equal to its argument. For example,  $\text{ceil}(1.1) = 2$ ,  $\text{ceil}(2.5) = 3$ ,  $\text{ceil}(5) = 5$ , etc.

If the execution time of some process is reduced and becomes equal to the execution time of any other process ;, then the two processes /and are not considered cohesive.

#### Example

The number of processes is n=6, and their execution times are `execution = [5, 5, 3, 6, 5, 3]`.

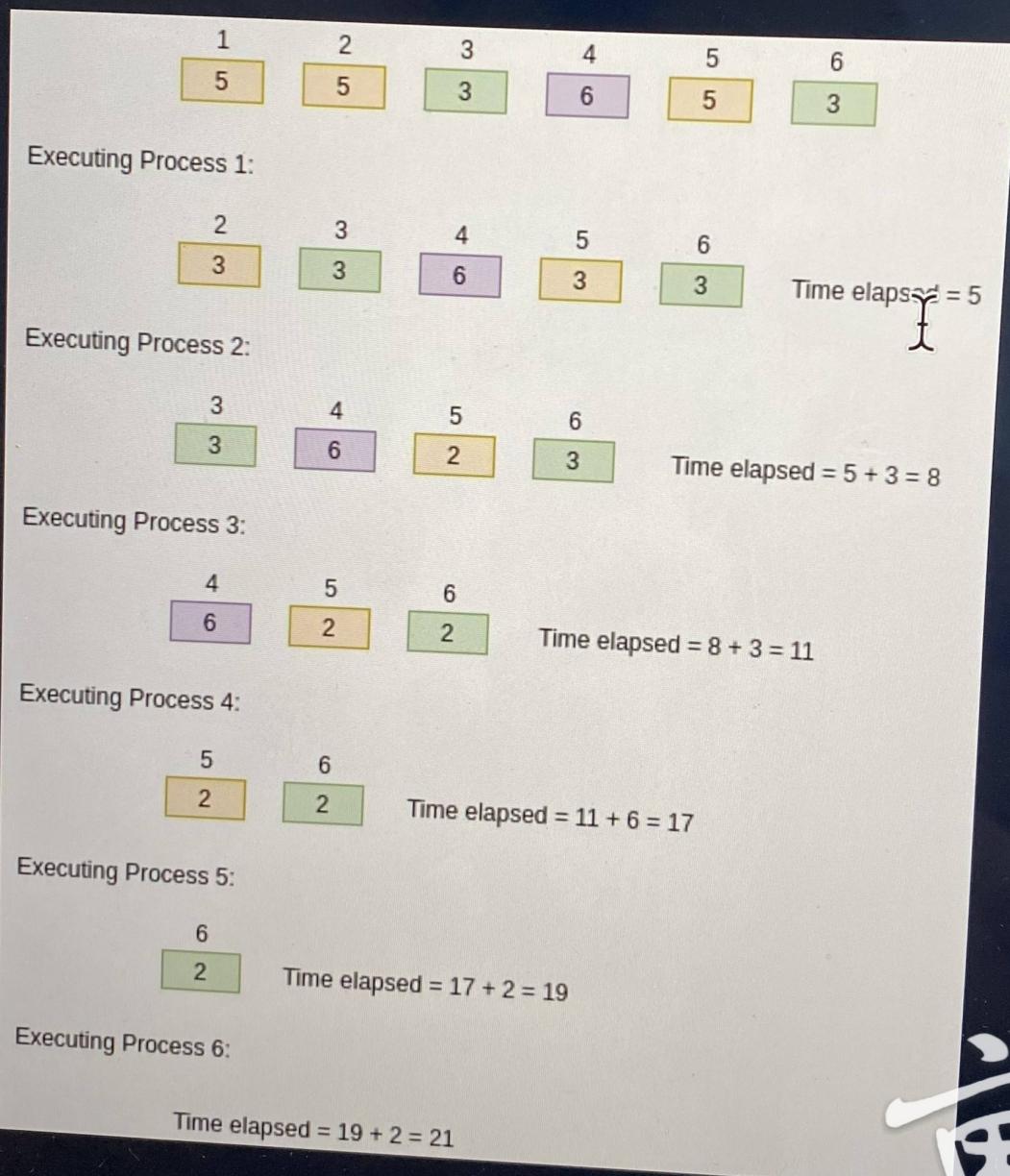
Their execution can be visualized as follows where each set of cohesive processes is marked with a different color.

Note that in the second row, all of the execution times that were initially 5 were reduced to  $\text{ceil}(5/2)= 3$  because they matched the amount of time to execute the first process. They are cohesive. Even though their values now equal those at original indices 3 and 6, they did not match originally. They still are not cohesive. The answer is 21.

## Example

The number of processes is  $n = 6$ , and their execution times are  $\text{execution} = [5, 5, 3, 6, 5, 3]$ .

Their execution can be visualized as follows where *each set of cohesive processes* is marked with a different color.



Note that in the second row, all of the execution times that were initially 5 have been reduced to 3.

@一亩三分地

Hash map<int,int>, key是值, value是当前的值。遍历数组, 每当key再出现一次, 先加一次当前value, 然后把当前value变为ceil(key)/2。

```
int main() {
    int res = 0;
    vector<int> execution={5,5,3,6,5,3};
    unordered_map<int,int> m;
    for(auto i:execution){
        if(m.find(i)==m.end())
            m[i]=i;
        else
            m[i] = m[i]/2 + (m[i]/2 *2 !=m [i]);
        res += m[i];
    }
}
```

```

    }
    cout<<res;
    return res;
}
//21
//{4,3,3,3}: 4+3+2+1=10
//{5, 8, 4, 4 ,8, 2} :5+8+4+2+4+2=25

```

## 5. 求Minimum Total Lag

给两个List center和destination, 求Minimum Total Lag。

分别用两个Priority Queue存储position, total += Math.abs(destination - center)即可.

## 6. 找出后一个数是前一个数的平方数的序列

<https://www.1point3acres.com/bbs/thread-920909-1-1.html>

2, 无序的整型数组, 求出平方数序列的最大长度, 比如对于[2, 8, 9, 16, 4, 3]应该输出3, 对应的平方数序列是2, 4, 16。

2是起始值, 这样的序列对应长度是3, 如果序列是2, 3, 5这样的就是长度0, 找出符合条件的序列的最大长度

## 7. findLongestList

BETA Can't read the text? [Switch theme](#)

**2. Code Question 2**

Amazon Web Services (AWS) has several processors for executing processes scheduled on its servers.

In order to maintain logical independence, a process is divided into segments. Each segment has two characteristics: the segment size ( $1 \leq \text{segment size} \leq 10^9$ ), and a pointer to the next segment so that the sequential order of execution is maintained within a process. Hence, this structure can be visualized as a linked list.

Given the segment structure of a process as a linked list, find the longest sub-list which has the segment sizes in non-increasing order. A sub-list of length 1 is always a valid sub-list. If there are multiple sub-lists of maximum length, return the sub-list which occurs earliest.

**Note:**

1. A sub-list is obtained by removing some nodes from the head and some nodes from the tail of the linked list.
2. Solve the problem in constant extra space

**Example**

There are  $n = 5$  segments with their segment sizes [2, 5, 4, 4, 5].

The longest non-increasing sub-list has 3 nodes:

A reference to the head of the 3 node singly-linked list is returned.

**Function Description**

Complete the function `findLongestList` in the editor below.

`findLongestList` has the following parameter:

`SinglyLinkedListNode* head`: reference to the start of a segment structure of processes

**Returns**

`SinglyLinkedListNode*`: reference to the segment

**Constraints**

- $1 \leq n \leq 10^5$
- $1 \leq \text{SinglyLinkedListNode.data} \leq 10^9$

**Input Format For Custom Testing**

**Sample Case 0**

**Sample Input For Custom Testing**

STDIN	FUNCTION
4 3 2 1 4	→ number of segments $n = 4$ → segment sizes = [3, 2, 1, 4]

**Sample Output**

3	2	1
---	---	---

**Explanation**

The following sub-lists are non-increasing:

Sub-list	Length
3 → 2 → 1	3
2 → 1	2
3 → 2	2
3	1
2	1
1	1
4	1

The answer is the sub-list: 3 → 2 → 1

sliding window

## 8. Common Prefix

1h 6m left

## 2. Code Question 2



ALL



1

2

Kindle Direct Publishing, Amazon's e-book self-publishing platform, is working on a new feature to help authors track the use of text characters in different ways. They have asked for your help in beta testing a new part of the feature involving text prefixes and suffixes. Given a string, split the string into two substrings at every possible point. The rightmost substring is a suffix. The beginning of the string is the prefix. Determine the lengths of the common prefix between each suffix and the original string. Sum and return the lengths of the common prefixes. Return an array where each element  $i$  is the sum for string  $i$ .

### Example

Consider the only string in the array  $\text{inputs} = [\text{abcd}]$ . Each suffix is compared to the original string.

Remove to leave suffix	Suffix	Common Prefix	Length
	'abcabc d'	'abcabc d'	7

string  $i$ .

### Example

Consider the only string in the array  $\text{inputs} = [\text{abcd}]$ . Each suffix is compared to the original string.

Remove to leave suffix	Suffix	Common Prefix	Length
"	'abcabc d'	'abcabc d'	7
'a'	'bcabcd'	"	0
'ab'	'cabcd'	"	0
'abc'	'abcd'	'abc'	3
'abca'	'bcd'	"	0
'abcab'	'cd'	"	0
'abcabc'	'd'	"	0

```
int main() {
    string s = "abcabcd";
    vector<int> res;
    res.push_back(s.length());
    for(int i=0;i<s.length()-1;i++){
        int currentL=0;
        for(int j=i+1;j<s.length();j++){
            if(s[j-(i+1)]==s[j])
                currentL++;
            else
                break;
        }
        res.push_back(currentL);
    }
    for(auto l:res)
        cout<<l<<",";
    return 0;
}
```

## 9. 学生排队 (findimbalance) LC 2104

给size为n的数组。数组里的数为n的permutation，比如[4, 1, 3, 2]。假设有个子数组s，定义该子数组imbalance计算方法子数组排序后 $s[i]-s[i-1] > 1$ 的总个数。要求输出所有子数组imbalance的总个数

比如[4, 1, 3, 2]

子数组 4 => 排序后4, imbalance = 0

子数组 1 => 排序后1, imbalance = 0

子数组 3 => 排序后3, imbalance = 0

子数组 2 => 排序后2, imbalance = 0  
 子数组 4,1 => 排序后1,4, imbalance = 1 因为 4-1=3>1  
 子数组 1,3 => 排序后1,3, imbalance = 1 因为 3-1=2>1  
 子数组 3,2 => 排序后2,3, imbalance = 0 因为 3-2=1=1  
 子数组 4,1,3 => 排序后1,3,4, imbalance = 1 因为 3-1=2>1,4-3=1=1  
 子数组 1,3,2 => 排序后1,2,3, imbalance = 0 因为 2-1=1=1,3-2=1=1  
 子数组 4,1,3,2 => 排序后1,2,3,4, imbalance = 0 因为 2-1=1=1,3-2=1=1,4-3=1=1  
 最后输出结果为3

O(N) monotonic-stack

```

public int imbalanceCount(int[] nums) {
    int n = nums.length;
    int[] indexArr = new int[n + 1];
    for (int i = 0; i < n; i++) {
        indexArr[nums[i]] = i;
    }
    int[] left = new int[n], right = new int[n];
    int[] maxStack = new int[n]; // monotonic-stack
    int index = -1;
    for (int i = 0; i < n; i++) {
        while (index >= 0 && nums[maxStack[index]] < nums[i]) {
            index--;
        }
        left[i] = index < 0 ? -1 : maxStack[index];
        maxStack[++index] = i;
    }
    index = -1;
    for (int i = n - 1; i >= 0; i--) {
        while (index >= 0 && nums[maxStack[index]] < nums[i]) {
            index--;
        }
        right[i] = index < 0 ? n : maxStack[index];
        maxStack[++index] = i;
    }
    int count = 0;
    for (int i = 0; i < n; i++) {
        if (nums[i] < n - 1) {
            int addOneIdx = indexArr[nums[i] + 1];
            if (addOneIdx > i) {
                count += (i + 1) * (addOneIdx - right[i]);
                if (left[i] >= 0) {
                    count += (left[i] + 1) * (right[i] - i);
                }
            } else {
                count += (left[i] - addOneIdx) * (n - i);
                if (right[i] < n) {
                    count += (i - left[i]) * (n - right[i]);
                }
            }
        }
    }
    return count;
}

```

$O(n^2)$ :

```
# 学生排队
def find_imbalances(rank):
    count = 0
    s = set()
    for i in range(len(rank)):
        s.clear()
        max = rank[i]
        min = rank[i]
        s.add(rank[i])
        for j in range(i+1, len(rank)):
            if rank[j] - 1 in s and rank[j] + 1 in s:
                pass
            elif min > rank[j] and rank[j] + 1 in s:
                min = rank[j]
            elif max < rank[j] and rank[j] - 1 in s:
                max = rank[j]
            elif min > rank[j]:
                min = rank[j]
                count += 1
            elif max < rank[j]:
                max = rank[j]
                count += 1
            elif rank[j] - 1 in s or rank[j] + 1 in s:
                count += 1
            else:
                count += 2
        s.add(rank[j])
        print(count, s)
    return count
```

## 10. 分割成两个字符串使得两个substr都有的 distinct char 的数 > k, 有多少种方式

BETA Can't read the text? [Switch theme](#)

## 2. Code Question 2

Amazon Prime Day is a day where many items are put on sale for Amazon Prime members. A list of sale items is assembled where each item is assigned a category denoted by a lowercase English letter.

Since the sale is to be held on two different days, the company has decided to split the list of items into two contiguous non-empty sub-lists - a prefix and a suffix. To ensure that both the days share a sufficient number of similar items, they also need to split it in a way such that the number of distinct categories shared by both the sub-lists is greater than  $k$ .

Formally, given a string, *categories*, find the number of ways to split the string into exactly two contiguous non-empty substrings such that the number of distinct characters occurring in both the substrings is greater than a given integer  $k$ .

### Example

Consider *categories* = "abbcac" and  $k = 1$ . We can split the list of categories into exactly two sub-lists(substrings) in the following ways:

Prefix	Suffix	No of shared categories	List of shared categories
a	bbcac	1	a
ab	bcac	2	a, b
abb	cac	1	a
abbc	ac	2	a, c
abbca	c	1	c

split the list of categories into exactly two sub-lists(substrings) in the following ways:

Prefix	Suffix	No of shared categories	List of shared categories
a	bbcac	1	a
ab	bcac	2	a, b
abb	cac	1	a
abbc	ac	2	a, c
abbca	c	1	c

There are two ways to split the list of categories such that the number of distinct characters shared by both the substrings is greater than 1 so return 2.

### Function Description

Complete the function *findNumWaysToSplit* in the editor below.

*findNumWaysToSplit* has the following parameters:

*string categories*: the categories

*int k*: shared distinct categories must be greater than this value

### Returns

*int*: the number of ways to split the given string

than this value

### Returns

*int*: the number of ways to split the given string

### Constraints

- $1 \leq \text{length}(\text{categories}) \leq 10^5$
- $0 \leq k \leq 26$
- The string *categories* consists of lowercase English characters.

### ► Input Format For Custom Testing

### ▼ Sample Case 0

#### Sample Input For Custom Testing

STDIN	FUNCTION
-----	-----
adbccdbada	→ categories
2	→ k

#### Sample Output

4

### Explanation

The number of ways to split is 4. Assuming 0-based indexing, we can split at -

- Index 2: since the characters {'a', 'd', 'b'} are present in both prefix and suffix, and count(>= 3) > k ( $= 2$ ).

```
bool valid(unordered_map<char, vector<int>> &m, int k){  
    int n=0;  
    for(auto tmp:m)  
        if(tmp.second[0]>0 && tmp.second[1]>0)  
            n++;  
    return n>k;  
}
```

```

int main(){
    int res = 0;
    string categories="abbcac";
    int k=1;
    unordered_map<char,vector<int>> m;
    //key:char value: vector[0] is the freq in left substr, vector[1] is the freq
    in 2nd substr
    for(auto c:categories){
        if(m.find(c)==m.end()){
            m[c].push_back(0);
            m[c].push_back(1);
        }
        else
            m[c][1]++;
    }
    for(int i=0;i<categories.size();i++){
        m[categories[i]][0]++;
        m[categories[i]][1]--;
        if(m[categories[i]][1]==0)
            m.erase(categories[i]);
        res += valid(m,k);
    }
    cout<<res;
    return 0;
}
// 2
/*
    string categories="abdccdbada";
    int k=2;
    return:4
*/

```

## 11. number of unique average values among the combined experience

先排序再求第一个和最后一个平均值存在set里，最后看set的size

1h 9m left



## 1. Code Question 1

ALL



There are  $n$  developers working at Amazon where the  $i^{th}$  developer has the experience points  $\text{experience}[i]$ . The company decided to pair the developers by iteratively pairing the developers with the highest and lowest remaining experience points for a hackathon. The *combined experience* of a pair is the average of the experience points of the two developers. Find the number of unique values among the combined experience of the pairs formed.

1

### Example

$\text{experience} = [1, 4, 1, 3, 5, 6]$

2

There are  $n = 6$  developers. The pairs formed are (1, 6), (1, 5), and (4, 3) making their experience points 3.5, 3, and 3.5 respectively. There are 2 distinct values, 3 and 3.5, so return 2 as the answer.

### Function Description

Complete the function `findUniqueValues` in the editor below.



@一亩三分地

`findUniqueValues` has the following parameter:

`int experience[n]:` the experience points for each developer

1h 9m left



`int experience[n]:` the experience points for each developer

### Returns

`int:` the number of unique values among the combined experience points of the pairs formed

ALL

### Constraints



- $2 \leq n \leq 10^5$
- $n$  is an even number
- $1 \leq \text{experience}[i] \leq 10^9$

1

2

### ► Input Format For Custom Testing

### ▼ Sample Case 0

#### Sample Input For Custom Testing

STDIN	FUNCTION
-----	-----
6	→ number of developers $n = 6$
1	→ $\text{experience} = [1, 1, 1, 1, 1, 1]$
1	
1	
1	



@一亩三分地

# 12. 换password，对于新的password做a->b, b->c, c->d, z->a的处理，然后看新的password是不是旧的substr

1h 8m left

## 2. Code Question 2

ALL

Amazon would like to enforce a password policy that when a user changes their password, the new password cannot be similar to the current one. To determine whether two passwords are similar, they take the new password, choose a set of indices and change the characters at these indices to the next cyclic character exactly once. Character 'a' is changed to 'b', 'b' to 'c' and so on, and 'z' changes to 'a'. The password is said to be *similar* if after applying the operation, the old password is a subsequence of the new password.

2

The developers come up with a set of  $n$  password change requests, where *newPasswords* denotes the array of new passwords and *oldPasswords* denotes the array of old passwords. For each pair *newPasswords*[*i*] and *oldPasswords*[*i*], return "YES" if the passwords are similar, that is, *newPasswords*[*i*] becomes a subsequence of *oldPasswords*[*i*] after performing the operations, and "NO" otherwise.

**Note:** A subsequence is a sequence that can be derived from the given sequence by deleting zero or more elements without changing the order of the remaining elements.

Example

The two lists of passwords are given as *newPasswords* = ["baacbab", "accdb", "baacba"], and *oldPasswords* = ["abdbc", "ach", "abb"].

ALL

Consider the first pair: *newPasswords*[0] = "baacbab" and *oldPasswords* = "abdbc". Change "ac" to "bd" at the 3<sup>rd</sup> and 4<sup>th</sup> positions, and "b" to "c" at the last position.

Diagram illustrating the transformation of the New Password and Old Password:

	New Password						Old Password					
1	b	a	a	c	b	a	b	a	b	d	b	c
	Purple coloured indices changed to next character						Orange coloured indices forms the subsequence					
2	b	a	b	d	b	a	c	a	b	d	b	c

The answer for this pair is YES.

The *newPasswords*[1] = "accdb" and *oldPasswords* = "ach". It is not possible to change the character of the new password to "h" which occurs in the old password, so there is no subsequence that matches. The answer for this pair is NO.

@一亩三分地

```

char transform(char c){
    if(c=='z')
        return 'a';
    return c+1;
}
bool isSubstr(string newStr,string oldstr){
    if(oldstr.length()>newStr.length())
        return false;
    int pold=0;
    int pnew=0;
    while(pnew<newStr.size() && pold<oldstr.size()){
        if(newStr[pnew]==oldstr[pold] || transform(newStr[pnew])==oldstr[pold])
            pold++;
        pnew++;
    }
    return pold==oldstr.size();
}
int main(){
    vector<string> newPassword={"baacbab","accdb","baacba"};
    vector<string> oldPassword={"abdbc","ach","abb"};
    vector<bool> res;
    for(int i=0;i<newPassword.size();i++)
        res.push_back(isSubstr(newPassword[i],oldPassword[i]));
    for(auto i:res)
        cout<<i<<endl;
    return 0;
}

```

## 13. lexicographically smallest palindrome

排序 O (NlogN)

```

int l=s.length();
string res=s.substr(0,l/2);
sort(res.begin(),res.end());
int ls=l/2;
if(l%2==1) //even
    res += s[l/2];
for(int i=ls-1;i>=0;i--)
    res.push_back(res[i]);

```

Hashmap (O(N))

```

int main(){
    string password="cbatabc";
    vector<int> v(26,0); //ind: 1-26 means 'a'-'z', record counts
    for(auto c:password)
        v[c-'a']++;
    //build first half string
    char mid;

```

```

string res;
for(int i=0;i<26;i++){
    string tmp(v[i]/2,'a'+i);
    res += tmp;
    if(v[i]%2==1)
        mid = 'a' + i;
}
//build next half
int l=res.length();
if(password.length()%2==1)
    res += mid;
for(int i=l-1;i>=0;i--)
    res += res[i];
cout<<res;
return 0;
}

```

## 14 Maximize the number of AZ

**Code Question 2**

Since "Amazon" is often shortened to "AZ" while texting, as part of an experiment, Amazon is keen on knowing how many "AZ" subsequences there are in a word in their product reviews. Find the maximum possible count of subsequences after making the given operation.

More formally, given a string consisting of uppercase English letters, at most one character can be added anywhere in the string. Add at most 1 uppercase character to a string to maximize its number of "AZ" subsequences.

**Note:** A subsequence of a string is created by deleting zero or more characters while maintaining the original order. For example, "AZ" is a subsequence of "SAKZ" but not a subsequence of "ZKA".

**Example**  
`s = "AKZ"`  
One choice is to add an 'A' just after 'K' to make "AKA2". The number of subsequences "AZ" in this is 2, which is the most possible.

**Function Description**

maximizeAZ has the following parameters:

- string s: the original string

**Returns**

- int: the maximum number of "AZ" subsequences after adding at most one character

**Constraints**

- 1 ≤ length of s ≤ 10<sup>5</sup>
- s will only consist of uppercase English letters

**Input Format For Custom Testing**

**Sample Case 0**

**Sample Input For Custom Testing**

STDIN	FUNCTION
A	→ s = "A"

**Sample Output**

1

**Explanation**

Add 'Z' to the end: "AZ" has one "AZ" subsequence, the maximum number possible.

**Sample Case 1**

Test Results

先遍历一边，碰到一个z就看他前面的a的数量，然后总数加上当前a的数量。遍历结束后看a多还是z多，总数再加上两者较大的就得到答案了。

```

int solution(string s){
    int nA=0;
    int nZ=0;
    int res=0;
    for(int i=0;i<s.length();i++){
        if(s[i]=='A')

```

```

nA++;
if(s[i]=='Z'){
    res += nA;
    nZ++;
}
res += max(nA,nZ);
return res;
}

```

## 15. minimum keyboard click (9宫格)

1. Code Question 1

A recently launched supplemental typing keypad gained significant popularity on Amazon Shopping due to its flexibility.

This keypad can be connected to any electronic device and has 9 buttons, where each button can have up to 3 lowercase English letters. The buyer has the freedom to choose which letters to place on a button while ensuring that the arrangement is valid. A keypad design is said to be valid if:

1. All 26 letters of the English alphabet exist on the keypad.
2. Each letter is mapped to exactly one button.
3. A button has at most 3 letters mapped to it.

Examples of some valid keypad designs are:

1 abc	2 def	3 ghi
4 jkl	5 mno	6 pqr
7 stu	8 vwx	9 yz

1 ajs	2 bot	3 cpu
4 dkv	5 hmz	6 gl
7 enw	8 fqx	9 iry

- In the left keypad, "hello" can be typed using the following button presses: [3] twice (prints 'h'), [2] twice (prints 'e'), [4] thrice (prints 'l'), [4] thrice (prints 'l'), [5] thrice (prints 'o'). Thus, total number of button presses =  $2 + 2 + 3 + 3 + 3 = 13$ .
- In the right keypad, "hello" can be typed using the following button presses: [5] once (prints 'h'), [7] once (prints 'e'), [6] twice (prints 'l'), [6] twice (prints 'l'), [2] twice (prints 'o'). Thus, total number of button presses =  $1 + 1 + 2 + 2 + 2 = 8$ .

The keypad click count is defined as the number of button presses required to print a given string. In order to send messages faster, customers tend to set the keypad design in such a way that the keypad click count is minimized while maintaining its validity.

```

int minimumKeypadClickCount(string text){
    unordered_map<char, int> m;
    for(int i=0;i<text.length();i++){
        m[text[i]]++;
    }
    vector<pair<char, int>> freq(m.begin(), m.end());
    sort(freq.begin(), freq.end(), [] (const auto &a, const auto &b){return
}

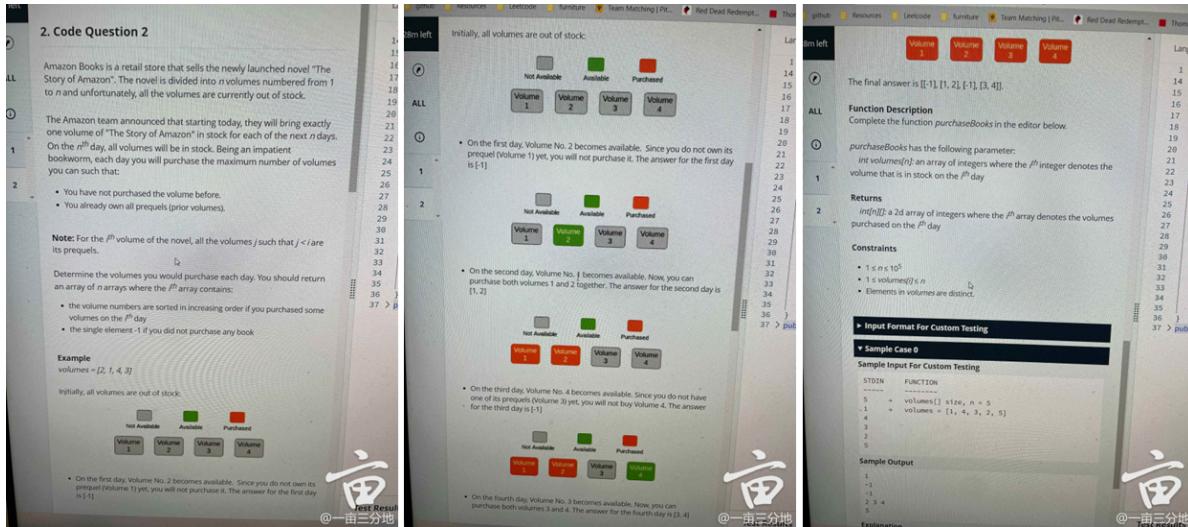
```

```

a.second>b.second;}); //对字符按出现的次数进行排序
vector<int> arr={1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3};
int count = 0;
for (int i=0;i<freq.size();i++){
    count+=freq[i].second*arr[i];
}
return count;
}

```

## 16. Amazon books retail



```

vector<vector<int>> bookRetail(vector<int> volumes){
    int n=volumes.size();
    //priority_queue store current books on sale
    priority_queue<int> pq;
    vector<vector<int>> result;
    int nextbook=1;
    for(int i=0;i<volumes.size();i++){
        pq.push(volumes[i]);
        vector<int> tmp;
        if(pq.size()==i+1 && pq.top()==i+1){
            for(int j=nextbook;j<=i+1;j++)
                tmp.push_back(j);
            nextbook = i+2;
            result.push_back(tmp);
        }
        else
            result.push_back(vector<int>(1, -1));
    }
}

```

## 17. Amazon geeks (linkedlist shopping cart)

**1. Code Question 1**

Amazon has launched a new website "Amazon Geeks" where visitors can shop for all things geek-related. The shopping cart on this website is a bit unusual. The cart items are represented in the form of a linked list where each node instance, a *SinglyLinkedListNode*, has a value, *data* (the item name), and a pointer to the next node, *next*.

You visited the website and initially, your cart had *n* items. You perform some operations with the cart. Each operation consists of two strings: an action and an item name.

There are three types of operations:

- PUSH\_HEAD *itemName***: Insert a new item with *data* as *itemName* as the head of the linked list
- PUSH\_TAIL *itemName***: Insert a new item with *data* as *itemName* as the tail of the linked list
- POP\_HEAD** :- Delete the current head of the linked list. The dash at the right is a filler value since there is no *itemName*.

Return a reference to the head of the final linked list after applying all the queries.

Note: You are not allowed to use extra memory other than to create new nodes for queries **PUSH\_HEAD** and **PUSH\_TAIL**.

**Example**  
Your initial shopping cart is as follows:

In the first operation, you perform **PUSH\_TAIL fan**. The new shopping cart is:

Next, **PUSH\_HEAD jam**.

Then **POP\_HEAD** -.

Finally **POP\_HEAD** -. The final shopping cart and the final list to return is:

**Function Description**  
Complete the function `getShoppingCart` in the editor below.

**getShoppingCart** has the following parameters:

- head**: reference to the head of a linked list of *n* strings
- operations[q][2]**: each row represents an operation to perform

**Returns**  
reference to the head of a linked list

**Constraints**

- $1 \leq n \leq 10^5$
- $1 \leq q \leq 10^5$
- $1 \leq |data|, |itemName| \leq 10$  (where  $|s|$  denotes the length of string  $s$ )
- The *initial basket items and each itemName* consist only of lowercase English letters.
- The list is guaranteed to be non-empty before performing a **POP\_HEAD**.
- The final list is non-empty.

<https://www.1point3acres.com/bbs/thread-919328-1-1.html>

```
#购物网站 cart 链表
def sol(head, operations):
    tail = head
    while tail.next:
        tail = tail.next
    for op in operations:
        common, item = op[0], op[1]
        if common == 'PUSH_TAIL':
            nxt = ListNode(item.val)
            if tail:
                tail.next = nxt
                tail = nxt
            if not head:
                head = tail
        elif common == 'PUSH_HEAD':
            nxt = ListNode(item.val)
            nxt.next = head
            head = nxt
        elif common == 'POP_HEAD':
            if head:
                head = head.next
            else:
                head = None
                tail = None
    return head
```

```
public static ListNode getshoppingcart(ListNode head, List<List<String>> operations){  
    ListNode dummy = new ListNode(val: null,head);  
    ListNode tail = dummy.next;  
    while (tail.next!=null){  
        tail = tail.next;  
    }  
    for (int i=0;i< operations.size();i++){  
        if(dummy.next == null){  
            tail=dummy;  
        }  
        if (operations.get(i).get(0) == "PUSH_HEAD") {  
            dummy.next = new ListNode(operations.get(i).get(1), dummy.next);  
        } else if (operations.get(i).get(0) == "PUSH_TAIL") {  
            ListNode node = new ListNode(operations.get(i).get(1), next: null);  
            tail.next = node;  
            tail = node;  
        } else if (operations.get(i).get(0) == "POP_HEAD") {  
            assert dummy.next != null;  
            dummy.next = dummy.next.next;  
        }  
    }  
    return dummy.next;  
}
```



## 18. 服务器移动

## 1. Code Question 1

Amazon stores its data on different servers at different locations. From time to time, due to several factors, Amazon needs to move its data from one location to another. This challenge involved keeping track of the locations of Amazon's data and report them at the end of the year.

At the start of the year, Amazon's data was located at  $n$  different locations. Over the course of the year, Amazon's data was moved from one server to another  $m$  times. Precisely, in the  $i^{th}$  operation, the data was moved from  $moveFrom[i]$  to  $moveTo[i]$ . Find the locations of the data after all  $m$  moving operations. Return the locations in ascending order.

**Note:** It is guaranteed that for any movement of data :

- There is data at  $moveFrom[i]$ .
- There is no data at  $moveTo[i]$ .

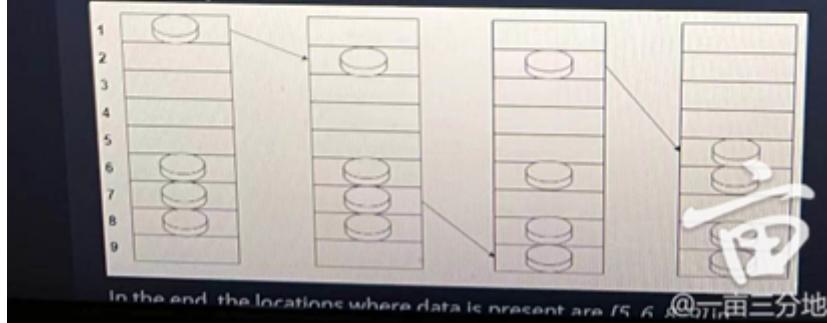
### Example

$locations = [1, 7, 6, 8]$

$moveFrom = [1, 7, 2]$

$moveTo = [2, 9, 5]$

Data begins at locations listed in *locations*. Over the course of the year, the data was moved three times. Data was first moved from  $moveFrom[0]$  to  $moveTo[0]$ , from 1 to 2. Next it moved from 7 to 9, and finally from location 2 to 5.



```
vector<int> moveData(vector<int> data, vector<int> moveFrom, vector<int> moveTo){  
    vector<int> data = {1,7,6,8};  
    vector<int> moveFrom={1,7,2};  
    vector<int> moveTo = {2,9,5};  
    set<int> s(data.begin(),data.end());  
    for(int i=0;i<moveFrom.size();i++){  
        s.erase(moveFrom[i]);  
        s.insert(moveTo[i]);  
    }  
    vector<int> res(s.begin(),s.end());  
    for(auto tmp:res)  
        std::cout << tmp << std::endl;  
    return res;  
}
```

## 19. s里可以拼出多少个t

BETA Can't read the text? [Switch theme](#)

## 2. Code Question 2

Amazon Engineering maintains a large number of logs of operations across all products. A software engineer is debugging an issue in a product. An efficient way to analyze logs is to write automated scripts to check for patterns. The engineer wants to find the maximum number of times a target word can be obtained by rearranging a subset of characters in a log entry.

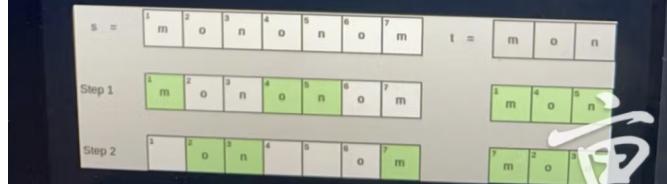
Given a log entry  $s$  and target word  $t$ , the target word can be obtained by selecting some subset of characters from  $s$  that can be rearranged to form string  $t$  and removing them from  $s$ . Determine the maximum number of times the target word can be removed from the given log entry.

**Note:** Both strings  $s$  and  $t$  consist only of lowercase English letters.

### Example

$s = "mononom"$

$t = "mon"$



给两个字符串s和t，求问使用s所有的字母最多能够重组出几个t。举个栗子： s="mononom", t="mon"，则答案是2。思路是用两个hashmap统计s和t的字母出现频率，然后遍历t的所有字母，找到s中出现次数/t中出现次数的最小值（向下取整）

```
int countMaxOperations(string s, string t){  
    unordered_map<char, int> ms;  
    unordered_map<char, int> mt;  
    int result = INT_MAX;  
    for(auto c:s)  
        ms[c]++;  
    for(auto c:t)  
        mt[c]++;  
    for(auto tmp:mt){  
        if(ms.find(tmp.first)==ms.end()) //didn't find this char  
            return 0;  
        else  
            result = min(result,ms[tmp.first]/tmp.second);  
    }  
}
```

## 20. 密码强度

Given a string password, find the strength of that password. The strength of a password, consisting only lowercase english letters only, is calculated as the sum of the number of distinct characters present in all possible substrings of that password.

Example:- password = "good"

possible sub string and count of distinct characters are

```
g = 1
o = 1
o = 1
d = 1
go = 2
oo = 1
od = 2
goo = 2
ood = 2
good = 3
1+1+1+1+2+1+2+2+2+3 = 16
```

Example:- password: "test"

Output: 19

Example:- password: "abc"

Output: 10

```
int passwordStrength
```

```
# 密码强度
def password(s):
    ans = 0
    n = len(s)
    yuan = {"a", "e", "i", "o", "u"}
    isyuan, isfu = False, False
    for c in s:
        if c in yuan:
            isyuan = True
        else:
            isfu = True
        if isyuan and isfu:
            ans += 1
        isyuan, isfu = False, False
    return ans

print(password("thisisbeautiful"))
```

## 21. 最大的 processor cluster

一排机器, input第一个是一个数组, 代表每个机器的processing power, 第二个数组代表boosting power, 第三个整数Max power。要求出满足条件最长的cluster (连续的sub array) , 在这个subarray 里满足: max(boosting power) + sum(processing power) \* k < , k是这个subarray的长度。

```
int findMaximumSustainableClusterSize(vector<int> processingPower, vector<int>
bootingPower, long powerMax){
    int n = processingPower.size();
    int i = 0;
    int j = 0;
```

```

deque<vector<int>> q;
long sum = 0;
int ans = 0;
while(j < n){
    sum += processingPower[j];
    while(!q.empty() && q.back()[0] <= bootingPower[j]){
        q.pop_back();
    }
    q.push_back({bootingPower[j], j});

    while(i <= j and q.front()[0] + sum*(j-i+1) > powerMax){
        if(q.front()[1] == i){
            q.pop_front();
        }
        sum -= processingPower[i];
        i++;
    }
    ans = max(ans, j-i+1);
    j++;
}

return ans;
}

```

## 22. countDecreasingRatings 连续递减的子序列个数 consecutive decreasing array

The figure consists of three side-by-side screenshots from a programming platform. The left screenshot shows a question about Amazon Shopping rating groups. The middle screenshot shows a diagram of a binary tree where nodes represent rating groups, and the right screenshot shows the official problem statement for 'countDecreasingRatings'.

**Left Screenshot (Question):**

- Section 2: Code Question 2**
- Description:** Amazon Shopping recently launched a new item whose daily customer ratings for  $n$  days is represented by the array  $\text{ratings}$ . They monitor these ratings to identify products that are not performing well. Find the number of groups that can be formed consisting of 1 or more consecutive days such that the rating continuously decreases over the days.
- Example:**  $\text{ratings} = [4, 3, 5, 4, 3]$
- Output:** There are 9 periods in which the rating consecutively decreases.
- Results:**
  - 5 one day periods: [4], [3], [5], [4], [3]
  - 3 two day periods: [4, 3], [5, 4], [4, 3]
  - 1 three day period: [5, 4, 3]

**Middle Screenshot (Diagram):**

A binary tree diagram visualizing the rating groups. The root node is 5, which branches into 4 and 3. Node 4 branches into 3 and 4. Node 3 branches into 4 and 3. This represents the hierarchical structure of the consecutive decreasing rating periods shown in the results.

**Right Screenshot (Official Problem Statement):**

- Function:** `countDecreasingRatings` contains one parameter: `int ratings[n]`: customer ratings for  $n$  days
- Returns:** `long`: the number of valid groups of ratings
- Constraints:**
  - $1 \leq n \leq 10^5$
  - $0 \leq \text{ratings}[i] \leq 10^9$
- Input Format For Custom Testing:**

STDIN	FUNCTION
3 2 1 3	<code>→ ratings[] size n = 3</code> <code>→ ratings = [2, 1, 3]</code>
- Sample Case 0:** Input: 3, Output: 4
- Explanation:** There are 4 groups of continuously decreasing ratings: [2], [1], [3], [2, 1].

给一个integer数组arr，判断有多少个subarray是consecutive decreasing，consecutive decreasing是指这些相邻元素，后一个 = 前一个 - 1。

例子：arr = [4, 3, 2, 5, 9, 8]

符合条件的subarray有[4], [4, 3], [4, 3, 2], [3], [3, 2], [2], [5], [9], [9, 8], [8]，所以答案是10。

解法：先把arr分成consecutive decreasing的subarray，记录每个subarray的长度，长度为n的subarray有 $n * (n + 1) / 2$ 种case。

上面的例子，先分成[4, 3, 2], [5], [9, 8]，记录所有的长度[3, 1, 2]，再对每个长度n求 $n * (n + 1) / 2$ ，最后相加。

```

int countDecreasingRatings(vector<int> ratings){
    int left=0;
    int right=1;
    int res=0;
    while(right<ratings.size()){
        if(ratings[right-1]==ratings[right]+1)
            res += right-start;
        else
            left=right;
        right++;
    }
    res += ratings.size(); //add all single-element vectors
}

```

## 23. FindMinHealth (LC 2214)

```

class Solution {
public:
    long long minimumHealth(vector<int>& damage, int armor) {
        int maxDamage=INT_MIN;
        long long totalDamage=0;
        for(int i=0;i<damage.size();i++){
            maxDamage=max(maxDamage,damage[i]);
            totalDamage += damage[i];
        }
        if(maxDamage>armor)
            return totalDamage+1-armor;
        else
            return totalDamage+1-maxDamage;
    }
};

```

## 24. 变种lc719, smallest distance pair; k smallest inefficiencies

## 2. Code Question 2

An Amazon team is being formed to work on a new project. They will choose a pair of developers according to their skills. The inefficiency of a team is defined as the absolute difference between the skills of the two developers. Given the skill values of  $n$  developers, find the  $k$  teams with the lowest inefficiencies among all possible pairs of teams. Return their inefficiencies sorted ascending.

**Note:** Pair  $(i, j)$  is considered the same as pair  $(j, i)$ .

### Example

If  $n = 3$ ,  $\text{skill} = [6, 9, 1]$ ,  $k = 2$

The following pairs of teams are possible:

Team No.	Skill 1	Skill 2	Inefficiency
1	6	9	$ 6 - 9  = 3$
2	6	1	$ 6 - 1  = 5$
3	9	1	$ 9 - 1  = 8$

- The first 2 smallest numbers among  $[3, 5, 8]$  are 3 and 5.
- Return the array  $[3, 5]$ . Note that the array is sorted ascending.

### Function Description

Complete the function `getSmallestInefficiencies` in the editor below.

`getSmallestInefficiencies` has the following parameters:

- `int skill[n]`: the skill values for  $n$  developers
- `int k`: the number of pairs required

### Returns

`int[k]`: the  $k$  smallest inefficiencies in sorted order

### Constraints

- $2 < n < 10^5$



Priority queue

```
class Solution {
public:
    int smallestDistancePair(vector<int>& nums, int k) {
```

```

vector<int> nums = {6,9,1};
int k=2;
sort(nums.begin(),nums.end());
priority_queue<vector<int>,vector<vector<int>>,greater<vector<int>>> pq;
for(int i=0;i<nums.size()-1;i++){
    pq.push({nums[i+1]-nums[i],i,i+1});
}
vector<int> tmp;
vector<int> res;
while(k>0){
    tmp=pq.top();
    pq.pop();
    res.push_back(tmp[0]);
    if(res[2]+1<nums.size()){
        pq.push({nums[res[2]+1]-nums[res[1]],res[1],res[2]+1});
    }
    k--;
}
for(auto c:res)
    cout << c << endl;
return res;
}
};


```

```

vector<int> LowestKIneffcient(vector<int> developers, int k){
    int n = developers.size();
    priority_queue<int> maxH ;
    for(int i = 0 ; i < n-1; i++){
        for(int j = i+1; j < n; j++){
            maxH.push(abs(developers[j] - developers[i]));
            if(maxH.size() > k){
                maxH.pop();
            }
        }
    }

    vector<int> ans ;
    while(!maxH.empty()){
        ans.push_back(maxH.top());
        maxH.pop();
    }
    reverse(ans.begin(), ans.end());
    return ans;
}


```

## 25. pascal triangle (变种LC2221)

## 1. Code Question 1

ALL

In order to ensure maximum security, the developers at Amazon employ multiple encryption methods to keep user data protected.

1

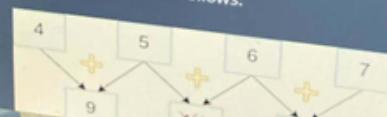
In one method, numbers are encrypted using a scheme called 'Pascal Triangle'. When an array of digits is fed to this system, it sums the adjacent digits. It then takes the rightmost digit (least significant digit) of each addition for the next step. Thus, the number of digits in each step is reduced by 1. This procedure is repeated until there are only 2 digits left, and this sequence of 2 digits forms the encrypted number.

Given the initial sequence of the digits of numbers, find the encrypted number. You should report a string of digits representing the encrypted number.

Example

numbers = [4, 5, 6, 7]

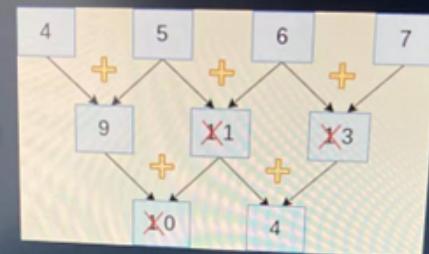
Encryption occurs as follows:



Example

numbers = [4, 5, 6, 7]

Encryption occurs as follows:



Hence, the encrypted number formed is 04.

Function Description

Complete the function `getEncryptedNumber` in the editor below.

`getEncryptedNumber` has the following parameter:

`int numbers[n]:` the initial sequence of digits

Returns

`string:` the encrypted number represented as a string of 2 characters.

Constraints

- $2 \leq \text{numbers.length} \leq 5 \cdot 10^3$
- $0 \leq \text{numbers}[i] \leq 9$



```
vector<int> triangularSum(vector<int>& nums) {
    if(nums.size() == 2)
        return nums;
    for(int i = 0 ; i < nums.size()-1; ++i)
        nums[i] = (nums[i] + nums[i+1])%10;
        nums.pop_back();
    return triangularSum(nums);
}
```

## 26. 缺失数字

给了一个List, 和一个Int k。要你找出满足k - List.size( )个大小的最小缺失数字和。

Example: [1, 3, 5, 7, 10] k = 7;

Process: miss nums: 2, 4, 6, 8, 9;

res: 我们需要  $7 - 5 = 2$  个缺失数字, 要想和最小, 那就从头开始选, 那就是 $2 + 4 = 6$ , 所以最后return 6

```
class Solution {
public:
    long long minimalKSum(vector<int>& nums, int k) {
        nums.push_back(0);
        sort(nums.begin(), nums.end());
        int maxval;
        long long ret = 0;
        for(int i=0; i<nums.size()-1; i++) {
            int tmp = nums[i+1] - nums[i];
            if(tmp > 0) {
                ret -= nums[i];
```

```
tmp--;
if(tmp >= k) {
    maxval = nums[i] + k;
    k = 0;
    break;
}
else
    k -= tmp;
}
}

if(k > 0) {
    ret -= nums.back();
    maxval = nums.back() + k;
}
ret += (((long long)(maxval)+1)*maxval)>>1;
return ret;
}
};
```

## 27. 分割字符串,每个字符串无重复 minimum redundancy-free segments

---

## 1. Code Question 1

ALL

Your team at Amazon is developing a new algorithm for suggesting passwords when a user sets up a new account.

1

A string of lowercase English characters is said to be *redundancy-free* if each character occurs at most once in the string. In order to ensure minimum redundancy, the developers suggest a password that has the minimum number of *redundancy-free* segments it can be divided into.

2

Given a string, *password*, find the minimum number of *redundancy-free* segments we can divide it into.

### Example

The given password = "aabccdea"

aabcdea  
a | abcde | a

The password can be partitioned into a minimum of 3 valid segments. Return 3.

### Function Description

Complete the function *findMinSegments* in the editor below.

*findMinSegments* has the following parameter:  
*string password*: the given password

Information

1

### Function Description

Complete the function *findMinSegments* in the editor below.

ALL

*findMinSegments* has the following parameter:

*string password*: the given password

0

### Returns

*int*: the number of segments in the string to

1

### Constraints

2

- $1 \leq \text{length of } \textit{password} \leq 10^5$
- All characters in *password* are lowercase English letters.

### Input Format For Custom Testing

#### Sample Case 0

##### Sample Input For Custom Testing

STDIN	FUNCTION
-----	-----
alabama	$\rightarrow$ password =
"alabama"	

##### Sample Output

4

##### Explanation

The minimum partitions are "al/", "ab", "a", "ma".

#### Sample Case 1

```
int main() {
//string str="aabccdea";
```

```
//string str="alabama";
string str="aaaaaaa";
int left=0;
int right=0;
int result=1;
unordered_set<char> s;
while(right<str.length()){
    if(s.find(str[right])!=s.end()){
        left=right;
        s.clear();
        result++;
    }
    s.insert(str[right]);
    right++;
}
cout<<result;
return 0;
}

// "aabbcdea": 3
// "alabama": 4
// "aaaaaaa": 6
```

**28 Max stock price** 给定k值，给连续k月并且k个值各不一样的区间求和，找到最大和。

9m left

Language

## 2. Code Question 2



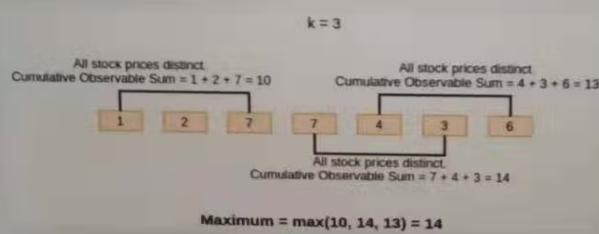
**ALL** The price of Amazon common stock is analyzed over a period of several months.

- 1 A group of  $k$  consecutive months is said to be *observable* if no two months in the group have the same stock price. In other words, all the prices in the period are distinct. The sum of stock prices of an *observable* group of months is called the *cumulative observable sum*. Given the price of Amazon stock for  $n$  months, find the maximum possible *cumulative observable sum* among all the *observable* groups of months. If there is no *observable* group, return -1.

### Example

Monthly stock prices are given as  $stockPrice = [1, 2, 7, 7, 4, 3, 6]$ , and the number of consecutive months to consider is  $k = 3$ .

The following groups of *observable* months can be formed:



Return the maximum of the values, 14.

亩三分地  
@Taobao

```
long maxSumOfStock(vector<int> stockPrice, int k){  
    long result = 0;  
    int left=0;  
    int right=0;  
    unordered_set<int> s;  
    while(right<stockPrice.size()) {  
        while(s.find(stockPrice[right])!=s.end() || right-left+1>k) {  
            s.erase(stockPrice[left++]);  
        }  
        s.insert(stockPrice[right]);  
        if(right-left+1==k){  
            long tmp=0;
```

```
        for(auto i:s)
            tmp += i;
            result = max(result,tmp);
    }
    right++;
}
if(result==0)
    return -1;
return result;
}
```

## 29 SearchWord & resultWord searchword需要添加多少个字符

一个searchWord和一个resultWord，问最少给searchWord末尾添加几个字符，可以让resultWord成为它的一个subsequence。举个栗子： searchWord="armaze", resultWord="amazon", 则应该返回2 (添加on)。思路是两个指针p1, p2分别遍历两个字符串，如果指向的字符相同，则将双指针同时向后移动一位，反之只移动指向searchWord的指针，直到任意指针到达末尾。返回resultString的长度与resultString指针位置的差

## 1. Code Question 1

Amazon Shopping provides a product-search feature that makes browsing products easier. Instead of showing exact matches only, it also displays *transformable* results for better browsing. A word *a* is said to be *transformable* to a word *b* if *a* is a subsequence of *b*. Given *searchWord* and *resultWord*, find the minimum number of characters that must be appended at the end of *searchWord*, such that *resultWord* is a subsequence of the modified *searchWord*.

**Note:** A subsequence of a string is a string that results from deleting 0 or more characters from the string without changing the order of the remaining characters. For example, *amazon* is a subsequence of *abcmmndaqzxopn* while *abc* is not a subsequence of *cdhbqaab*.

### Example

```
searchWord = "armaze"  
resultWord = "amazon"
```



Add 2 characters, 'on', to *searchWord* to make *resultWord*, a subsequence of *searchWord*.

Test

```
int searchwordToResultword(string searchword, string resultword){  
    int ps=0;  
    int pr=0;  
    while(ps<searchword.length() && pr<resultword.length()){  
        if(searchword[ps]==resultword[pr])  
            pr++;  
        ps++;  
    }  
    return resultword.length()-pr;  
}
```

## 30. User System Design 用户登陆API设计

写一个简单的api，有三个功能 register, log in, log out。register的时候要输入name和password，如果这个用户已经register过了要返回username already exists，没有的话返回registered successfully; log in时也要name和password，如果该name并没有register或者已经logged in，或者password错误，要返回log in unsuccessful，如果都满足就返回logged in successfully；最后是log out，也是很直观的逻辑，正常的话返回成功，没有register或者没有log in的name要返回log out失败。这道题不是要你去写class，而是一个函数，通过输入a list of strings然后你自己去parse出以上三个指令。HashMap储存username和password

The screenshot shows a HackerRank challenge titled "31 MaxGreyness 灰度题". On the left, there's a sidebar with a timer (1h 8m left) and a navigation menu (ALL). The main area displays API documentation for three sign-in pages:

	Register	Login	Logout
<b>Function</b>	Registers a new user with the username and password	Verifies the username and password, then grants or denies access	username logs out of the website
<b>API Request</b>	<code>register &lt;username&gt; &lt;password&gt;</code>	<code>login &lt;username&gt; &lt;password&gt;</code>	<code>logout &lt;username&gt;</code>
<b>Returns</b>	<ul style="list-style-type: none"> <li>If the registration was successful, Registered Successfully</li> <li>If the user already exists, Username already exists</li> </ul>	<ul style="list-style-type: none"> <li>If the login was successful, Logged In Successfully</li> <li>If the login was unsuccessful, Login Unsuccessful</li> </ul>	<ul style="list-style-type: none"> <li>If the logout was successful, Logged Out Successfully</li> <li>If the given username wasn't logged in, Logout Unsuccessful</li> </ul>

Below the table, it says: "Given a log of API requests, return the list of returns from the mock website." A "Notes:" section lists:

- Initially, there are no users registered.
- If a user is already logged in and makes a login request, the new request is unsuccessful. The original login remains active.
- Each log is an API request and is in one of the three allowed formats.
- The order of execution of each request is the same as the order of input.

The right side of the screen shows a Java code editor with the following code:

```

Language: Java 8
Autocomplete Ready
1 > import java.io.*;-
12
13
14
15 <class Result {
16
17 <
18     * Complete the 'implementAPI' function below.
19     *
20     * The function is expected to return a STRING.
21     * The function accepts STRING_ARRAY logs as
22     * parameter.
23
24 <public static List<String> implementAPI(List<String>
25 logs) {
26     // Write your code here
27 }
28
29 }
30
31 > public class Solution { -
59

```

## 31 MaxGreyness 灰度题

---

Several satellites provide observational black and white images which are stored in data centers at Amazon Web Services (AWS).

A black and white image is composed of pixels and is represented as an  $(n \cdot m)$  grid of cells. Each pixel can have a value of 0 or 1, where 0 represents a white pixel and 1 represents a black pixel. The *greyness* of a cell  $(i, j)$  is affected by the pixel values in the  $i^{\text{th}}$  row and the  $j^{\text{th}}$  column. More formally, the *greyness* of the cell  $(i, j)$  is the difference between the number of black pixels in the  $i^{\text{th}}$  row and the  $j^{\text{th}}$  column and the number of white pixels in the  $i^{\text{th}}$  row and the  $j^{\text{th}}$  column.

Find the maximum *greyness* among all the cells of the grid.

Note: The value of cell  $(i, j)$  is counted both in the  $i^{\text{th}}$  row and in the  $j^{\text{th}}$  column.

给一个2D array,  $\text{array}[i][j]$ 表示在 $(i, j)$ 点处的灰度值, 0为白1为黑,  $(i, j)$ 处的灰度值为第*i*行所有点灰度值 + 第*j*列所有点灰度值之和 (黑+1, 白-1), 求array中灰度值最大的点的灰度值为多少, 两层for循环分别求出colsGrey[]和rowsGrey[]然后再遍历一遍求max即可

```

int maxGreyness(vector<string> grid){
    vector<int> sumRow(grid.size(), 0);
    vector<int> sumCol(grid[0].size(), 0);
    for(int i=0;i<grid.size();i++){
        for(int j=0;j<grid[0].size();j++){
            sumRow[i] += 2*(grid[i][j]=='1')-1;
            sumCol[j] += 2*(grid[i][j]=='1')-1;
        }
    }
    int maxRow=INT_MIN;
    int maxCol=INT_MIN;
    for(auto i:sumRow)
        maxRow=max(maxRow,i);
    for(auto i:sumCol)
        maxCol=max(maxCol,i);
    return maxRow+maxCol;
}

```

## 32 ShipmentImbalance (leetcode 2104) 子数组Max/Min求差值，再加和

求array的所有subArray的 max - min的和总和.

O(n^2):

```

long long subArrayRanges(vector<int>& A) {
    long long res = 0;
    for (int i = 0; i < A.size(); i++) {
        int ma = A[i], mi = A[i];
        for (int j = i; j < A.size(); j++) {
            ma = max(ma, A[j]);
            mi = min(mi, A[j]);
            res += ma - mi;
        }
    }
    return res;
}

```

O(n):

```

long long subArrayRanges(vector<int>& A) {
    long res = 0, n = A.size(), j, k;
    stack<int> s;
    for (int i = 0; i <= n; ++i) {
        while (!s.empty() && A[s.top()] > (i == n ? -2e9 : A[i])) {
            j = s.top(), s.pop();
            k = s.empty() ? -1 : s.top();
            res -= (long)A[j] * (i - j) * (j - k);
        }
        s.push(i);
    }
    s = stack<int>();
    for (int i = 0; i <= n; ++i) {
        while (!s.empty() && A[s.top()] < (i == n ? 2e9 : A[i])) {

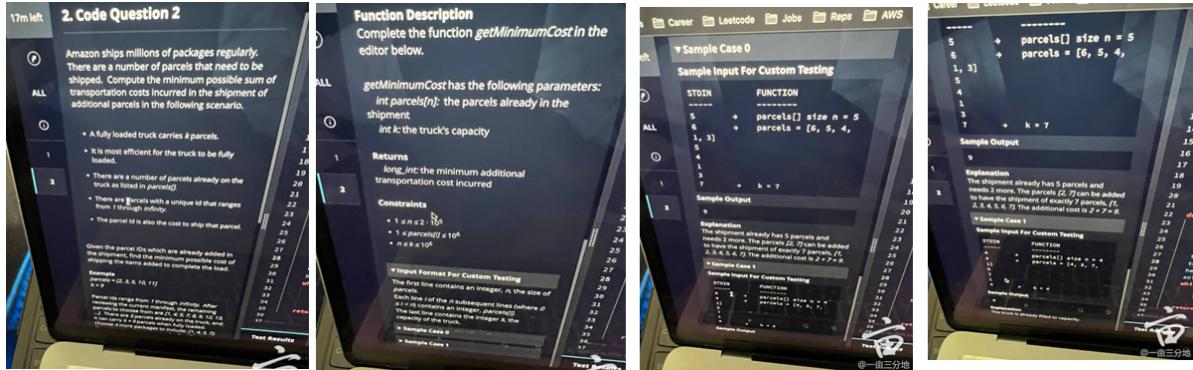
```

```

        j = s.top(), s.pop();
        k = s.empty() ? -1 : s.top();
        res += (long)A[j] * (i - j) * (j - k);
    }
    s.push(i);
}
return res;
}

```

## 33 MinShippingCost 包裹运输(LC2195)



```

class Solution {
public:
    long long minimalKSum(vector<int>& nums, int k) {
        int count=k;
        nums.push_back(0);
        nums.push_back(INT_MAX);
        sort(nums.begin(),nums.end());
        long long res=0;
        int startNum=0;
        for(int num:nums){
            int diff = num-startNum;
            if(diff>1 && count >0){
                long range = min(diff-1,count);
                res += (startNum+1+startNum+range)*range/2;
                count -= range;
            }
            startNum=num;
            if(count<=0)
                break;
        }
        return res;
    }
};

```

## 34. PrimeMovieAward

awards 分组，每组max-min不超过k，就直接sort一遍，然后贪心的把最大能加到当前组的加进来，加不进来的count++，换个新组

3m 42s left BETA Can't read the text? [Switch theme](#)

## 2. Code Question 2

ALL Amazon Prime Video is a subscription video-on-demand over-the-top streaming and rental service. The team at Prime Video is developing a method to divide movies into groups based on the number of awards they have won. A group can consist of any number of movies, but the difference in the number of awards won by any two movies in the group must not exceed  $k$ .

The movies can be grouped together irrespective of their initial order. Determine the minimum number of groups that can be formed such that each movie is in exactly one group.

**Example**  
The numbers of awards per movie are  $\text{awards} = [1, 5, 4, 6, 8, 9, 2]$ , and the maximum allowed difference is  $k = 3$ .

One way to divide the movies into the minimum number of groups is:

- The first group can contain  $[2, 1]$ . The maximum difference between awards of any two movies is 1 which does not exceed  $k$ .
- The second group can contain  $[5, 4, 6]$ . The maximum difference between awards of any two movies is 2 which does not exceed  $k$ .
- The third group can contain  $[8, 9]$ . The maximum difference between awards of any two movies is 1 which does not exceed  $k$ .

The movies can be divided into a minimum of 3 groups.

**Function Description**  
Complete the function `minimumGroups` in the editor below.

`minimumGroups` has the following parameters:

- `int awards[n]`: the number of awards each movie has earned
- `int k`: the maximum difference in awards counts

**Returns**  
`int`: the minimum number of groups into which all the movies can be divided

**Constraints**

- $1 \leq n \leq 10^6$
- $1 \leq k \leq 10^4$
- $1 \leq \text{awards}[i] \leq 10^9$

**Input Format For Custom Testing**

**Sample Case 0**

**Sample Input For Custom Testing**

STDIN	FUNCTION
7 + 1 + 3 6 8 9	awards[] size $n = 7$ awards = [1, 13, 6, 8, 9, 3, 5]

@一亩三分地

```
int minGroups(vector<int> awards, int k){  
    int left=0; int right=0;  
    sort(awards.begin(), awards.end());  
    int res=0;  
    while(right<awards.size()) {  
        if(awards[right]-awards[left]>k) {  
            left=right;  
            res++;  
        }  
        right++;  
    }  
    return res+1;  
}
```

# 35 MaxAggregateTemperatureChange 最大浮动温度

**1. Code Question 1**

Alexa is Amazon's virtual AI assistant. It makes it easy to set up your Alexa-enabled devices, listen to music, get weather updates, and much more. The team is working on a new feature that evaluates the aggregate temperature change for a period based on the changes in temperature of previous and upcoming days.

Taking the change in temperature data of  $n$  days, the aggregate temperature change evaluated on the  $i^{\text{th}}$  day is the maximum of the sum of the changes in temperatures until the  $i^{\text{th}}$  day, and the sum of the change in temperatures in the next  $(n-i)$  days, with the  $i^{\text{th}}$  day temperature change included in both.

Given the temperature data of  $n$  days, find the maximum aggregate temperature change evaluated among all the days.

Example  
`tempChange = [6, -2, 5]`

The aggregate temperature change on each day is evaluated as:

The aggregate temperature change on each day is evaluated as:

For Day 1, there are no preceding days' information, but the day itself is included in the calculation. Temperature changes = [6] for the before period. For succeeding days, there are values [6, -2, 5] and [6 + (-2) = 4]. Again, the change for day 1 is included. The maximum of 6 and 4 is 6.

For Day 2, consider [6, -2] > 6 + (-2) = 4, and [-2, 5] > (-2) + 5 = 3. The maximum of 3 and 4 is 4.

For Day 3, consider [6, -2, 5] > 6 + (-2) + 5 = 9, and [5]. The maximum of 9 and 5 is 9.

The maximum aggregate temperature change is  $\max(9, 4, 9) = 9$ .

**Function Description**  
Complete the function `getMaxAggregateTemperatureChanges` in the editor.

```
long getMaxAggreTempChange(vector<int> changes){  
    int n=changes.size();  
    vector<long> prefixSum(n);  
    prefixSum[0]=changes[0];  
    for(int i=1;i<n;i++){  
        prefixSum[i]=prefixSum[i-1]+changes[i];  
    long res=max(prefixSum[i],prefixSum[n-1]);  
    for(int i=1;i<n;i++){  
        long tmp=max(prefixSum[i],prefixSum[n-1]-prefixSum[i-1]);  
        res=max(res,tmp);  
    }  
    return res;  
}
```

# 36. 股价findEarliest month

## 2. Code Question 2

The interns at Amazon were asked to review the company's stock value over a period. Given the stock prices of  $n$  months, the *net price change* for the  $i^{\text{th}}$  month is defined as the absolute difference between the average of stock prices for the first  $i$  months and for the remaining  $(n - i)$  months where  $1 \leq i < n$ . Note that these averages are rounded down to an integer.

Given an array of stock prices, find the month at which the *net price change* is minimum. If there are several such months, return the earliest month.

Note: The average of a set of integers here is defined as the sum of integers divided by the number of integers, rounded down to the nearest integer. For example, the average of [1, 2, 3, 4] is the floor of  $(1 + 2 + 3 + 4) / 4 = 2.5$  and the floor of 2.5 is 2.

### Example

stockPrice = [1, 3, 2, 3]



The minimum *net price change* is 0, and it occurs in the 2<sup>nd</sup> month. Return 2.

### Function Description

Complete the function `findEarliestMonth` the editor below.

`findEarliestMonth` has the following parameter:

`int stockPrice[n]:` the stock prices

### Returns

`int:` the earliest month in which the *net price change* is minimum

### Constraints

- $2 \leq n \leq 10^5$
- $1 \leq stockPrice[i] \leq 10^9$



prefixSum 记录从开始到现在的总和。

股票stock[]切一刀，切在哪里时左半边Avg和右半边Avg的差值最小，注意数组0-index，返回月份要+1。

```
int earliestMonth(vector<int> stockPrice){  
    int n=stockPrice.size();  
    vector<long> prefix(n);  
    prefix[0] = stockPrice[0];  
    for(int i=1;i<n;i++)  
        prefix[i]=prefix[i-1]+stockPrice[i];  
    long minDiff=INT_MAX;  
    int res=0;  
    for(int i=0;i<n-1;i++){  
        long left = prefix[i]/long(i+1);  
        long right = (prefix[n-1]-prefix[i])/long(n-i-1);  
        long diff=abs(left-right);
```

```

        if(diff<minDiff){
            res=i;
            minDiff=diff;
        }
    }
    if(prefix[n-1]/n<minDiff)
        res = n-1;
    return res;
}

```

## 37. 给一组数组, 删掉k个连续元素, 问剩下最小的和是多少?

**1. Code Question 1**

The virtual private server on Amazon's Web Service cloud is used to run programs that help performing calculations on large data sets. Recently, the performance of some programs has degraded.

Technical support has recommended that processes that consume a lot of main memory should be deleted. Unfortunately, the command-line shell that you use only lets you delete processes that form a single contiguous segment of a given fixed size. The size of a contiguous segment is the number of contiguous processes in the main memory.

Find the minimum amount of main memory used by all of your processes in your instance after you delete a contiguous segment of processes.

**Example**  
`processes = [10, 4, 8, 13, 20]  
m = 2`

The processes in main memory are visualized below:

Select a fixed contiguous segment size of  $m = 2$ . The best single contiguous segment of size 2 to delete is the segment composed of process 5, which is 20KB, and process 4, which is 13KB. This results in the minimum total main memory consumption of  $10KB + 4KB + 8KB = 22KB$ . Return 22.

**Function Description**  
Complete the function `minimizeMemory` in the editor below.

**minimizeMemory** has the following parameters:

- `int processes[n]`: kilobytes occupied by each process
- `int m`: the fixed number of contiguous applications in a segment

**Returns**  
`int`: the minimum amount of main memory taken up after the deletion of a contiguous segment of size  $m$ .

sliding window, 求长度为K的subarray的和的最大值, 然后用总和减去

```

int getMinSum(vector<int> nums, int k){
    int left=0;
    int right=0;
    int maxSum = 0;
    int currentSum = 0;
    int totalSum = 0;
    while(right<nums.size()){
        currentSum += nums[right];
        totalSum += nums[right];
        if(right>=k-1){
            maxSum=max(currentSum,maxSum);
            currentSum -= nums[left++];
        }
        right++;
    }
    return totalSum-maxSum;
}

```

## 38. River Control

A Mars rover is directed to move within a square matrix. It accepts a sequence of commands to move in any of the four directions from each cell: [UP, DOWN, LEFT or RIGHT]. The rover starts from cell 0. and may not move diagonally or outside of the boundary.

Each cell in the matrix has a position equal to:

(row \* size) + column

where row and column are zero-indexed, size = row length of the matrix.

Return the final position of the rover after all moves.



**Example**

n = 4

commands = [RIGHT, UP, DOWN, LEFT, DOWN, DOWN]

The rover path is shown below.

The rover path is shown below.

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

RIGHT: Rover moves to position 1

UP: Position unchanged, as the move would take the rover out of the boundary.

DOWN: Rover moves to Position 5.



LEFT: Rover moves to position 4

DOWN: Rover moves to position 8

DOWN: The rover ends up in position 12.

The function returns 12.

```
int riverControl(vector<string> int n)
```

## 39. P-matching score

## 2. Code Question 2

Amazon has an extensive customer database. To prevent customers from having very similar usernames, Amazon's Customer Database Management Team has decided to define a "p-matching" score between two usernames.

Consider two users with their respective User IDs as  $username_1$  of length  $n$  and  $username_2$  of length  $m$ . Each User ID is represented as a string of lowercase English letters. The p-matching score of  $username_1$  with respect to  $username_2$  is the maximum number of distinct indices  $i$  such that the string formed by concatenating characters  $username_1[i], username_1[i+1], \dots, username_1[i+(m-1)\times p]$  can be rearranged to  $username_2$  where  $0 \leq i; i+(m-1)\times p < n; 1 \leq p$ .

Given two strings representing user IDs, find the p-matching between the given user IDs.

### Example

$username_1 = "acaccaa"$ ,  $username_2 = "aac"$  and  $p = 2$ .

- Starting at index  $i = 0$ , increment  $i$  by  $p = 2$ . The string formed by concatenating characters  $username_1[0], username_1[2], username_1[4]$  is  $newString = "aac"$ . It equals  $username_2$  so  $i = 0$  is a valid starting index.
- Starting at index  $i = 1$ , concatenate  $username_1[1], username_1[3], username_1[5]$  into  $newString = "cca"$ . No rearrangement of this string will make it equal to string  $username_2$  so  $i = 1$  is not a valid starting index.
- Starting at index  $i = 2$ , concatenate  $username_1[2], username_1[4], username_1[6]$  into  $newString = "aca"$ . It can be rearranged to  $"aac"$  so  $i = 2$  is a valid starting index.

There are 2 valid starting indices, so the p-matching score is 2.

### Function Description

Complete the function `getPMatchingScore` in the editor below.

`getPMatchingScore` has the following parameters:

`string username1`: the username of 1<sup>st</sup> user.  
`string username2`: the username of 2<sup>nd</sup> user.  
`int p`: the interval at which the characters from `username1` are concatenated

### Returns

int: the p-matching score of `username1` with respect to `username2`.

### Returns

int: the p-matching score of `username1` with respect to `username2`.

### Constraints

- $1 \leq |username_2| \leq |username_1| \leq 10^6$
- $1 \leq p \leq 10^6$
- The strings `username1` and `username2` only contain lowercase Latin letters.

### ▼ Input Format For Custom Testing

The first line contains the string `username1`.

The second line contains the string `username2`.

The last line contains a positive integer `p`.

### ▼ Sample Case 0

#### Sample Input For Custom Testing

```
STDIN      FUNCTION
-----
acbba →  username1 = "acbba"
acb     →  username2 = "acb"
2       →  p = 2
```

#### Sample Output

```
2
```

#### Explanation

Take starting indices as 0 and 1 (zero-based indexing). Beyond that there are not enough characters available.

The newly formed strings are "abc" and "cba" respectively, both of which can be rearranged to `username2 = "acb"`.

### ► Sample Case 1



```
int pMatchingScore(string s1, string s2, int p){\n    int result = 0;\n    int start=0;\n    while(start<s1.length()-(s2.length()-1)*p){\n        int ps1=start;\n        int ps2=0;\n        result += valid(s1,s2,ps1,p);\n        start++;\n    }\n    bool valid(string &s1, string &s2, int ps1, int p){\n        ps2=0;
```

```
while(ps < s2.length() && ps1 < s1.length()){
    if(s1[ps1] == s2[ps2]){
        ps2++;
        ps1 += p;
    }
    else return false;
}
return ps == s2.length();
}
```

要理解为什么能用slicing window，你得先发现遍历原字符串是存在字符串复用的可能性的。. 1point 3 acres

传统的sliding window每次往右边移动一格就能得到一个需要evaluate的substring，但这个题因为有个长度为k的间隔 你每次要往右移动k个坐标，才能得到下一个要evaluate的substring 然后你会发现这个substring跟你刚才evaluate的那个只有首尾两个字符之差。比如原字符串string1是abcdefg，string2是ace，k = 2，那就相当于对aceg做一次定长为3的sliding window 对bdfh也做一次，总共分成了k个需要进行sliding window的substring

## 40. Processing Logs 交易日志处理 返回超过阈值的用户ID

A Company parses logs of online store user transactions/activity to flag fraudulent activity.

The log file is represented as an Array of arrays. The arrays consist of the following data:

```
[<userId> <userId> <# of transactions>]
```

For example:

```
[345366 89921 45]
```

Note: the data is space delimited

So, the log data would look like:

```
[  
[345366 89921 45],  
[029323 38239 23]  
...  
]
```

Write a function to parse the log data to find distinct users that meet or cross a certain threshold.

The function will take in 2 inputs:

logData : Log data in form an array of arrays

Output:

It should be an array of `userids` that are sorted.

If same `userid` appears in the transaction as `userId1` and `userId2`, it should count as one occurrence, not two.

Example:

Input:

logData:

Example:

Input:

logData:

```
[  
[345366 89921 45],  
[029323 38239 23],  
[38239 345366 15],  
[029323 38239 77],  
[545366 38239 23],  
[029323 345366 13],  
[38239 38239 23]  
...  
]
```

threshold: 3

Output: [345366 , 38239, 029323]

Explanation:

Given the following counts of `userids`, there are only 3 `userids` that meet or exceed the threshold of 3.

345366 -4 , 38239 -5 , 029323-3 , 89921-1

```
vector<string> processLog(vector<string> log, int k){  
    vector<string> result;  
    unordered_map<string,int> m;  
    for(auto s: log){  
        string s1;  
        string s2;  
        int nspace=0;  
        for(int i=0;i<s.length();i++){  
            if(s[i]==' ')  
                nspace++;  
            else if(nspace==0)  
                s1 += s[i];  
            if(nspace==1)  
                s2 += s[i];  
        }  
        result.push_back(s1);  
        result.push_back(s2);  
    }  
    return result;  
}
```

```

    }
    if(s1==s2)
        m[s1]++;
    else{
        m[s1]++;
        m[s2]++;
    }
}
for(auto tmp:m){
    if(m.second>k)
        result.push_back(tmp.first);
}
return result;
}

```

## 41 Pick song with total time = rideDuration - 30

**BETA** Can't read the text? [Switch theme](#)

### 1. Code Question 1

Amazon Music is working on a new feature to pair songs together to play while on the bus. The goal of this feature is to select two songs from a list that will end exactly 30 seconds before the listener reaches their stop. You are tasked with writing the method that selects the songs from a list. Each song is assigned a unique ID, numbered from 0 to N-1.

**Assumptions:**

1. You will pick exactly two songs.
2. You cannot pick the same song twice.
3. If you have multiple pairs with the same total time, select the pair with the longest song.
4. There are at least two songs available.

Write an algorithm to find the IDs of two songs whose combined runtime will finish exactly 30 seconds before the bus arrives, keeping the original order.

**Input**  
The input to the function/method consists of two arguments - *rideDuration*, an integer representing the duration of the ride in seconds; *songDurations*, a list of integers representing the duration of the songs.

**Output**  
Return a pair of integers representing the IDs of two songs whose combined runtime will finish exactly 30 seconds before the rider reaches their stop. If no such pair is possible, return a pair with <-1, -1>.

**Constraints**  
 $0 \leq \text{songDurations}[i] \leq 1000$   
 $0 \leq i < \text{number of songs}$

**Example**  
Input:  
*rideDuration* = 90

**list. Each song is assigned a unique ID, numbered from 0 to N-1.**

**Assumptions:**

1. You will pick exactly two songs.
2. You cannot pick the same song twice.
3. If you have multiple pairs with the same total time, select the pair with the longest song.
4. There are at least two songs available.

**Write an algorithm to find the IDs of two songs whose combined runtime will finish exactly 30 seconds before the bus arrives, keeping the original order.**

**Input**  
The Input to the function/method consists of two arguments - *rideDuration*, an Integer representing the duration of the ride in seconds; *songDurations*, a list of integers representing the duration of the songs.

**Output**  
Return a pair of integers representing the IDs of two songs whose combined runtime will finish exactly 30 seconds before the rider reaches their stop. If no such pair is possible, return a pair with <-1, -1>.

**Constraints**  
 $0 \leq \text{songDurations}[i] \leq 1000$   
 $0 \leq i < \text{number of songs}$

**Example**  
Input:  
*rideDuration* = 90  
*songDurations* = [1, 10, 25, 35, 60]

**Explanation:**  
During the ride duration of 90 seconds, the rider listens to the third and fourth songs (2nd and 3rd index, respectively) which end exactly 30 seconds before the bus arrives at their stop. If two songs have the same duration, select the option with the lowest index.

```

vector<int> findsong(int rideDuration, vector<int> songDuration){
    unordered_map<int,int> s;
    int target = rideDuration - 30;
    vector<int> result(2,-1);
    for(int i=0;i<songDuration.size();i++){
        if(s.find(target-songDuration[i])!=s.end()){
            if(abs([target-2*songDuration])>abs(songDuration[result[0]]-
songDuration[result[1]])){
                result[0]=s[target-songDuration[i]];
                result[1]=i;
            }
        }
        s[songDuration[i]]=i;
    }
    return result;
}

```

# 42 .View items from the economy mart platform

The screenshot shows a programming challenge interface. On the left, there is a 'Code Question 2' section with a note about Economy Mart's unusual way of displaying items. It lists two types of entries: 'INSERT' followed by item name and price, and 'VIEW' followed by item ID. A table of entries is provided:

	Parameter 1	Parameter 2	Parameter 3
Entry 1	"INSERT"	itemName	price
Entry 2	"VIEW"	"-"	"-"

Note: The price of each item is in string format so you may need to convert it to an integer before using it.

The right side contains two code editor panes. The top pane shows a snippet of C++ code with comments explaining the logic for handling log entries and determining the cheapest item. The bottom pane shows a diagram illustrating the state of a database after log entries. The database has five items: coffee (worth 1), milk (worth 4), gum (worth 5), and pizza (worth 3). The diagram shows a cursor pointing to the coffee entry, indicating it is the cheapest item in the sorted list.

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
using namespace std;

int main()
{
    vector<vector<string>> entries = {{"INSERT", "milk", "4"}, 
    {"INSERT", "coffee", "3"}, \ 
                                {"VIEW", "-"}, {"VIEW", "-"}, {"INSERT", "pizza", "5"}, 
    {"INSERT", "apple", "3"}, {"INSERT", "gum", "1"}, \
                                {"VIEW", "-"}, {"VIEW", "-"}};

    vector<pair<string, string>> v;
    int k=0;
    for(auto e:entries){
        if(e[0]=="INSERT")
            v.push_back(pair<string, string>(e[2], e[1]));
        else{
            sort(v.begin(), v.end());
            cout<<v[k].second<<endl;
            k++;
        }
    }
    return 0;
}
//coffee
//apple
```

# 43. findMaximumMaximaCount

Amazon has a string of categories of items purchased by a particular customer, each represented as a lowercase English Letter. To analyze customer behavior we define a metric called the MaximaCount of a category c is the maximum among all categories is defined as the number of indices i such taht the frequency of char is maximum in the prefix of the string up to the index i given the string categories, find the maximum MaximaCount amoung all the categories

Given categories = "bccaaacb" there are three categories [a,b,c]

MaximaCount of a = 4 at indices 5,6,7,8

MaximaCount of b = 2 at indices 1,2

MaximaCount of c = 6 at indices 2,3,4,5,7,8

complete the findMaximumMaximaCount in the edior below the function returs an integer denoting the maximum attainable favourabilit

categories = adbcbcbcc

out = 6

the string categories consists of lorcase english characters only.

```
int maxMaximaCount(string s)
{
    priority_queue<pair<int, char>> pq; // To get the character with the maximum
    frequency at any time
    map<char, int> mp; // A map that will store the frequency of each character
    map<int, set<char>> freq; // A map to store all the characters that have a
    same frequency. Used a set because searching and deleting in set will be faster
    map<char, int> maxima; // A map that will store the maxima of a character
    for (auto &x : s)
    {
        if (mp[x] > 0)
        {
            auto &cur = freq[mp[x]];
            cur.erase(cur.find(x)); // erasing the character from the old
frequency
        }
        mp[x]++; // increasing the frequency of the current character and then
adding it to the frequency map and priority queue
        freq[mp[x]].insert(x);
        pq.push({mp[x], x});
        int val = pq.top().first; // getting the element with the maximum
frequency at the current position.
        for (auto &y : freq[val]) // next updated the maxima of all the elements
that have the highest frequency
            maxima[y]++;
    }
    int ans = 0;
    for (auto &x : maxima)
        ans = max(ans, x.second); // finding the max maxima
    return ans;
}
```

## 44. check similar password

Amazon would like to enforce a new password policy that when a user changes their password, the new password cannot be similar to the current one. To determine whether two passwords are similar, they take the new password choose a set of indices and change the characters at these indices to the next cyclic character exactly once. If 'a' is changed to 'b', 'b' to 'c' and so on, and 'z' changes to 'a' the password is said to be similar after applying the operation the old password is a subsequence of the new password.

The developers come up with a set of n password change requests, where newPasswords denotes the array of new passwords and oldPasswords denotes the array of old password. For each pair newPasswords[i] and oldPasswords[i], return "YES" if the passwords are similar, that is, newPasswords[i] becomes a subsequence of oldPasswords[i] after performing the operation and "NO" otherwise.

Example the two lists of passwords are given as newPasswords = ["baacbabc", "accdb", "baacba"], and oldPasswords = ["abdbc", "ach", "abb"]

consider the first pair: newPasswords[0] = "baacbabc" and oldPasswords = "abdbc". Change "ac" to "bd" at the 3rd and 4th positions and "b" to "c" at the last position  
the answer for the pair is YES

the newPasswords[1] = "accdb" and oldPasswords = "ach". it is not possible to change the character of the new password to "h" which occurs in the old password so there is no subsequence that matches the answer for this pair is NO

newPasswords[2] = "baacba" and oldPasswords = "abb" the answer for this pair is YES

return ["YES", "NO", "YES"]

Starting code is

```
public static List checkSimilarPasswords(List newPasswords, List oldPasswords){  
}
```

```
#include<bits/stdc++.h>  
using namespace std;  
int main()  
{  
    int n;  
    cin >> n;  
    vector<string> nwp;  
    vector<string> oldp;  
    int s = 0, mx = 0;  
    for (int i = 0; i < n; i++)  
    {  
        string p;  
        cin >> p;  
        nwp.push_back(p);  
    }  
    for (int i = 0; i < n; i++)  
    {  
        string p;  
        cin >> p;  
        oldp.push_back(p);  
    }  
    for (int i = 0; i < n; i++)  
    {  
        int p1 = 0;  
        for (int j = 0; j < nwp[i].size(); j++)
```

```
{  
    if (oldp[i][p1] == nwp[i][j] || oldp[i][p1] == nwp[i][j] + 1 ||  
(oldp[i][p1] == 'a' && nwp[i][j] == 'z'))  
    {  
        p1 += 1;  
    }  
    if (p1 == oldp[i].size())  
        break;  
}  
if (p1 == oldp[i].size())  
    cout << "YES";  
else  
    cout << "NO";  
}  
}
```

## 45.makePowerNonDecreasing

---

## 1. Code Question 1

AWS provides scalable systems. A set of  $n$  servers are used for horizontally scaling an application. The goal is to have the computational power of the servers in non-decreasing order. To do so, you can increase the computational power of each server in any contiguous segment by  $x$ . Choose the values of  $x$  such that after the computational powers are in non-decreasing order, the sum of the  $x$  values is minimum.

### Example

There are  $n = 5$  servers and their computational power = [3, 4, 1, 6, 2].

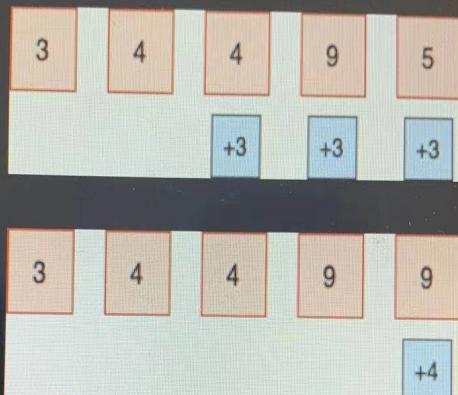


Add 3 units to the subarray (2, 4) and 4 units to the subarray (4, 4). The final arrangement of the servers is: [3, 4, 4, 9, 9]. The answer is  $3 + 4 = 7$ .



@一亩三分地

Add 3 units to the subarray (2, 4) and 4 units to the subarray (4, 4). The final arrangement of the servers is: [3, 4, 4, 9, 9]. The answer is  $3 + 4 = 7$ .



### Function Description

Complete the function `makePowerNonDecreasing` in the editor below.

`makePowerNonDecreasing` has the

following parameter(s):

*int power[n]: the computational powers of  $n$  different servers*



@一亩三分地

```
vector<int> makePowerNonDecreasing(vector<int> v){  
    int result=0;  
    if(v.size()==1)  
        return v;  
    for(int i=1;i<v.size();i++){  
        v[i]+=result;  
        if(v[i]<v[i-1])  
            result += v[i-1]-v[i];  
    }  
    return result;  
}
```

## 46 get maxValue, array starting with one, adjacent value should have difference $\leq 1$

The screenshot shows a challenge titled "1. Code Question 1" on the HackerRank platform. The challenge is part of a Day 1 orientation at Amazon. It asks to complete a function named `getMaxValue` that takes a vector of integers as input. The first element of the array must be 1, and the difference between adjacent integers must not be greater than 1. The code editor shows the beginning of the function definition:

```
1 > #include <bits/stdc++.h> ...  
9  
10 /*  
11 * Complete the 'getMaxValue' function below.  
12 *  
13 * The function is expected to return an INTEGER.  
14 * The function accepts INTEGER_ARRAY arr as parameter.  
15 */  
16  
17 int getMaxValue(vector<int> arr) {  
18  
19 }  
20  
21 > int main() ...
```

The challenge interface includes a timer showing "1h 9m left", a sidebar with navigation links, and a footer with "MacBook Pro" and the logo "@一亩三分地".

```
class Solution {
public:
    int maximumElementAfterDecrementingAndRearranging(vector<int>& arr) {
        sort(arr.begin(), arr.end());
        arr[0]=1;
        for(int i=1;i<arr.size();i++){
            if(arr[i]>arr[i-1]+1)
                arr[i]=arr[i-1]+1;
        }
        return arr[arr.size()-1];
    }
};
```

## 47. 搬箱子一次搬2/3个 count frequency + coin change, hash\_map + dp (LC 2244)

---

1h 9m left



ALL



1

2

There were a large number of orders placed on Amazon Prime Day. The orders are packed and are at the warehouse ready to be delivered. The delivery agent needs to deliver them in as few trips as possible.

In a single trip, the delivery agent can choose packages following either of two rules:

- Choose two packages with the same weight
- Choose three packages with the same weight

Determine the minimum number of trips required to deliver the packages. If it is not possible to deliver all of them, return -1.

#### Example

*weights = [2, 4, 6, 6, 4, 2, 4]*

The agent needs a minimum of 3 trips as shown below. Return 3 as the

Info

1  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31 >  
58

Test R

Lenovo

```
class Solution {
public:
    int minimumRounds(vector<int>& tasks) {
        unordered_map<int,int> m;
        int l=0;
        int result=0;
        for(auto i:tasks){
            m[i]++;
            l=max(l,m[i]);
        }
        l=max(l,4);
        vector<int> dp(l+1,0);
```

```

dp[2]=1;
dp[3]=1;
dp[4]=2;
for(int i=5;i<=l;i++){
    dp[i]=min(dp[i-2]+1,dp[i-3]+1);
}
for(auto tmp:m){
    if(tmp.second == 1)
        return -1;
    result += dp[tmp.second];
}
return result;
};

};

#

```

## 48 Merge Intervals

```

vector<vector<int>> getNonInclusiveRetailor(vector<vector<int>> v){
    sort(v.begin(),v.end());
    int result = 0;
    vector<int> current;
    for(int i=0;i<v.size();i++){
        if(current.empty() || v[i][0]>=current[1])
            current = v[i];
        else{
            current[1]=max(current[1],v[i][1]);
            result++;
        }
    }
    return result;
}

```

## 49 number of subarray with k odd numbers

brute force:  $O(n^2)$

```

int countSubarray(vector<int> v, int k){
    vector<int> ind; //record the index of odd numbers
    ind.push_back(-1);
    for(int i=0;i<v.size();i++){
        if(i%2==0)
            ind.push_back(i);
    }
    ind.push_back(v.size());
    for(int j=1;j+k-1<ind.size();j++){
        int left = ind[j]-ind[j-1];
        int right = ind[j+k]-ind[j+k-1];
        result += left*right;
    }
    return result;
}

```

# 50 find longest non-increasing singlylindedlist

 Picture3

## max consecutive ones(LC 1004)

把 k 个 0 变成 1 后最长的 1, 滑动窗口保持当前window最多只有k个0.

```
class Solution {
public:
    int longestOnes(vector<int>& nums, int k) {
        int left=0;
        int right=0;
        int result=0;
        while(right<nums.size()){
            if(nums[right]==0)
                k--;
            while(k<0)
                k += (nums[left++]==0);
            result=max(result,right-left+1);
            right++;
        }
        return result;
    }
};
```

## Minimum number of swaps required to make a binary string palindrome

<https://www.1point3acres.com/bbs/thread-922001-1-1.html>

<https://devsolus.com/2022/04/26/minimum-number-of-swaps-required-to-make-a-binary-string-palindrome/>

Give a binary string consisting of 0's and 1's only. E.g., 0100101. We are allowed to pick any two indexes and swap them. We have to return the minimum number of swaps required to make

it a palindrome or -1 if it cannot. The string 0100101 can be made a palindrome by swapping (3,4)-> 0101001 and swapping (0,1) -> 1001001 which is a palindrome. In this case, the correct answer is 2.

You could compare the outermost two digits: if they are the same, nothing needs to happen, nor should these digits ever be involved in a swap.

If they are different, take note of that, and don't swap anything yet.

Continue with the second digit at each end. Compare them. If they are different, and this is the second time we have a difference, note how we can solve those two differences with one swap. An example:

10.....10

With one swap this can become:

```
10.....01
```

So we can resolve two differences with one swap. This means that if we find an even number of differences — as we walk along the binary string from the two ends inwards — there is a solution, and it takes half that number of swaps.

If the number of differences is odd, there is still a solution if the input string is odd, because then that middle digit can be swapped with a digit from that last difference, and so get a palindrome.

There is however no solution when the number of differences (in this algorithm) is odd and the number of digits in the input is even.

Now, you don't actually have to perform the swaps, as the challenge only asks for the number of swaps.

This leads to the following code:

```
def numswaps(binary):
    n = len(binary)
    count = 0
    for i in range(n // 2):
        if binary[i] != binary[n - i - 1]:
            count += 1
    if count % 2 == 1 and n % 2 == 0:
        return -1
    return (count + 1) // 2
```

## Z Sequence

A Z sequence is defined as:

$$Z_i = P \times X(Z_{i-1}) + Q \text{ for } i > 0$$

$$Z_0 = 2 \text{ for } i = 0$$

$X(K)$  is defined as the number of set bits in the binary form of a number  $K$ .

Print the number of set bits in the binary form of  $Z_N$

Example

$$N = 2$$

$$P = 1$$

$$Q = 3$$

Approach

So,  $Z[0] = 2$ ,  $Z[1] = PX[2] + Q$ . Now  $X[2] = 1$  as 2 can be written as 10 in binary form So,  $Z[1] = 11 + 3 = 4$ . Similarly,  $Z[2] = PX[4] + Q$ . Now 4 can be written as 100. So,  $Z[2] = 11 + 3 = 4$ . Now answer is the number of set bits in  $Z[2] = 4$ , so 1.

### EXAMPLE:

Sample input

1

3 5 1

Sample output

1

Explanation

Based on expression, sequence will be like 2, 8, 8,...

As N=1 so Z1 is 8 Number of set bits in 8(i.e 1000) is 1

```
#include <bits/stdc++.h>
using namespace std;

int setBitsInZofI(int n, int p, int q)
{
    // z(0) = 2
    unsigned int pz = 2;
    unsigned int ans = pz;

    for (int i = 1; i <= n; i++)
    {
        ans = p * __builtin_popcount(pz) + q;
        pz = ans;
    }
    return __builtin_popcount(ans);
}
```

## 被2或者4整除

---

## 1. Code Question 1

Given an integer denoting a total number of wheels, help Amazon Logistics find the number of different ways to choose a fleet of vehicles from an infinite supply of two-wheeled and four-wheeled vehicles such that the group of chosen vehicles has that exact total number of wheels. Two ways of choosing vehicles are considered to be different if and only if they contain different numbers of two-wheeled or four-wheeled vehicles.

For example, if our array *wheels* = [4,5,6] our return array would be *res* = [2, 0, 2]. Case by case, we can have 1 four-wheel or 2 two-wheel to have 4 wheels. We cannot have 5 wheels. We can have 1 four-wheel and 1 two-wheel or 3 two-wheel vehicles in the final case.

### Function Description

Complete the function *chooseFleets* in the editor below. The function should return an array of integers representing the answer for each *wheels[i]*.

*chooseFleets* has the following parameter(s):

*wheels[wheels[0],...wheels[n-1]]*: an array of integers

### Constraints

- $1 \leq n \leq 10^5$
- $1 \leq \text{wheels}[i] \leq 10^6$

# LC 1730

<https://www.1point3acres.com/bbs/thread-915292-1-1.html>

## box 版本号

<https://www.1point3acres.com/bbs/thread-922190-1-1.html>

<p>1h 41m left</p> <p>BETA Can't read the text? Switch theme</p> <p><b>1. Code Question 1</b></p> <p>ALL</p> <p>Your team at Amazon has been contracted by a telecommunications company that is trying to upgrade junction boxes all over Techlandia. Some of the junction boxes have already been upgraded, and other boxes have not. Your task is to identify the oldest boxes that need to be upgraded first but leaving the newer model boxes so that they will not be prioritized.</p> <p>1 All the junction boxes are identified by an alphanumeric identifier, followed by space delimited version information. The older generation uses space delimited lowercase English strings to identify the version, but the newer generation uses space delimited positive integers to identify the version. Your task is to sort the junction boxes in the following order:</p> <p>2 The older generation junction boxes should be returned first, sorted by lexicographic ordering of alphabetic version.</p> <p>3 If there are any ties in the older generation, ties should be broken by the alphanumeric identifier.</p> <p>4 The newer generation boxes must all come after the older generation, in the original order they were given in the input.</p> <p>Write a function or method to return a list of strings representing the correctly prioritized orders according to this system.</p> <p><b>Input</b> The input to the function/method consists of one argument: <i>boxList</i>; a list of strings representing all of the identifiers and version information.</p> <p><b>Output</b> Return a list of strings representing the correctly prioritized orders according to this system.</p> <p><b>Constraints</b> <math>0 \leq \text{number of boxes} \leq 10^3</math></p> <p><b>Note</b> The junction box identifier consists of only lower case English characters and numbers. Sorting for tiebreaks should use ASCII value – as an example, the order identifier 'a1' should come before the order identifier 'aa'.</p>	<p>1h 41m left</p> <p><b>Input</b> The input to the function/method consists of one argument: <i>boxList</i>, a list of strings representing all of the identifiers and version information.</p> <p><b>Output</b> Return a list of strings representing the correctly prioritized orders according to this system.</p> <p><b>Constraints</b> <math>0 \leq \text{number of boxes} \leq 10^3</math></p> <p><b>Note</b> The junction box identifier consists of only lower case English characters and numbers. Sorting for tiebreaks should use ASCII value – as an example, the order identifier 'a1' should come before the order identifier 'aa'.</p> <p><b>Examples</b> Input: <i>boxList</i> = [ykc 82 01] [eo first qpx] [09z cat hamster] [06f 12 25 6] [az0 first qpx] [236 cat dog rabbit snake]</p> <p>Output: [236 cat dog rabbit snake] [09z cat hamster] [az0 first qpx] [eo first qpx] [ykc 82 01] [06f 12 25 6]</p> <p>Explanation: The four old generation junction boxes should come first, with the "cat dog rabbit snake" box coming before the "cat hamster type". Since the two boxes of type "first qpx" have the same version information, they should come next, using the "az0" identifier to come before the "eo" identifier. Finally, the already upgraded junction boxes should come last, in the original order, they were provided in the file.</p>
---	---

先把新的旧的根据string最后一个字母分出来

然后对旧的排序，再合并到一起。

## Route Pair

1h 41m left	<p><b>BETA</b> Can't read the text? <a href="#">Switch theme</a></p> <h3>3. Code Question 2</h3> <p><b>ALL</b> Amazon Prime Air is developing a system that divides shipping routes using flight optimization routing systems to a cluster of aircrafts that can fulfill these routes. Each shipping route is identified by a unique integer identifier, requires a fixed non-zero amount of travel distance between airports, and is defined to be either a forward shipping route or a return shipping route. Identifiers are guaranteed to be unique within their own route type, but not across route types.</p> <p><b>1</b></p> <p><b>2</b></p> <p><b>3</b> Each aircraft should be assigned two shipping routes at once: one forward route and one return route. Due to the complex scheduling of flight plans, all aircraft have a fixed maximum operating travel distance, and cannot be scheduled to fly a shipping route that requires more travel distance than the prescribed maximum operating travel distance. The goal of the system is to optimize the total operating travel distance of a given aircraft. A forward/return shipping route pair is considered to be "optimal" if there does not exist another pair that has a higher operating travel distance than this pair, and also has a total less than or equal to the maximum operating travel distance of the aircraft.</p> <p>For example, if the aircraft has a maximum operating travel distance of 3000 miles, a forward/return shipping route pair using a total of 2900 miles would be optimal if there does not exist a pair that uses a total operating travel distance of 3000 miles, but would not be considered optimal if such a pair did exist.</p> <p>Your task is to write an algorithm to optimize the sets of forward/return shipping route pairs that allow the aircraft to be optimally utilized, given a list of forward shipping routes and a list of return shipping routes.</p> <p><b>Input</b> The input to the function/method consists of three arguments:</p> <ul style="list-style-type: none"> <li>• <i>maxTravelDist</i>, an integer representing the maximum operating travel distance of the given aircraft;</li> <li>• <i>forwardRouteList</i>, a list of pairs of integers where the first integer represents the unique identifier of a forward shipping route and the</li> </ul>	<p><b>1h 41m left</b></p> <p><b>Output</b> Return a list of pairs of integers representing the pairs of IDs of forward and return shipping routes that optimally utilize the given aircraft. If no route is possible, return a list with empty pair.</p> <p><b>Examples</b></p> <p><b>Example 1:</b> Input: <code>maxTravelDist = 7000 forwardRouteList = [[1,2000],[2,4000],[3,6000]] returnRouteList = [[1,2000]]</code></p> <p><b>1</b></p> <p><b>2</b></p> <p><b>3</b> Output: <code>[[2,1]]</code></p> <p><b>Explanation:</b> There are only three combinations, [1,1], [2,1], and [3,1], which have a total of 4000, 6000, and 8000 miles, respectively. Since 6000 is the largest use that does not exceed 7000, [2,1] is the only optimal pair.</p> <p><b>4</b></p> <p><b>Example 2:</b> Input: <code>maxTravelDist = 10000 forwardRouteList = [[1, 3000], [2, 5000], [3, 7000], [4, 10000]] returnRouteList = [[1, 2000], [2, 3000], [3, 4000], [4, 5000]]</code></p> <p><b>Output:</b> <code>[[2, 4], [3, 2]]</code></p> <p><b>Explanation:</b> There are two pairs of forward and return shipping routes possible that optimally utilizes the given aircraft. Shipping Route ID#2 from the forwardShippingRouteList requires 5000 miles travelled, and Shipping Route ID#4 from returnShippingRouteList also requires 5000 miles travelled. Combined, they add up to 10000 miles travelled. Similarly, Shipping Route ID#3 from forwardShippingRouteList requires 7000 miles travelled, and Shipping Route ID#2 from returnShippingRouteList requires 3000 miles travelled. These also add up to 10000 miles travelled. Therefore, the pairs of forward and return shipping routes that optimally utilize the aircraft are [2, 4] and [3, 2].</p>
-------------	--	--

#

## Minimum difference (LC 2256)