

File: ./algorithm/hard/6/README.md

count string

—Асуулт— “english” : A regular expression is used to describe a set of strings. For this problem the alphabet is limited to ‘a’ and ‘b’.

We define to be a valid regular expression if: 1) is "" or "". 2) is of the form "", where and are regular expressions. 3) is of the form "" where and are regular expressions. 4) is of the form "" where is a regular expression.

Regular expressions can be nested and will always have two elements in the parentheses. (" is an element, " is not; basically, there will always be pairwise evaluation) Additionally, " will always be the second element; " is invalid.

The set of strings recognized by are as follows: 1) If is "", then the set of strings recognized . 2) If is "", then the set of strings recognized . 3) If is of the form "" then the set of strings recognized = all strings which can be obtained by a concatenation of strings and , where is recognized by and by . 4) If is of the form "" then the set of strings recognized = union of the set of strings recognized by and . 5) If is of the form "" then the the strings recognized are the empty string and the concatenation of an arbitrary number of copies of any string recognized by .

Task Given a regular expression and an integer , count how many strings of length are recognized by it.

Input Format

The first line contains the number of test cases . test cases follow. Each test case contains a regular expression, , and an integer, .

Constraints

It is guaranteed that will conform to the definition provided above. Output Format

Print lines, one corresponding to each test case containing the required answer for the corresponding test case. As the answers can be very big, output them modulo .

Sample Input 3

((ab)|(ba)) 2

((a|b)) 5

((a)(b(a*))) 100

Sample Output 2

32

100 Explanation

For the first case, the only strings recognized are "" and "". Of the possible strings of length , of them fit that expression. For the second case, the RegEx recognizes any string of any

length containing only 's and 's. The number of strings of length recognized by this expression is . For the third case, the RegEx recognizes any string having one , preceded and followed by any number of 's. There are strings of length which have a single in them.

—Асуулт—“Монголоор” : Энэ бодлогод бид зөвхөн ‘a’ ба ‘b’ гэсэн хоёр үсгийг агуулсан үгийн олонлогийг дүрсэлдэг тогтмол илэрхийлэл (regular expression) ашиглана.

Тогтмол илэрхийлэл дараах нөхцлүүдийг хангах бол зөвшөөрөгдөнө:

“a” эсвэл “b” бол шууд зөв.

“(R1R2)” хэлбэртэй бол зөв. Энд R1 ба R2 нь өөрсдөө тогтмол илэрхийлэл.

“(R1|R2)” хэлбэртэй бол зөв. Энд | нь эсвэл (OR)-ийн утгатай.

“(R1*)” хэлбэртэй бол зөв. Энэ нь R1-ийн дурын тооны давталтыг илэрхийлнэ (0 ба түүнээс дээш удаа).

Тайлбар:

Илэрхийлэл үргэлж хоёр элементийг хамарсан хаалттай бүтэцтэй байна. Жишээ нь: ((a|b)*) зөв, харин (a|) бол буруу.

- тэмдэг үргэлж хоёр дахь элемент байж болохгүй ((a) буруу), зөвхөн “(R1)” хэлбэртэй байна.

Илэрхийлэл ямар үгсийг хүлээн зөвшөөрөх вэ:

“a” гэвэл зөвхөн “a” үгийг.

“b” гэвэл зөвхөн “b” үгийг.

“(R1R2)” бол R1 ба R2-оор тодорхойлогдох үгсийг нийлүүлэн гарсан бүх үгсийг.

“(R1|R2)” бол R1 эсвэл R2-оор тодорхойлогдох үгс.

“(R1*)” бол R1-оор тодорхойлогдох үгийг дурын тоогоор давтсан үгс (0 удаа давтах ч боломжтой, өөрөөр хэлбэл хоосон үг ч багтана).

Даалгавар: Тогтмол илэрхийлэл r болон бүхэл тоо l өгөгдөх ба, уг илэрхийлэлд таарах яг урт нь l үг хэд байгааг ол.

Оролт:

Эхний мөрөнд туршилтын тоо t (жишээ нь: 3). Дараагийн t мөр бүрд:

Тогтмол илэрхийлэл r

Бүхэл тоо l

Гаралт:

Туршилт бүрийн хувьд 1 мөрөнд, тухайн тогтмол илэрхийлэлд таарах яг урт l үг хэд байгааг хэвлэ.

Тоо их байж болзошгүй учраас $10^9 + 7$ модулоор гаргана.

Жишээ оролт: 3

$((ab)|(ba))^2$

$((a|b))^5$

$((a)(b(a^*)))$ 100 Гаралт:

2

32

100 Тайлбар:

Эхний илэрхийлэл (ab) ба (ba) хоёрын аль нэгийг зөвшөөрнө. 2 тэмдэгттэй үг 2 ширхэг л байна.

Хоёр дахь нь a болон b үсгийг дурын удаа давтахыг зөвшөөрнө. Тиймээс урт нь 5 байх бүх боломжит үгс: $2^5 = 32$

Гурав дахь нь ганц b үсэг дунд нь орсон, хоёр талаараа дурын тооны a үсэгтэй үсгийг зөвшөөрнө. Жишээ нь: $aaabaa$, $baaa$, $aaaaab$, гэх мэт. Яг нэг b ба бусад нь a гэсэн үгс, урт нь 100 байх боломжит ийм үгсийн тоо: 100. —Код—” ” `import java.io.; import java.math.; import java.security.; import java.text.; import java.util.; import java.util.concurrent.; import java.util.function.; import java.util.regex.; import java.util.stream.*; import static java.util.stream.Collectors.joining; import static java.util.stream.Collectors.toList;`

```
class Result { static List<List<int[]>> graph; static List<int[]> dfa; static List<List> matrix; static List finalStates; static List epsilon; static boolean[] visited;
```

```
static void buildNFA(String s, int[] index, int in, int out) {
    if (s.charAt(index[0]) == 'a') {
        graph.get(in).add(new int[]{out, 1});
        index[0]++;
        return;
    }
    if (s.charAt(index[0]) == 'b') {
        graph.get(in).add(new int[]{out, 2});
        index[0]++;
        return;
    }
    int m1 = graph.size();
    graph.add(new ArrayList<>());
    int m2 = graph.size();
    graph.add(new ArrayList<>());
    index[0]++;
    buildNFA(s, index, m1, m2);
    if (s.charAt(index[0]) == '*') {
        graph.get(m2).add(new int[]{m1, 0});
    }
}
```

```

        graph.get(in).add(new int[]{m2, 0});
        graph.get(m2).add(new int[]{out, 0});
        index[0] += 2;
        return;
    }
    if (s.charAt(index[0]) == '|') {
        graph.get(in).add(new int[]{m1, 0});
        graph.get(m2).add(new int[]{out, 0});
        index[0]++;
        buildNFA(s, index, m1, m2);
        index[0]++;
        return;
    }
    graph.get(in).add(new int[]{m1, 0});
    buildNFA(s, index, m2, out);
    index[0]++;
}

static void dfsEpsilon(int node) {
    if (visited[node]) return;
    visited[node] = true;
    epsilon.add(node);
    for (int[] e : graph.get(node)) {
        if (e[1] == 0) dfsEpsilon(e[0]);
    }
}

static void dfs(List<Integer> state, int type) {
    visited = new boolean[graph.size()];
    epsilon = new ArrayList<>();
    for (int node : state) {
        for (int[] e : graph.get(node)) {
            if (e[1] == type) dfsEpsilon(e[0]);
        }
    }
}

static void NFAtoDFA() {
    Map<List<Integer>, Integer> map = new HashMap<>();
    List<List<Integer>> states = new ArrayList<>();
    dfa = new ArrayList<>();
    epsilon = new ArrayList<>();
    visited = new boolean[graph.size()];
    dfsEpsilon(0);
    Collections.sort(epsilon);
    states.add(new ArrayList<>(epsilon));
    map.put(new ArrayList<>(epsilon), 0);
    dfa.add(new int[]{-1, -1});
    int n = 1;
    for (int i = 0; i < n; i++) {

```

```

        dfs(states.get(i), 1);
        Collections.sort(epsilon);
        if (!epsilon.isEmpty()) {
            List<Integer> eCopy = new ArrayList<>(epsilon);
            if (!map.containsKey(eCopy)) {
                map.put(eCopy, n++);
                states.add(eCopy);
                dfa.add(new int[]{-1, -1});
            }
            dfa.get(i)[0] = map.get(eCopy);
        }
        dfs(states.get(i), 2);
        Collections.sort(epsilon);
        if (!epsilon.isEmpty()) {
            List<Integer> eCopy = new ArrayList<>(epsilon);
            if (!map.containsKey(eCopy)) {
                map.put(eCopy, n++);
                states.add(eCopy);
                dfa.add(new int[]{-1, -1});
            }
            dfa.get(i)[1] = map.get(eCopy);
        }
    }
    matrix = new ArrayList<>();
    for (int i = 0; i < n; i++) {
        matrix.add(new ArrayList<>(Collections.nCopies(n, 0)));
    }
    for (int i = 0; i < n; i++) {
        if (dfa.get(i)[0] != -1) matrix.get(i).set(dfa.get(i)[0],
matrix.get(i).get(dfa.get(i)[0]) + 1);
        if (dfa.get(i)[1] != -1) matrix.get(i).set(dfa.get(i)[1],
matrix.get(i).get(dfa.get(i)[1]) + 1);
    }
    finalStates = new ArrayList<>();
    for (int i = 0; i < states.size(); i++) {
        visited = new boolean[graph.size()];
        epsilon = new ArrayList<>();
        for (int node : states.get(i)) dfsEpsilon(node);
        if (epsilon.contains(1)) finalStates.add(i);
    }
}

static void mmult(List<List<Integer>> A, List<List<Integer>> B) {
    int n = A.size();
    List<List<Integer>> R = new ArrayList<>();
    for (int i = 0; i < n; i++) R.add(new ArrayList<>(Collections.nCopies(n,
0)));
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++) {
            long sum = 0;

```

```

        for (int k = 0; k < n; k++) {
            sum = (sum + A.get(i).get(k) * 1L * B.get(k).get(j)) %
1000000007;
        }
        R.get(i).set(j, (int) sum);
    }
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            A.get(i).set(j, R.get(i).get(j));
}

```

```

static void mexp(List<List<Integer>> M, int power) {
    int n = M.size();
    List<List<Integer>> R = new ArrayList<>();
    for (int i = 0; i < n; i++) {
        R.add(new ArrayList<>(Collections.nCopies(n, 0)));
        R.get(i).set(i, 1);
    }
    while (power > 0) {
        if ((power & 1) != 0) mmult(R, M);
        power >>= 1;
        if (power > 0) mmult(M, M);
    }
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            M.get(i).set(j, R.get(i).get(j));
}

```

```

public static int countStrings(String r, int l) {
    graph = new ArrayList<>();
    graph.add(new ArrayList<>());
    graph.add(new ArrayList<>());
    int[] index = {0};
    buildNFA(r, index, 0, 1);
    NFAtoDFA();
    mexp(matrix, l);
    long res = 0;
    for (int j : finalStates) {
        res = (res + matrix.get(0).get(j)) % 1000000007;
    }
    return (int) res;
}

```

```

}

```

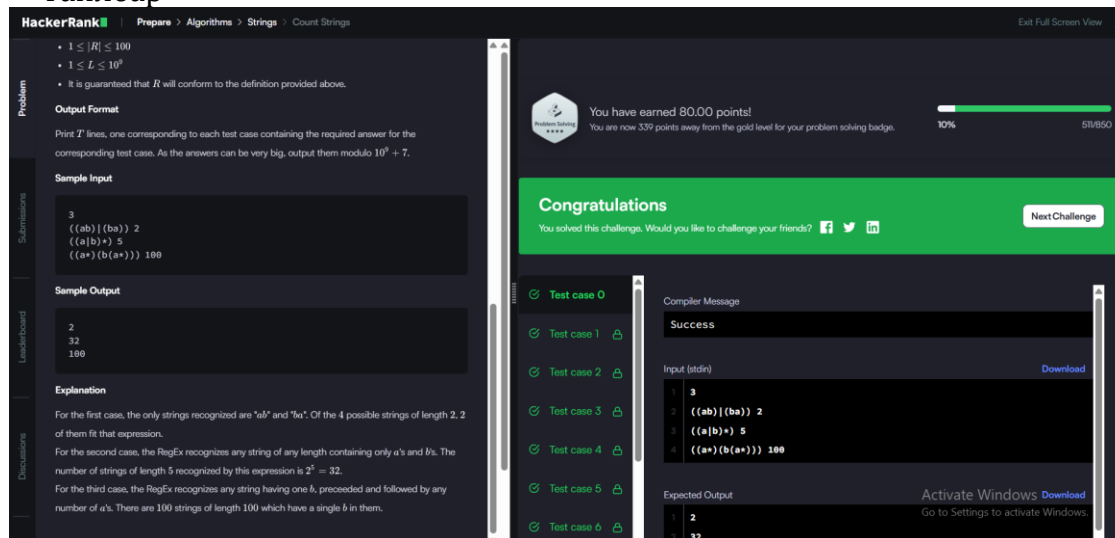
```

public class Solution { public static void main(String[] args) throws IOException {
    BufferedReader bufferedReader = new BufferedReader(new
    InputStreamReader(System.in)); BufferedWriter bufferedWriter = new
    BufferedWriter(new OutputStreamWriter(System.out)); int t =
    Integer.parseInt(bufferedReader.readLine().trim()); for (int t1tr = 0; t1tr < t; t1tr++) {

```

```
String[] parts = bufferedReader.readLine().split(" "); String r = parts[0]; int l = Integer.parseInt(parts[1]); int result = Result.countStrings(r, l); bufferedWriter.write(String.valueOf(result)); bufferedWriter.newLine(); } bufferedReader.close(); bufferedWriter.close(); }
```

—Тайлбар—



File: ./algorithm/med/2/README.md

Highest Value Palindrome

—Асуулт—“english” :

Palindromes are strings that read the same from the left or right, for example madam or 0110.

You will be given a string representation of a number and a maximum number of changes you can make. Alter the string, one digit at a time, to create the string representation of the largest number possible given the limit to the number of changes. The length of the string may not be altered, so you must consider 's left of all higher digits in your tests. For example is valid, is not.

Given a string representing the starting number, and a maximum number of changes allowed, create the largest palindromic string of digits possible or the string '-1' if it is not possible to create a palindrome under the constraints.

Example

Make replacements to get .

Make replacement to get .

Function Description

Complete the `highestValuePalindrome` function in the editor below.

`highestValuePalindrome` has the following parameter(s):

`string s`: a string representation of an integer
`int n`: the length of the integer
`string int k`: the maximum number of changes allowed
Returns

`string`: a string representation of the highest value achievable or `-1`

Input Format
The first line contains two space-separated integers, `n` and `k`, the number of digits in the number and the maximum number of changes allowed. The second line contains an `n`-digit string of numbers.

Constraints

Each character in the number is an integer where `0 ≤ digit ≤ 9`. **Output Format**

Sample Input 0

STDIN	Function
4 1	<code>n = 4, k = 1</code>
3943	<code>s = '3943'</code>

Sample Output 0

3993

Sample Input 1

6 3 092282

Sample Output 1

992299

Sample Input 2

4 1 0011

Sample Output 2

0011

-1 Explanation

Sample 0

There are two ways to make a palindrome by changing no more than `k` digits: `##` — Асуулт —
“Монголоор” : Бодлогын тайлбар : Палиндром гэдэг нь хоёр талаасаа ижилхэн уншигддаг тоо буюу тэмдэгт мөр юм. Жишээ нь: “madam” болон “0110” бол палиндромууд. Танд нэг тоог тэмдэгт мөр (`string`) хэлбэрээр өгнө. Та тухайн тэмдэгт мөр дотроос зарим цифрүүдийг өөрчлөх боломжтой. Гэхдээ хамгийн ихдээ `k` удаа өөрчилж болно. Тэмдэгт мөрийн уртыг өөрчилж болохгүй. Зорилго нь: боломжийн хамгийн их утгатай палиндром үүсгэх юм. Хэрвээ палиндром үүсгэх боломжгүй бол “-1” гэж буцаах хэрэгтэй. Функцийн тайлбар : `highestValuePalindrome` гэдэг функц бичих шаардлагатай. Энэ функц гурван параметр авна: Нэгт: `s` — анхны тоог илэрхийлэх тэмдэгт мөр Хоёрт: `n` — тэмдэгт мөрийн урт Гуравт: `k` — өөрчилж болох дээд тоо Функцийн зорилго бол: хамгийн том утгатай палиндром үүсгээд буцаах. Хэрвээ бүтэх боломжгүй бол “-1” буцаана. Оролт болон гаралтын жишээ: Жишээ 1:

Оролт: n = 4, k = 1, s = "3943" Гаргалт: "3993" Тайлбар: Зөвхөн 4-ийг 9 болгож 1 өөрчлөлт хийснээр палиндром үүссэн. Жишээ 2: Оролт: n = 6, k = 3, s = "092282" Гаргалт: "992299" Тайлбар: 3 өөрчлөлт хийснээр хамгийн том палиндром үүссэн. Жишээ 3: Оролт: n = 4, k = 1, s = "0011" Гаргалт: "-1" Тайлбар: Зөвхөн 1 удаа өөрчлөх боломжтой ч палиндром болгоход 2 өөрчлөлт хэрэгтэй тул боломжгүй. ## —Код—
"java"

```
import java.io.; import java.math.; import java.security.; import java.text.; import java.util.;
import java.util.concurrent.; import java.util.function.; import java.util.regex.; import
java.util.stream.*; import static java.util.stream.Collectors.joining; import static
java.util.stream.Collectors.toList;
```

```
class Result {
```

```
/*
 * Complete the 'highestValuePalindrome' function below.
 *
 * The function is expected to return a STRING.
 * The function accepts following parameters:
 * 1. STRING s
 * 2. INTEGER n
 * 3. INTEGER k
 */
```

```
public static String highestValuePalindrome(String s, int n, int k) {
```

```
    char[] arr = s.toCharArray();
    int left = 0;
    int right = n - 1;
    int changes = 0;

    while (left < right) {
        if (arr[left] != arr[right]) {
            char maxChar = (char) Math.max(arr[left], arr[right]);
            arr[left] = maxChar;
            arr[right] = maxChar;
            changes++;
        }
        left++;
        right--;
    }

    if (changes > k) {
        return "-1";
    }

    left = 0;
    right = n - 1;
    while (left <= right) {
```

```

        if (left == right) {
            if (k - changes > 0) {
                arr[left] = '9';
            }
        } else {
            if (arr[left] < '9') {
                if (k - changes >= 2 && arr[left] == s.charAt(left) &&
arr[right] == s.charAt(right)) {

                    arr[left] = '9';
                    arr[right] = '9';
                    changes += 2;
                } else if (k - changes >= 1 && s.charAt(left) !=
s.charAt(right)) {

                    arr[left] = '9';
                    arr[right] = '9';
                    changes++;
                }
            }
        }
        left++;
        right--;
    }

    return new String(arr);
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    // Input: Read the number of digits, maximum changes, and the string
    representation
    int n = scanner.nextInt();
    int k = scanner.nextInt();
    String s = scanner.next();

    // Call the function to find the highest value palindrome
    String result = highestValuePalindrome(s, n, k);

    // Output: Print the result
    System.out.println(result);

    // Close the scanner
    scanner.close();
}
}

```

```

public class Solution {

    public static void main(String[] args) throws IOException {
        BufferedReader bufferedReader = new BufferedReader(new
        InputStreamReader(System.in));
        BufferedWriter bufferedWriter = new BufferedWriter(new
        FileWriter(System.getenv("OUTPUT_PATH")));

        String[] firstMultipleInput =
        bufferedReader.readLine().replaceAll("\\s+$", "").split(" ");

        int n = Integer.parseInt(firstMultipleInput[0]);

        int k = Integer.parseInt(firstMultipleInput[1]);

        String s = bufferedReader.readLine();

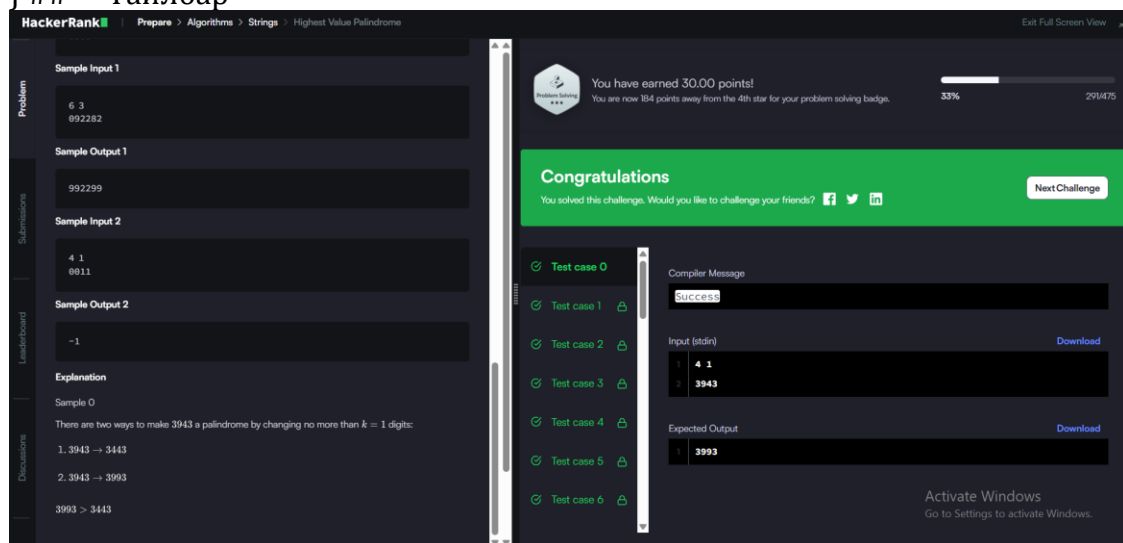
        String result = Result.highestValuePalindrome(s, n, k);

        bufferedWriter.write(result);
        bufferedWriter.newLine();

        bufferedReader.close();
        bufferedWriter.close();
    }
}

```

} ## —Тайлбар—



Энэ

програмын зорилго нь өгөгдсөн тоон тэмдэгт мөрийг хамгийн том утгатай палиндром болгох явдал юм. Тухайн тэмдэгт мөрийг зөвхөн хязгаарлагдмал тооны удаа өөрчилж болно. Хэрэв палиндром болгоход шаардлагатай өөрчлөлт нь зөвшөөрөгдсөн хязгаараас хэтэрвэл, ямар ч өөрчлөлт хийж чадахгүй тул “-1” гэсэн хариуг буцаадаг.

Програм эхлээд тэмдэгт мөрийг тэмдэгт бүрт задлан боловсруулж, зүүн ба баруун талаас зэрэг шалгадаг. Хоёр талаасаа ялгаатай тоонууд байвал, илүү том утгатайг нь аль аль талд нь бичиж палиндром болгоход шаардлагатай өөрчлөлтийг тоолдог.

Хэрвээ энэ эхний алхамд зөвшөөрсөн өөрчлөлтийн хэмжээнээс давсан бол шууд “боломжгүй” гэж үзээд дуусдаг. Хэрвээ хангалттай өөрчлөлт хийх боломж байвал, үлдсэн өөрчлөлтийн эрхийг ашиглан палиндромын дотоод утгыг аль болох “9” болгож ихэсгэхийг зорьдог. Ялангуяа хоёр талаасаа хоёуланг нь өөрчлөх боломжтой тохиолдолд, хоёр өөрчлөлтөөр нэг хосыг “9” болгож чаддаг.

Хэрвээ яг голд нэг орон тоо байгаа бол үлдсэн өөрчлөлтийн эрх байвал, тэр оронг мөн “9” болгоно. Эцэст нь, бүх өөрчлөлтүүдийг хийсний дараа хамгийн их утгатай палиндромыг бүтээж буцаадаг.

File: ./algorithm/med/3/README.md

Pairs

—Асуулт—“english” : Given an array of integers and a target value, determine the number of pairs of array elements that have a difference equal to the target value.

Example

There are three values that differ by : , , and . Return .

Function Description

Complete the pairs function below.

pairs has the following parameter(s):

int k: an integer, the target difference int arr[n]: an array of integers Returns

int: the number of pairs that satisfy the criterion Input Format

The first line contains two space-separated integers and , the size of and the target value. The second line contains space-separated integers of the array .

Constraints

each integer will be unique Sample Input

STDIN	Function
5 2	arr[] size n = 5, k =2
1 5 3 4 2	arr = [1, 5, 3, 4, 2]

Sample Output t

3 Explanation

There are 3 pairs of integers in the set with a difference of 2: [5,3], [4,2] and [3,1]. .

—Асуулт—“Монголоор” : Өгөгдсөн бүхэл тоонуудын массив болон зорилтот утгыг ашиглан, ялгаа нь тухайн зорилтот утгатай тэнцүү байх массив дахь хосуудын тоог тодорхойл. Жишээ: Жишээ нь, доорх массив дахь 3 утга дараах ялгаатай байна: 5 ба 3, 4 ба 2, 3 ба 1. Тэгэхээр хариу нь 3 байна. Функцийн тодорхойлолт: pairs функц нь дараах параметруудтэй байна: int k: нэг бүхэл тоо, зорилтот ялгааны утга int arr[n]: бүхэл тоонуудын массив Буцаах утга: int: өгөгдсөн шалгуурыг хангаж буй хосуудын тоо Оролтын формат: Эхний мөр нь хоёр зайгаар тусгаарлагдсан бүхэл тоо агуулна: массивын хэмжээ n ба ялгааны зорилтот утга k Хоёрдугаар мөр нь n ширхэг бүхэл тоо агуулна — массивын элементүүд Хязгаарлалт: Массив дахь бүхэл тоо бүр давтагдашгүй байна Жишээ оролт: 5 2 1 5 3 4 2 Жишээ гаралт: 3 Тайлбар: Массив дахь 3 хос ялгаа нь 2-той тэнцүү байна. Эдгээр нь: [5, 3] [4, 2] [3, 1] Тиймээс хариу нь 3 юм. —Код—” ” import java.io.; import java.math.; import java.security.; import java.text.; import java.util.; import java.util.concurrent.; import java.util.function.; import java.util.regex.; import java.util.stream.*; import static java.util.stream.Collectors.joining; import static java.util.stream.Collectors.toList;

```
class Result {
```

```
/*
 * Complete the 'pairs' function below.
 *
 * The function is expected to return an INTEGER.
 * The function accepts following parameters:
 * 1. INTEGER k
 * 2. INTEGER_ARRAY arr
 */
```

```
public static int pairs(int k, List<Integer> arr) {
// Write your code here
```

```
Set set = new HashSet<>(); int count = 0; for (int num : arr) { if (set.contains(num - k))
count++; if (set.contains(num + k)) count++; set.add(num); } return count; }
```

```
public class Solution { public static void main(String[] args) throws IOException {
BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(System.in)); BufferedWriter bufferedWriter = new
BufferedWriter(new FileWriter(System.getenv("OUTPUT_PATH")));
```

```
String[] firstMultipleInput =
bufferedReader.readLine().replaceAll("\\s+$", "").split(" ");
```

```
int n = Integer.parseInt(firstMultipleInput[0]);
```

```
int k = Integer.parseInt(firstMultipleInput[1]);
```

```
List<Integer> arr =
```

```

Stream.of(bufferedReader.readLine().replaceAll("\\s+$", "").split(" "))
    .map(Integer::parseInt)
    .collect(toList());

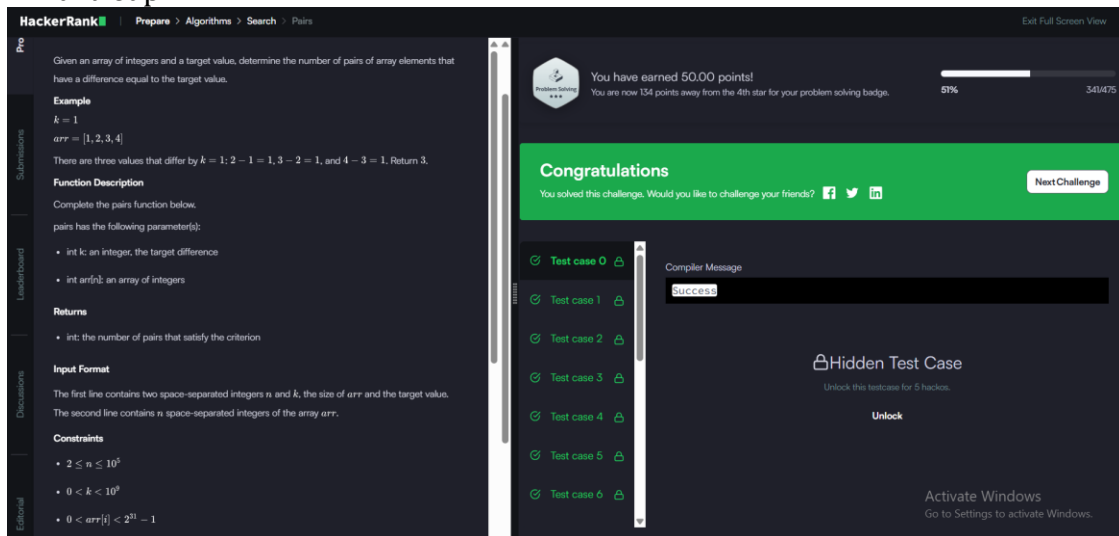
int result = Result.pairs(k, arr);

bufferedWriter.write(String.valueOf(result));
bufferedWriter.newLine();

bufferedReader.close();
bufferedWriter.close();
}
}

```

—Тайлбар—



File: ./algorithm/med/4/README.md

min-penalty-path

—Асуулт—“english” : Consider an undirected graph containing nodes and edges. Each edge has an integer cost, c , associated with it.

The penalty of a path is the bitwise OR of every edge cost in the path between a pair of nodes, and p . In other words, if a path contains edges e_1, e_2, \dots, e_k , then the penalty for this path is $c_{e_1} \text{ OR } c_{e_2} \text{ OR } \dots \text{ OR } c_{e_k}$.

Given a graph and two nodes, s and t , find the path between s and t having the minimal possible penalty and print its penalty; if no such path exists, print -1 to indicate that there is no path from s to t .

Note: Loops and multiple edges are allowed. The bitwise OR operation is known as `or` in Pascal and `|` in C++ and Java.

Input Format

The first line contains two space-separated integers, (the number of nodes) and (the number of edges), respectively.

Each line of the subsequent lines contains three space-separated integers u , v , and w , respectively, describing edge connecting the nodes u and v and its associated penalty w .

The last line contains two space-separated integers, (the starting node) and (the ending node), respectively.

Constraints

Output Format

Print the minimal penalty for the optimal path from node s to node t ; if no path exists from node s to node t , print `-1`.

Sample Input

```
3 4 1 2 1 1 2 1000 2 3 3 1 3 100 1 3
```

Sample Output

3

—Асуулт— “Монголоор” : Хамгийн бага торгуульд замыг олох зорилготойгоор, нодууд ба ирмэгүүдээс тогтсон чиглэлгүй граф өгөгджээ. Граф дахь бүх ирмэг бүр тодорхой бүхэл тоон зардал (торгууль) агуулна. Нэг замын торгууль нь тухайн замд багтсан бүх ирмэгийн торгуулийн битийн “OR” үйлдлийн нийлбэр юм. Өөрөөр хэлбэл, хэрэв зам нь ирмэгүүдийн дарааллаар явбал, түүний торгууль нь $w_1 \text{ OR } w_2 \text{ OR } \dots \text{ OR } w_n$ байна. Танд нэг граф болон хоёр орой (нодууд) өгөгдөх ба, энэ хоёр оройн хоорондын хамгийн бага торгуульд замыг олоорой. Хэрэв ийм зам байхгүй бол `-1` гэж хэвлэнэ үү. Тайлбар: Граф нь давхардсан ирмэг болон өөр дээрээ буцдаг ирмэг (loop) агуулж болно. Bitwise OR үйлдлийг Pascal хэл дээр `or`, харин C++ ба Java дээр `|` гэж бичдэг. Оролтын формат: Эхний мөрөнд хоёр бүхэл тоо: n (нодуудын тоо) болон m (ирмэгийн тоо) Дараагийн m мөр бүрд гурван бүхэл тоо: $u \ v \ w$ — u болон v нодуудыг холбосон ирмэг ба торгууль w Сүүлийн мөрөнд хоёр бүхэл тоо: s болон t — эхлэл ба төгсгөлийн нодууд Хязгаарлалт: Тусгай хязгаарлалт заагаагүй боловч, граф нь өөр дээрээ ирмэгтэй болон олон ирмэгтэй байж болно. Гаралтын формат: Хамгийн бага боломжит торгуульд замыг хэвлэнэ. Хэрвээ зам байхгүй бол `-1` гэж хэвлэнэ үү. Жишээ оролт: 3 4

```
1 2 1
1 2 1000
2 3 3
1 3 100
```

1 3 Жишээ гаралт: 3 Тайлбар: Хамгийн бага торгууль зам нь: $1 \rightarrow 2 \rightarrow 3$ Торгууль нь: $1 \text{ OR } 3 = 3$ Тиймээс хариу нь 3 болно. —Код—”” #include <bits/stdc++.h>

```
using namespace std;
```

```
string ltrim(const string &); string rtrim(const string &); vector split(const string &);
```

```
/ Complete the 'beautifulPath' function below. The function is expected to return an
INTEGER. * The function accepts following parameters: * 1. 2D_INTEGER_ARRAY edges * 2.
INTEGER A * 3. INTEGER B */
```

```
int beautifulPath(vector<vector> edges, int A, int B) { vector<vector<pair<int, int>>>
graph(10e4 + 1); // {node, {child, weight}} vector<vector> seen(10e4 + 1, vector(1024,
false)); // {node, {with what weight we came}} for (vector &edge : edges) {
graph[edge[0]].push_back({edge[1], edge[2]}); graph[edge[1]].push_back({edge[0],
edge[2]}); } int penaltyCost = INT_MAX; queue<vector> todo; // {weight, node} minHeap
todo.push({0, A}); seen[A][0] = true; // Assume we reached node A with 0 total weight
while (!todo.empty()) { vector cur = todo.front(); todo.pop(); // If we reach target, save min
penalty if (cur[1] == B) { penaltyCost = min(penaltyCost, cur[0]); } for (pair<int, int> &child
: graph[cur[1]]) { if (seen[child.first][cur[0] | child.second] == false) { // Update new
weight by OR'ing with parent's weight todo.push({cur[0] | child.second, child.first});
seen[child.first][cur[0] | child.second] = true; } } } return (penaltyCost == INT_MAX ? -1 :
penaltyCost); }
```

```
int main() { ofstream fout(getenv("OUTPUT_PATH"));
```

```
string first_multiple_input_temp;
getline(cin, first_multiple_input_temp);
```

```
vector<string> first_multiple_input =
split(rtrim(first_multiple_input_temp));
```

```
int n = stoi(first_multiple_input[0]);
```

```
int m = stoi(first_multiple_input[1]);
```

```
vector<vector<int>> edges(m);
```

```
for (int i = 0; i < m; i++) {
    edges[i].resize(3);
```

```
    string edges_row_temp_temp;
    getline(cin, edges_row_temp_temp);
```

```
    vector<string> edges_row_temp = split(rtrim(edges_row_temp_temp));
```

```
    for (int j = 0; j < 3; j++) {
        int edges_row_item = stoi(edges_row_temp[j]);
```



```

        edges[i][j] = edges_row_item;
    }
}

string second_multiple_input_temp;
getline(cin, second_multiple_input_temp);

vector<string> second_multiple_input =
split(rtrim(second_multiple_input_temp));

int A = stoi(second_multiple_input[0]);
int B = stoi(second_multiple_input[1]);

int result = beautifulPath(edges, A, B);

fout << result << "\n";

fout.close();

return 0;
}

string ltrim(const string &str) { string s(str);
    s.erase(
        s.begin(),
        find_if(s.begin(), s.end(), not1(ptr_fun<int, int>(isspace)))
    );
    return s;
}

string rtrim(const string &str) { string s(str);
    s.erase(
        find_if(s.rbegin(), s.rend(), not1(ptr_fun<int, int>(isspace))).base(),
        s.end()
    );
    return s;
}

vector split(const string &str) { vector tokens;

string::size_type start = 0;
string::size_type end = 0;

```

```

while ((end = str.find(" ", start)) != string::npos) {
    tokens.push_back(str.substr(start, end - start));

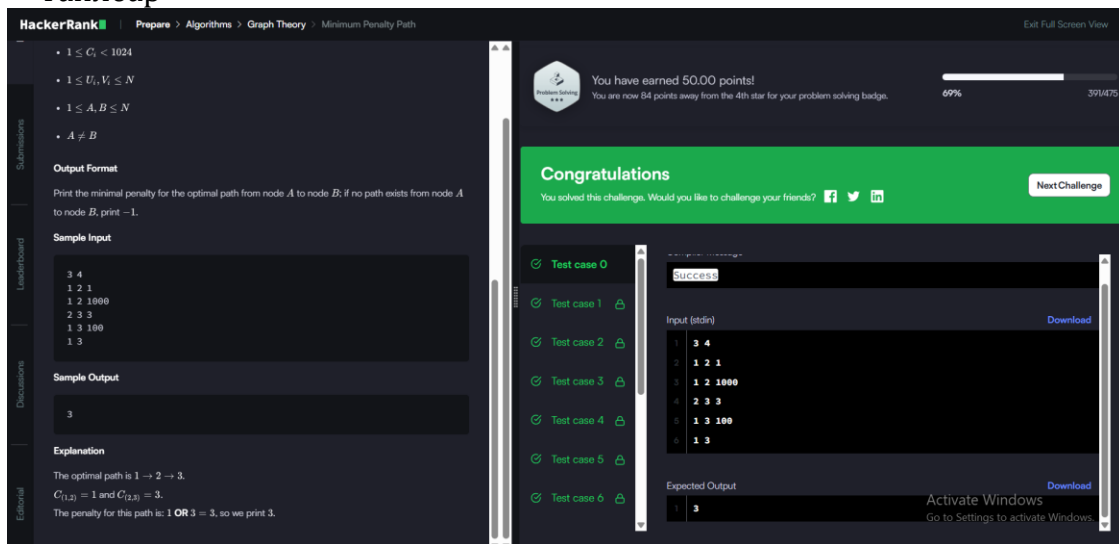
    start = end + 1;
}

tokens.push_back(str.substr(start));

return tokens;
}

```

—Тайлбар—



File: ./algorithm/med/5/README.md

lily

—Асуулт—“english” : Whenever George asks Lily to hang out, she’s busy doing homework. George wants to help her finish it faster, but he’s in over his head! Can you help George understand Lily’s homework so she can hang out with him?

Consider an array of distinct integers, . George can swap any two elements of the array any number of times. An array is beautiful if the sum of among is minimal.

Given the array , determine and return the minimum number of swaps that should be performed in order to make the array beautiful.

Example

One minimal array is . To get there, George performed the following swaps:

Swap	Result
	[7, 15, 12, 3]
3 7	[3, 15, 12, 7]
7 15	[3, 7, 12, 15]

It took swaps to make the array beautiful. This is minimal among the choices of beautiful arrays possible.

Function Description

Complete the `lilysHomework` function in the editor below.

`lilysHomework` has the following parameter(s):

`int arr[n]`: an integer array Returns

`int`: the minimum number of swaps required Input Format

The first line contains a single integer, `n`, the number of elements in `arr`. The second line contains space-separated integers, `arr[0] arr[1] ... arr[n-1]`.

Constraints

Sample Input

STDIN	Function
4	<code>arr</code> size <code>n = 4</code>
2 5 3 1	<code>arr = [2, 5, 3, 1]</code>

Sample Output

2 Explanation

Define `beautiful` to be the beautiful reordering of `arr`. The sum of the absolute values of differences between its adjacent elements is minimal among all permutations and only two swaps (`arr[0]` with `arr[2]` and then with `arr[1]`) were performed.

—Асуулт—“Монголоор” : Жорж Лилигээ хамт тоглохыг урих бүрд Лили гэрийн даалгавар хийж завгүй байдаг. Жорж түүнд хурдан дуусгахад нь туслахыг хүсдэг ч даалгавар нь түүний ойлгож чадахааргүй хэцүү байна! Та Жоржид Лилигийн даалгаврыг ойлгуулахад нь тусалж чадах уу, ингэснээр Лили Жоржтой хамт тоглож чадна.

Бодлого:

`arr` гэж нэрлэгдэх бүхэл тоонуудын дахин давтагдаагүй массив өгөгдсөн гэж үзье. Жорж энэ массив дахь аль ч хоёр элементийг дурын тооны удаа сольж болно. Массив “үзэсгэлэнтэй” байх нь дараах нөхцөлтэй: массив дахь хөрш элементүүдийн ялгааны абсолют утгуудын нийлбэр хамгийн бага байх. Танд өгөгдсөн `arr` массивыг ашиглан, массивыг үзэсгэлэнтэй болгохын тулд хийгдэх ёстой хамгийн бага солилцоо (`swap`)-ны тоог олоорой. Жишээ: Нэг боломжит үзэсгэлэнтэй массив бол `[3, 7, 12, 15]` юм.

Үүнийг бүтээхийн тулд Жорж дараах солилцоог хийжээ: Солилцоо Үр дүн 3 ↔ 7 [3, 15, 12, 7] 7 ↔ 15 [3, 7, 12, 15] Ингээд нийт 2 солилцоо хийсэн бөгөөд энэ нь боломжит үзэсгэлэнтэй массивууд дундаас хамгийн бага нь юм. Функцийн тодорхойлолт: Параметр: `int arr[n]` — бүхэл тоонуудын массив Буцаах утга: `int` — массивыг үзэсгэлэнтэй болгохын тулд шаардлагатай хамгийн бага солилцооны тоо Оролтын формат: Эхний мөрөнд нэг бүхэл тоо `n` — массивын хэмжээ Хоёр дахь мөрөнд `n` ширхэг зайгаар тусгаарлагдсан бүхэл тоо `arr` өгөгдөнө Хязгаарлалт: Массивын бүх элементүүд давтагдахгүй Жишээ оролт: 4 2 5 3 1 Жишээ гаралт: 2 Тайлбар: `arr` массивын үзэсгэлэнтэй хувилбар бол 1 2 3 5 юм. Хөрш элементүүдийн ялгаанууд: $|2-1| + |3-2| + |5-3| = 1 + 1 + 2 = 4$ Энэ нь боломжит хамгийн бага утгуудын нэг бөгөөд, энэ хувилбар руу хүрэхийн тулд ердөө 2 солилцоо хийсэн (Жишээлбэл: 5 ↔ 1, дараа нь 2 ↔ 3 гэх мэт). —Код—`“java8” import java.io.; import java.util.; import java.util.stream.*;`

```
class Result {
public static int lilysHomework(int arr[], int arr2[])
{
    boolean isIdentical = true;
    Map<Integer, Integer> locations = new HashMap<Integer, Integer>();
    for (int i = 0; i < arr.length; i++) {
        locations.put(arr[i], i);
        if (arr[i]!=arr2[i])
        {
            isIdentical = false;
        }
    }
    int swapCount=0;
    if (isIdentical)
    {
        return swapCount;
    }
    for (int i=0; i<arr.length; i++)
    {
        if (arr[i]!=arr2[i])
        {
            int arrIndex = locations.get(arr2[i]);
            locations.put(arr[i], arrIndex);

            int temp = arr[i];
            arr[i]= arr2[arrIndex];
            arr[arrIndex] = temp;

            swapCount++;
        }
    }
    return swapCount;
}
```

```
}
```

```
public class Solution { public static void main(String[] args) throws IOException { Scanner  
scn = new Scanner(System.in); int numberOfElements = scn.nextInt();
```

```
    int[] values = new int[numberOfElements];  
    for (int i = 0; i < numberOfElements; i++) {  
        int value = scn.nextInt();  
        values[i] = value;  
    }  
    int valuesCopy[] = values.clone();  
    int sortedArray[] = values.clone();
```

```
    Arrays.sort(sortedArray);
```

```
    int[] reverseSortedArray = IntStream.range(0, sortedArray.length)  
        .map(i -> sortedArray[sortedArray.length-i-1])  
        .toArray();
```

```
    int regArraySwapCount = Result.lilysHomework(values, sortedArray);
```

```
    int reverseArraySwapCount = Result.lilysHomework(valuesCopy,  
reverseSortedArray);
```

```
    int result = Math.min(regArraySwapCount, reverseArraySwapCount);  
    System.out.println(result);
```

```
}
```

—Тайлбар—

The screenshot displays the HackerRank interface for the 'Lily's Homework' problem. The left sidebar contains navigation links: Problem, Submissions, Leaderboard, and Discussions. The main content area on the left provides the problem description, an example input/output, and the function signature: `lilysHomework(int arr[])`. The right sidebar shows a 'Congratulations' message, a list of test cases (all passed), the compiler message 'Success', and the input/output for the first test case. The top right corner indicates the user has earned 40.00 points and is 44 points away from the 4th star.

HackerRank Prepare > Algorithms > Sorting > Lily's Homework Exit Full Screen View

Whenever George asks Lily to hang out, she's busy doing homework. George wants to help her finish it faster, but he's in over his head! Can you help George understand Lily's homework so she can hang out with him?

Consider an array of n distinct integers, $arr = [a[0], a[1], \dots, a[n-1]]$. George can swap any two elements of the array any number of times. An array is beautiful if the sum of $|arr[i] - arr[i-1]|$ among $0 < i < n$ is minimal.

Given the array arr , determine and return the minimum number of swaps that should be performed in order to make the array beautiful.

Example

$arr = [7, 15, 12, 3]$

One minimal array is $[3, 7, 12, 15]$. To get there, George performed the following swaps:

Swap	Result
[7, 15, 12, 3]	
3 7	[3, 15, 12, 7]
7 15	[3, 7, 12, 15]

It took 2 swaps to make the array beautiful. This is minimal among the choices of beautiful arrays possible.

Function Description

Complete the `lilysHomework` function in the editor below.

`lilysHomework` has the following parameter(s):

- `int arr[]`: an integer array

Returns

- `int`: the minimum number of swaps required

You have earned 40.00 points!
You are now 44 points away from the 4th star for your problem solving badges. 84% 43/475

Congratulations
You solved this challenge. Would you like to challenge your friends? [f](#) [t](#) [in](#) [Next Challenge](#)

Test case 0 [^](#)
Test case 1 [^](#)
Test case 2 [^](#)
Test case 3 [^](#)
Test case 4 [^](#)
Test case 5 [^](#)
Test case 6 [^](#)

Compiler Message
Success

Input (stdin)
4
2 5 3 1 [Download](#)

Expected Output
2 [Download](#)

Activate Windows
Go to Settings to activate Windows.

File: ./algorithm/med/bodlogo1/README.md

ПХ. 40 бодлого

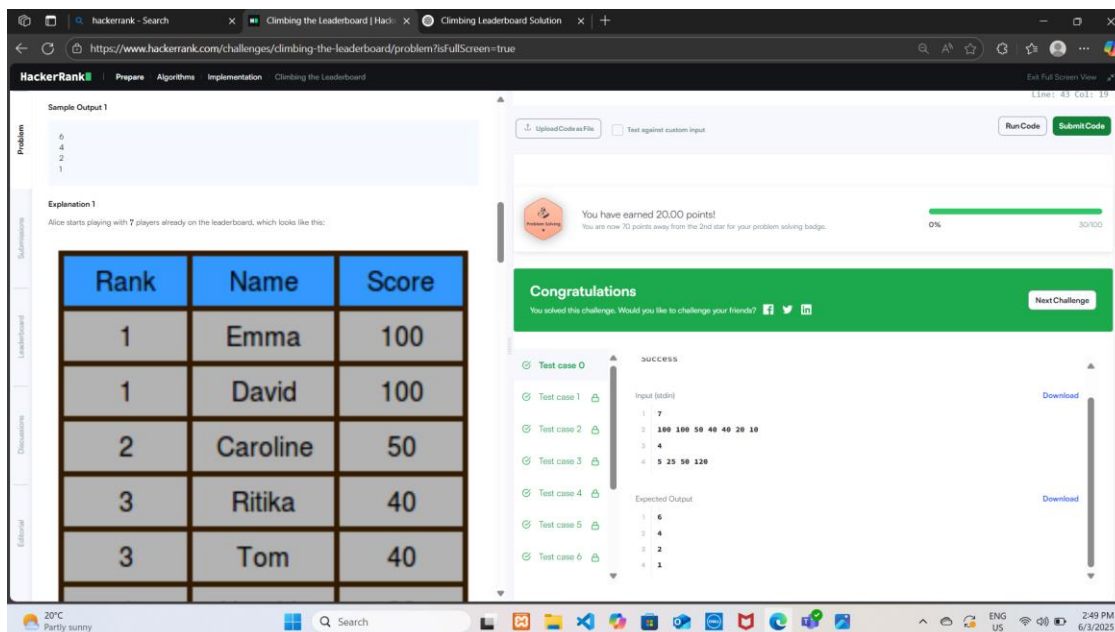
Бодлого 1. Climbing Leader Board тоглогч тус бүр лидербордын оргилд хүрэхийг тулд, тоглоомын дараах үнэлгээг дагаж, өөрийн байр сууриа хянадаг. Энэхүү тоглоом (Dense Ranking)-ийг ашигладаг. - Хамгийн өндөр оноотой тоглогч 1-р байранд орно. - Ижил оноотой тоглогчид ижил зэрэглэл авна, дараагийн тоглогч нь шууд дараагийн зэрэглэл авна. Жишээ: ranked = [100, 90, 90, 80] player = [70, 80, 105] Лидерборд дараах байдлаар эрэмбэлэгдэнэ: 100 (1-р байр), 90 (2-р байр), 90 (2-р байр), 80 (3-р байр) Хэрвээ тоглогчийн оноо дараах байдалтай байвал: - 70 → 4-р байр - 80 → 3-р байр - 105 → 1-р байр Буцаах ёстой үр дүн: [4, 3, 1]

Функцийн тайлбар Доорх climbingLeaderboard функцыг гүйцээн хэрэгжүүлнэ үү. climbingLeaderboard функц дараах параметруудтэй: int ranked[n]: лидерборд дээрх оноонууд (ихээс бага руу эрэмбэлэгдсэн) int player[m]: тоглогчийн тоглолт бүрийн дараах оноонууд

Буцаах утга int[m]: тоглогчийн оноо бүрийн дараа авсан байрлалуудын жагсаалт

Оролтын формат Эхний мөрөнд n бүхэл тоо өгөгдөнө — лидерборд дээрх тоглогчдын тоо. Дараагийн мөрөнд хоорондоо зайгаар тусгаарлагдсан n ширхэг бүхэл тоо өгөгдөнө — лидербордын оноонууд (ranked[]), буурахаар эрэмбэлэгдсэн байна. Гурав дахь мөрөнд m бүхэл тоо өгөгдөнө — тоглогчийн тоглосон тоглоомуудын тоо. Сүүлийн мөрөнд хоорондоо зайгаар тусгаарлагдсан m ширхэг бүхэл тоо өгөгдөнө — тоглогчийн оноонууд (player[]), өсөхөөр эрэмбэлэгдсэн байна.

Хязгаарлалт $1 < n < 2 \cdot 10^5$ $1 < m < 2 \cdot 10^5$ $0 < \text{ranked}[i] < 10^9$ эсвэл $0 < i < n$ буурахаар эрэмбэлсэн байна. $0 < \text{player}[j] < 10^9$ эсвэл $0 < j < m$ өсөхөөр эрэмбэлсэн байна.



Үр дүн:

холбоос: <https://www.hackerrank.com/challenges/climbing-the-leaderboard/problem?isFullScreen=true>

File: ./algorithm/med/bodlogo2/README.md

Compare the Triplets

Алис болон Боб хоёр гурван төрлийн шалгуураар оноо авдаг. Тэдгээр шалгуур бүрт 0-оос 100 хүртэлх бүхэл тоон оноо өгдөг. Бид эдгээр шалгуурт оноо өгөхдөө дараах байдлаар харьцуулалт хийнэ:

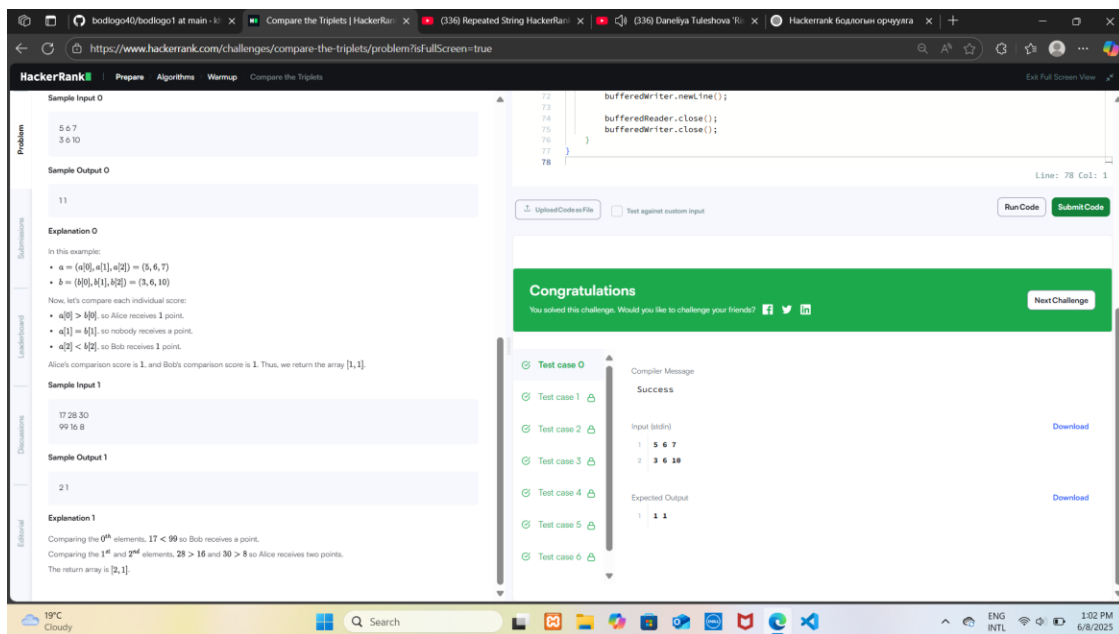
- Хэрэв Алисын оноо тухайн шалгуур дээр Бобын онооноос их байвал Алис 1 оноо авна.
- Хэрэв Бобын оноо тухайн шалгуур дээр Алисын онооноос их байвал Боб 1 оноо авна.
- Хэрэв хоёул адил оноо авсан бол хэн ч оноо авахгүй.

Алис болон Бобын оноог гурван шалгуур тус бүрээр өгсөн гэж үзье. Эдгээр оноонуудыг харьцуулж, тус бүрийн авсан оноог дараалалтайгаар буцаан.

Оролт: - Эхний мөрөнд гурван бүхэл тоо (Алисын оноо): $a[0]$, $a[1]$, $a[2]$ - Дараагийн мөрөнд гурван бүхэл тоо (Бобын оноо): $b[0]$, $b[1]$, $b[2]$

Гаралт: - Хоёр бүхэл тоо бүхий массив/жагсаалт буцаана: [Алисын оноо, Бобын оноо]

Үр дүн:



Screenshot (355)

Бодогын холбоос: <https://www.hackerrank.com/challenges/compare-the-triplets/problem?isFullScreen=true>

File: [./algorithm/med/bodlogo3/README.md](#)

Бодлогын нэр: Extra Long Factorials

n бүхэл тооны факториал буюу $n!$ нь дараах томъёогоор тодорхойлогдоно:

$$n! = n \times (n-1) \times (n-2) \times \dots \times 2 \times 1$$

Бүтэл тооны факториалыг тооцоолон хэвлэ.

Функцийн тодорхойлолт: `extraLongFactorials` гэдэг функцыг доорх загварын дагуу гүйцээнэ. Энэ нь зөвхөн үр дүнг хэвлэх (`print`) үүрэгтэй бөгөөд утга буцаах шаардлагагүй.

n : нэг бүхэл тоо

Тайлбар: Факториалын утга нь маш хурдан томорч, `long long` (C/C++ дахь 64-bit) хувьсагч ч хадгалах боломжгүй болж болзошгүй. Иймээс `BigInteger` гэх мэт том тоог дэмждэг төрлүүдийг ашиглах шаардлагатай. Java, Python, Ruby гэх мэт хэлнүүдэд энэ боломж байдаг бол C/C++ дээр тусгай аргаар зохицуулах хэрэгтэй болдог.

Оролт формат Оролт нь ганц бүхэл тоо n байна.

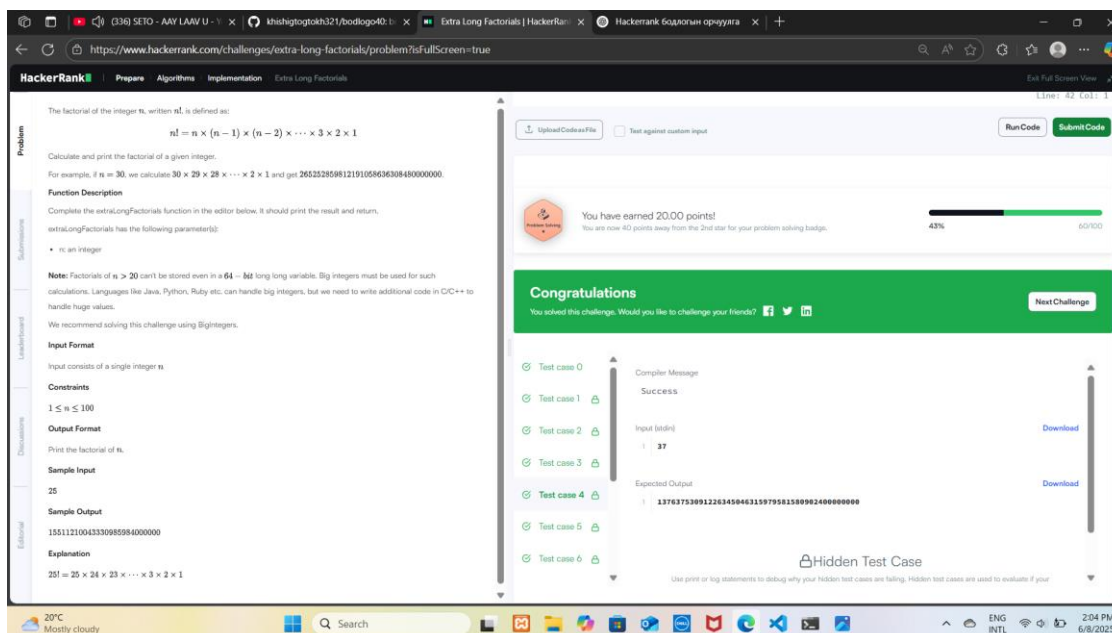
Хязгаарлалт $1 \leq n \leq 1000$

Гаралт формат $n!$ буюу n тооны факториалын утгыг ганц мөрөнд хэвлэнэ.

Жишээ оролт: 25

Жишээ гаралт: 115511210043330985984000000

Тайлбар: $25! = 25 \times 24 \times 23 \times \dots \times 2 \times 1$



Үр дүн:

Бодлогын холбоос: <https://www.hackerrank.com/challenges/extra-long-factorials/problem>

File: `./algorithm/med/bodlogo4/README.md`

Бодлогын нэр: Non-Divisible Subset

Массив S ялгаатай бүхэл тоонууд өгөгдсөн, дараах нөхцөлийг хангах хамгийн их хэмжээтэй дэд массив элементүүдийг олж хэмжээг хэвлэ. Тухайн дэд массив элементүүдийн хоёр тооны нийлбэр k тоонд хуваагдахгүй байх ёстой.

Жишээ:

$k = 4$ $S = [19, 10, 12, 10, 24, 25, 22]$

Үүсгэж болох зарим багц:

$S'[0] = [10, 12, 25]$

$S'[1] = [19, 22, 24]$

Бүх боломжит хувилбаруудыг шалгасны дараа хамгийн урт зөв багц нь 3 элементтэй байна.

Функцийн тайлбар Функцийн нэр: nonDivisibleSubset

Оролт параметрууд: int S[n]: бүхэл тоонуудын массив

int k: хуваагч тоо

Буцах утга: int: дээрх нөхцөлийг хангах S массивын хамгийн олон элементтэй дэд массив элементүүд хэмжээ.

Оролтын формат: 1-р мөр: n ба k гэсэн 2 бүхэл тоо (хоорондоо зайгаар тусгаарлагдсан) 2-р мөр: n ширхэг бүхэл тоо, тус бүр нь S[i] утга

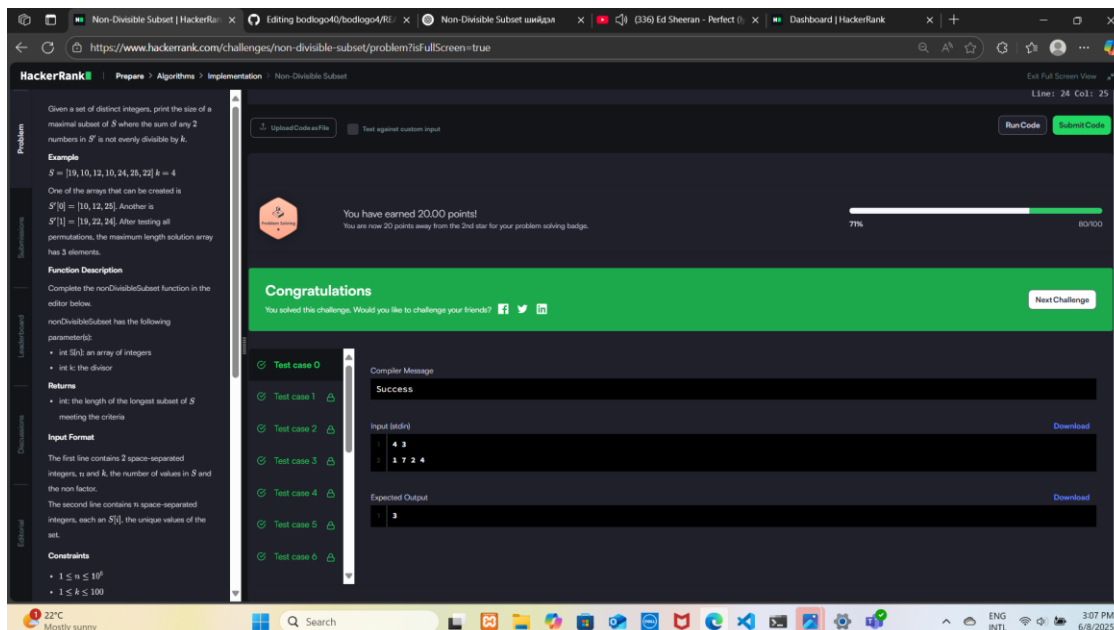
Хязгаарлалт: $1 \leq n \leq 10^5$ $1 \leq k \leq 100$ $1 \leq S[i] \leq 10^9$

Бүх өгөгдсөн тоонууд давхардалгүй байна

Жишээ оролт: 4 3 S[] size n = 4, k = 3

1 7 2 4 S = [1, 7, 2, 4]

Жишээ гаралт: 3



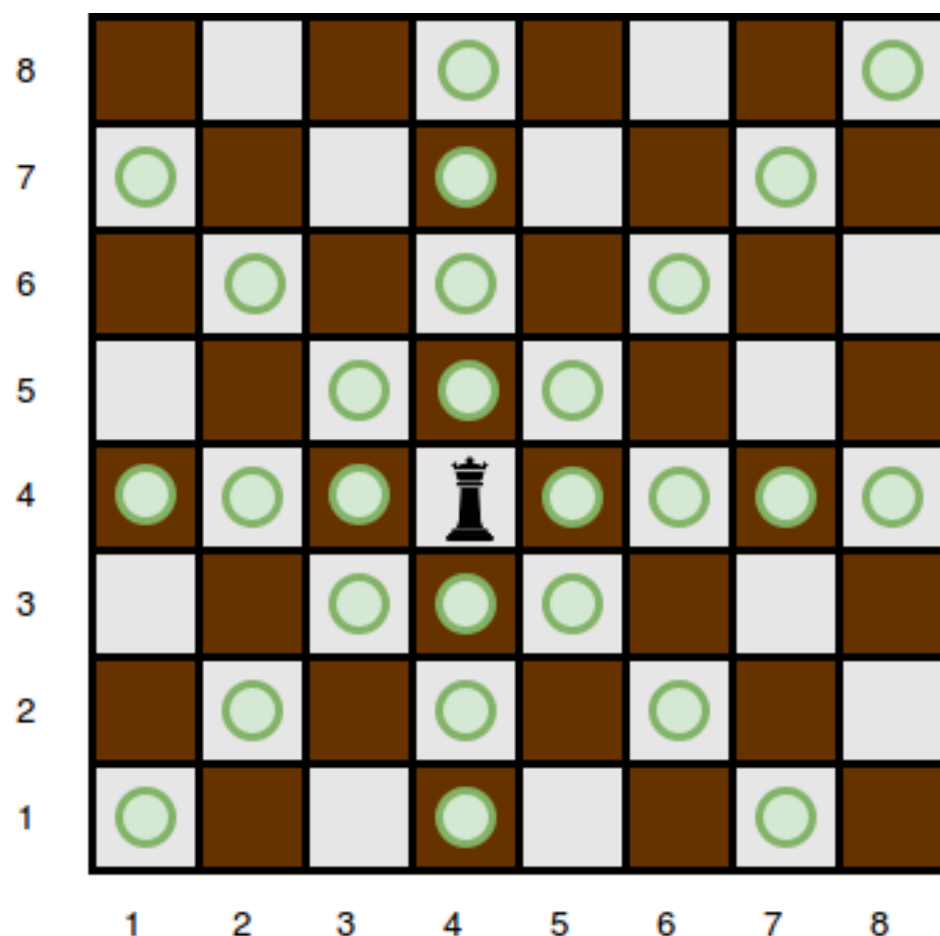
Үр дүн:

Бодлогын холбоос: <https://www.hackerrank.com/challenges/non-divisible-subset/problem>

File: ./algorithm/med/bodlogo5/README.md

Queen's Attack II

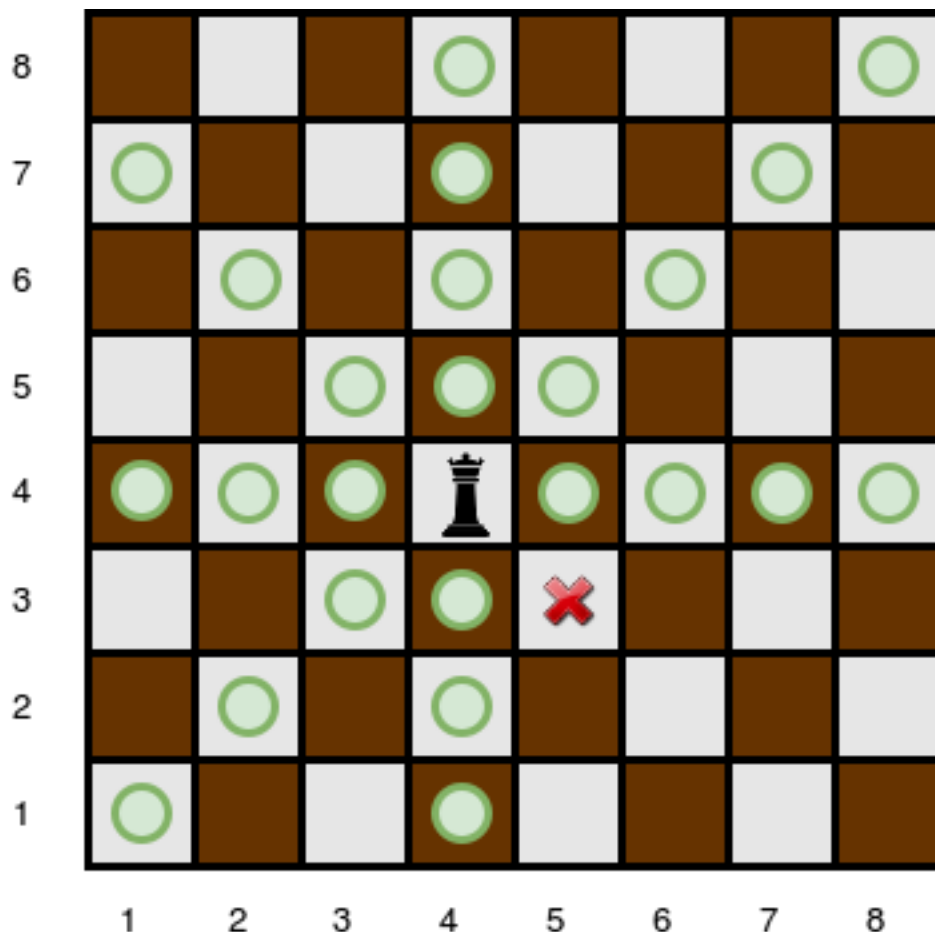
Танд хатан болон хэд хэдэн саад байрлуулсан квадрат шатрын самбар өгөгдөнө. Хатан хэдэн нүд рүү довтолж чадахыг тодорхойл. Хатан $n \times n$ хэмжээтэй шатрын самбар дээр байрласан байна. Самбарын мөрүүд нь доороос дээш чиглэлтэйгээр 1-ээс n хүртэл дугаарлагддаг. Баганууд нь зүүнээс баруун тийш чиглэлтэйгээр мөн 1-ээс n хүртэл дугаарлагддаг. Самбарын аль ч нүдийг (r, c) хосоор илэрхийлэх ба энэ нь тухайн нүдний мөр r , багана c -г илтгэнэ. Хатан (r_q, c_q) байрлалд байрласан. Нэг удаагийн нүүдлээрээ хатан дараах найман чиглэлд байгаа дурын нүд рүү довтолж чадна: зүүн, баруун, дээш, доош, мөн дөрвөн диагональ чиглэл. Доорх зурган дээр (4, 4) байрлалд байгаа хатан ямар нүднүүд рүү довтолж чадахыг ногоон дугуйгаар тэмдэглэсэн байна:



1485426500-a4039ebb00-chess1

Самбар дээр сааднууд байрлах ба эдгээр нь тухайн чиглэлд саадаас цааш байрлах нүднүүд рүү хатан довтолохоос сэргийлнэ. Жишээлбэл, дээрх зурган дээрх (3, 5)

байрлалд байгаа саад нь хатанг дараах нүднүүд рүү довтолохоос сэргийлнэ: (3, 5), (2, 6), болон (1, 7).



image

Хатангийн байрлал болон самбар дээрх бүх саадны байршлууд өгөгдсөн нөхцөлд, хатан (r_q, c_q) байрлалаас хэдэн нүд рүү довтолох боломжтойг олж, хэвлэ.

Доорх queensAttack нэртэй функцийг гүйцээх шаардлагатай.

Функцийн тайлбар:

- int n, самбарын хэмжээ (n x n)
- int k, саадны тоо
- int r_q, int c_q, хатангийн мөр ба баганын дугаар
- int obstacles[k][2] k ширхэг саадны байршил (мөр, багана)

Буцаах утга: - Хатангийн довтолж чадах нүдний тоо (int төрөлтэй).

Оролтын формат: Эхний мөрөнд хоёр бүхэл тоо: n (самбарын хэмжээ), k (саадны тоо).

Дараагийн мөрөнд хатангийн байрлалын координат: r_q (мөр), c_q (багана).

Дараагийн k мөр бүрт нэг саадны байрлал өгөгдөнө: $r[i]$ ба $c[i]$.

Хязгаарлалт: $0 < n \leq 100,000$ $0 \leq k \leq 100,000$

Нэг нүдэнд олон саад байж болно.

Хатан байрлаж буй нүдэнд хэзээ ч саад байхгүй. Дэд даалгаврууд: 30% оноо авах нөхцөлд:

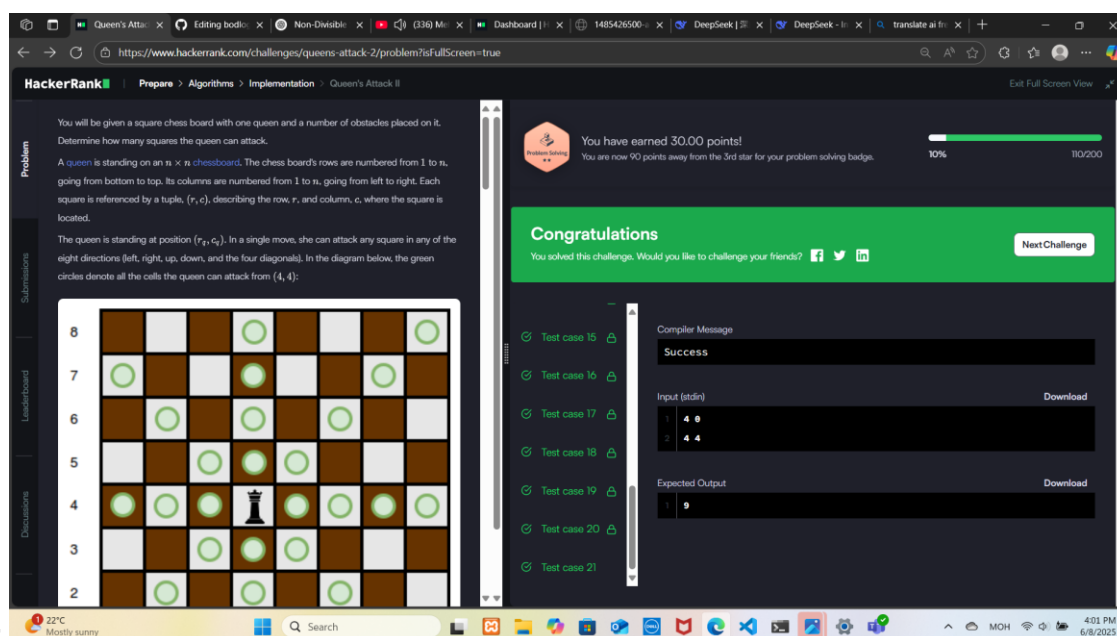
$0 < n \leq 100$ $0 \leq k \leq 100$

Нийт 55% оноо авах нөхцөлд:

$0 < n \leq 1000$ $0 \leq k \leq 10^5$

Оролт: 40 44

Гаралт: 9



Үр дүн:

Бодлогийн холбоос: <https://www.hackerrank.com/challenges/queens-attack-2/problem>

File: ./algorithm/med/min-loss/README.md

min-loss

—Асуулт—“english” :

Lauren has a chart of distinct projected prices for a house over the next several years. She must buy the house in one year and sell it in another, and she must do so at a loss. She wants to minimize her financial loss. Example Her minimum loss is incurred by purchasing in year at and reselling in year at . Return . Function Description Complete the minimumLoss function in the editor below. minimumLoss has the following parameter(s):
int price[n]: home prices at each year
Returns int: the minimum loss possible
Input Format
The first line contains an integer , the number of years of house data. The second line contains space-separated long integers that describe each . Constraints All the prices are distinct. A valid answer exists. Subtasks for of the maximum score. Sample Input 0 3 5 10 3
Sample Output 0 2 Explanation 0 Lauren buys the house in year at and sells it in year at for a minimal loss of . Sample Input 1 5 20 7 8 2 5
Sample Output 1 2 Explanation 1 Lauren buys the house in year at and sells it in year at for a minimal loss of .

#—Асуулт—“Монголоор” : Лаурен ирэх хэдэн жилд орон сууцны үнийн төсөөллийг харуулсан хүснэгттэй. Тэр орон сууцыг нэг жилд худалдаж аваад, өөр жилд заавал алдагдалтайгаар зарах ёстой. Түүний зорилго бол хамгийн бага алдагдал хүлээх явдал юм.

Жишээ:

Тэр хамгийн бага алдагдалтай байх тохиолдол нь: Нэг жилд 5₮-өөр худалдаж аваад, дараа жилд 3₮-өөр зарсан. Энэ тохиолдолд түүний алдагдал 2₮ байна. Параметрууд: price[n]: Жил бүрийн орон сууцны үнэ Буцаах утга: int: Боломжит хамгийн бага алдагдлын хэмжээ

Оролтын формат:

Эхний мөрөнд n — орон сууцны үнийн жилүүдийн тоо байна.

Хоёр дахь мөрөнд n ширхэг зайгаар тусгаарлагдсан бүхэл тоо байна — тус бүр тухайн жилд орон сууц хэдэн төгрөгөөр байсан.

Нөхцөлүүд:

Бүх үнийн утгууд давтагдахгүй.

Зөв хариу байх нь баталгаатай.

Хэсэг оноо авах дэд даалгавар: $n \leq 10^5$

Жишээ Оролт 0:

3

5 10 3 Жишээ Гаралт 0:

2 Тайлбар 0:

Лаурен 5₮-өөр худалдан авч, 3₮-өөр зарсан. Алдагдал = 2₮.

Жишээ Оролт 1:

5

20 7 8 2 5 Жишээ Гаралт 1:

2 Тайлбар 1:

7₮-өөр авч, 5₮-өөр зарсан — хамгийн бага алдагдалтай хувилбар юм. #—Код—“java”

```
import java.io.*;
```

```
import java.math.*;
```

```
import java.security.*;
```

```
import java.text.*;
```

```
import java.util.*;
```

```
import java.util.concurrent.*;
```

```
import java.util.function.*;
```

```
import java.util.regex.*;
```

```
import java.util.stream.*;
```

```
import static java.util.stream.Collectors.joining; import static  
java.util.stream.Collectors.toList;
```

```
class Result {
```

```
/*
```

```
 * Complete the 'minimumLoss' function below.
```

```
 *
```

```
 * The function is expected to return an INTEGER.
```

```
 * The function accepts LONG_INTEGER_ARRAY price as parameter.
```

```
 */
```

```
public static int minimumLoss(List<Long> price) {
```

```
    // Create a sorted copy of the prices list
```

```
    List<Long> sorted = new ArrayList<>(price);
```

```
    Collections.sort(sorted);
```

```
    // Map to store the original index of each price
```

```
    HashMap<Long, Integer> IndexMap = new HashMap<>();
```

```
    for (int i = 0; i < price.size(); i++) {
```

```

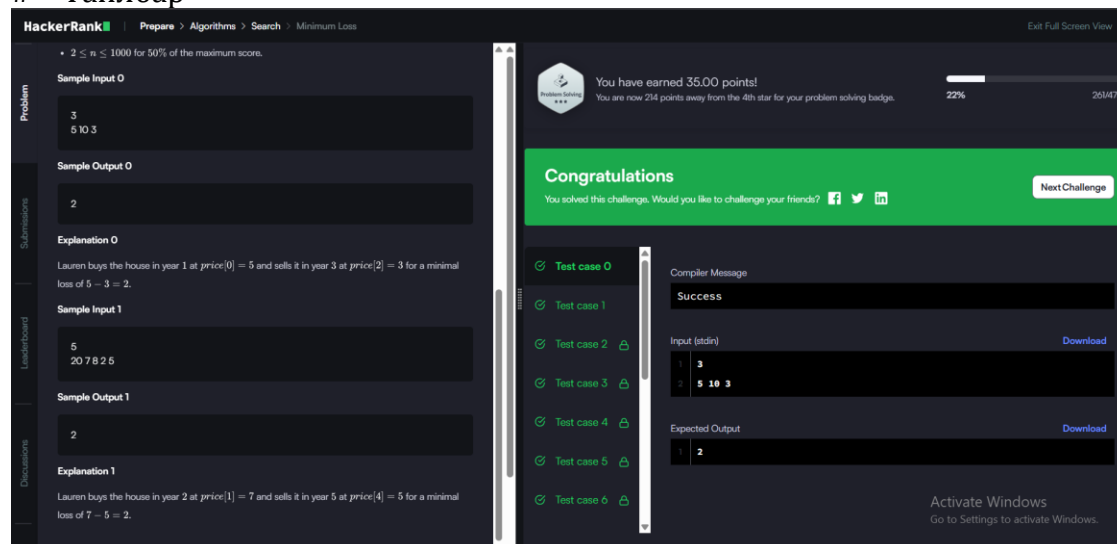
        IndexMap.put(price.get(i), i);
    }
    long minLoss = Long.MAX_VALUE;
    // Iterate through the sorted prices to calculate the minimum loss
    for (int i = 0; i < sorted.size() - 1; i++) {
        Long next = sorted.get(i + 1);
        Long current = sorted.get(i);
        // Check if the "next" price comes after the "current" price in the
        // original list
        if (next - current < minLoss
            && IndexMap.get(next) < IndexMap.get(current)) {
            minLoss = next - current;
        }
    }
    return (int) minLoss;
}} public class Solution {

    public static void main(String[] args) throws IOException {
        BufferedReader bufferedReader = new BufferedReader(new
        InputStreamReader(System.in));
        BufferedWriter bufferedWriter = new BufferedWriter(new
        FileWriter(System.getenv("OUTPUT_PATH")));
        int n = Integer.parseInt(bufferedReader.readLine().trim());
        List<Long> price =
        Stream.of(bufferedReader.readLine().replaceAll("\\s+$", "").split(" "))
            .map(Long::parseLong)
            .collect(toList());

        int result = Result.minimumLoss(price);
        bufferedWriter.write(String.valueOf(result));
        bufferedWriter.newLine();
        bufferedReader.close();
        bufferedWriter.close();
    }
}

```


#—Тайлбар—



Хувилбар үүсгэх: price нэртэй анхны үнийн жагсаалтыг хуулж, sorted нэртэй жагсаалтад хадгална. Дараа нь үүнийг өсөх дарааллаар эрэмбэлнэ.

Индексд хадгалах: IndexMap нэртэй HashMap үүсгээд, анхны жагсаалтын үнэ бүрийн индексд хадгална. Ингэснээр ямар үнэ хэдэн онд байсан бэ гэдгийг санах боломжтой болдог.

Хамгийн бага алдагдал хайх: Эрэмбэлэгдсэн sorted жагсаалтыг ашиглан дараалсан хоёр үнэ (current ба next)-ийн ялгааг тооцно.

Алдагдлын нөхцөлийг шалгах: Хоёр үнэ хоорондын ялгаа (next - current) өмнө олдсон хамгийн бага алдагдлаас бага байж, өндөр үнэ бага үнийн дараа байх ёстой (энэ нь анхны жагсаалтад дарааллаараа байх ёстой). Энэ нь: Лаурен заавал хямд үнээр хожим нь зарж байгаа эсэхийг шалгаж байна гэсэн үг.

Алдагдлыг шинэчлэх: Хэрэв дээрх нөхцөлүүд биелсэн бол minLoss хувьсагчийг шинэ утгаар шинэчилнэ.

Хариу буцаах: Бүх харьцуулалтыг хийсний дараа хамгийн бага алдагдлыг int төрөлд хөрвүүлэн буцаана.

File: ./data_structure/hard/1/README.md

Find the running median

—Асуулт—“english” : The median of a set of integers is the midpoint value of the data set for which an equal number of integers are less than and greater than the value. To find the median, you must first sort your set of integers in non-decreasing order, then:

If your set contains an odd number of elements, the median is the middle element of the sorted sample. In the sorted set, is the median. If your set contains an even number of elements, the median is the average of the two middle elements of the sorted sample. In the sorted set, is the median. Given an input stream of integers, perform the following task for each integer:

Add the integer to a running list of integers. Find the median of the updated list (i.e., for the first element through the element). Print the updated median on a new line. The printed value must be a double-precision number scaled to decimal place (i.e., format). Example

Sorted Median [7] 7.0 [3, 7] 5.0 [3, 5, 7] 5.0 [2, 3, 5, 7] 4.0 Each of the median values is stored in an array and the array is returned for the main function to print.

Note: Add formatting to the print statement.

Function Description Complete the runningMedian function in the editor below.

runningMedian has the following parameters: - int a[n]: an array of integers

Returns - float[n]: the median of the array after each insertion, modify the print statement in main to get proper formatting.

Input Format

The first line contains a single integer, , the number of integers in the data stream. Each line of the subsequent lines contains an integer, , to be inserted into the list.

Constraints

Sample Input

STDIN	Function
6	a[] size n = 6
12	a = [12, 4, 5, 3, 8, 7]
4	
5	
3	
8	
7	

Sample Output

12.0 8.0 5.0 4.5 5.0 6.0 Explanation

There are integers, so we must print the new median on a new line as each integer is added to the list:

—Асуулт—“Монголоор” :

Бүхэл тоонуудын цуглуулгын медиан гэдэг нь тухайн өгөгдлийн дунд хэсгийн утга бөгөөд энэ утгаас бага, их тоонуудын тоо ижил байна. Медианыг олохын тулд эхлээд бүх тоонуудыг өсөх дарааллаар эрэмбэлнэ. Дараа нь:

Хэрвээ таны цуглуулгад сондгой тооны элемент байвал, медиан нь дунд талын элемент байна. Эрэмбэлэгдсэн цуглуулгад тэр нь медиан болно.

Хэрвээ таны цуглуулгад тэгш тооны элемент байвал, медиан нь хоёр дунд талын элементийн дундаж байна. Эрэмбэлэгдсэн цуглуулгад энэ нь медиан болно.

Өгөгдсөн урсгалаар ирэх n ширхэг бүхэл тооны хувьд дараах үйлдлийг бүр тоо дээр гүйцэтгэнэ:

i дахь тоог жагсаалт руу нэмнэ.

Шинэчлэгдсэн жагсаалтын медианыг олно (жишээ нь эхний i тооны хувьд).

Шинэ медианыг шинэ мөрөнд хэвлэнэ. Хэвлэгдэх утга нь нэг оронгийн нарийвчлалтай хөвөгч таслалттай тоо байх ёстой (жишээ нь %.1f формат).

Жишээ:

csharp Copy Edit Эрэмбэлэгдсэн Медиан [7] 7.0 [3, 7] 5.0 [3, 5, 7] 5.0 [2, 3, 5, 7] 4.0
Медиануудын утгыг массивт хадгалж, энэ массивыг үндсэн функц хэвлэнэ. Тайлбар:
Хэвлэлийн мөрөнд формат нэмэх шаардлагатай. Функцийн тодорхойлолт:
runningMedian функцыг дараах байдлаар гүйцээнэ: runningMedian нь дараах параметртэй: int a[n]: бүхэл тоонуудын массив Буцаах утга: float[n]: бүр тоо нэмэгдэх бүрд үүсэх жагсаалтын медиан Оролтын формат: Эхний мөрөнд нэг бүхэл тоо өгөгдөнө — урсгалаар ирэх тоонуудын тоо n. Дараагийн n мөр тус бүрд нэг бүхэл тоо a[i] өгөгдөнө.

Жишээ оролт:

STDIN	Function
6	a[] хэмжээтэй массив (n = 6)
12	a = [12, 4, 5, 3, 8, 7]
4	
5	
3	
8	
7	

Жишээ гаралт:

12.0
8.0

5.0
4.5
5.0
6.0

—Код—” ”

```
#include <bits/stdc++.h>

using namespace std;

string ltrim(const string &); string rtrim(const string &);

/ Complete the 'runningMedian' function below. The function is expected to return a
DOUBLE_ARRAY. * The function accepts INTEGER_ARRAY a as parameter. /vector
runningMedian(vector const&a) { if(0==a.size()) return vector(); map<int,int> aux;
aux.insert({a[0],1}); auto lowerMed = aux.begin(); vector retq; retq.push_back(1.0(lowerMed-
>first)); int pos=0; for(int i=1; i<a.size();i++) { if(!aux.count(a[i])) aux.insert({a[i],1});
else{aux[a[i]]++;} if((a[i]<first)&&(i&1)) pos--; else if((a[i]>=lowerMed-
>first)&&(0==(i&1))) pos++; if(0>pos) { lowerMed--; pos=lowerMed->second-1; } else
if(lowerMed->second<=pos) { lowerMed++; pos=0; } if(0==(i&1)) {
retq.push_back(1.0(lowerMed->first)); } else { auto x=lowerMed; if(pos+1>=lowerMed-
>second) x++; retq.push_back(0.5(lowerMed->first + x->first)); } } return retq; } int main() {
ofstream fout(getenv("OUTPUT_PATH"));

string a_count_temp;
getline(cin, a_count_temp);

int a_count = stoi(ltrim(rtrim(a_count_temp)));

vector<int> a(a_count);

for (int i = 0; i < a_count; i++) {
    string a_item_temp;
    getline(cin, a_item_temp);

    int a_item = stoi(ltrim(rtrim(a_item_temp)));

    a[i] = a_item;
}

vector<double> result = runningMedian(a);

for (size_t i = 0; i < result.size(); i++) {
    fout << result[i];

    if (i != result.size() - 1) {
        fout << "\n";
    }
}
```

```

    }
}

fout << "\n";

fout.close();

return 0;
}

string ltrim(const string &str) { string s(str);
    s.erase(
        s.begin(),
        find_if(s.begin(), s.end(), not1(ptr_fun<int, int>(isspace)))
    );
    return s;
}

string rtrim(const string &str) { string s(str);
    s.erase(
        find_if(s.rbegin(), s.rend(), not1(ptr_fun<int, int>(isspace))).base(),
        s.end()
    );
    return s;
}

```

—Тайлбар—

The screenshot shows the HackerRank interface for the 'Find the Running Median' problem. The left sidebar contains links for 'Submissions', 'Leaderboard', 'Discussions', and 'Editorial'. The main content area displays the problem details:

- STDIN**:
6
12
4
5
3
8
7
- Function**:
a[] size n = 6
a = [12, 4, 5, 3, 8, 7]
- Sample Output**:
12.0
8.0
5.0
4.5
5.0
6.0
- Explanation**:
There are $n = 6$ integers, so we must print the new median on a new line as each integer is added to the list:
1. list = {12}, median = 12.0
2. list = {4, 12}, median = $\frac{4+12}{2} = 8.0$
3. list = {4, 5, 12}, median = 5.0
4. list = {3, 4, 5, 12}, median = $\frac{4+5}{2} = 4.5$
5. list = {3, 4, 5, 8, 12}, median = 5.0
6. list = {3, 4, 5, 7, 8, 12}, median = $\frac{5+7}{2} = 6.0$

On the right, a 'Congratulations' banner states: 'You have earned 50.00 points! You are now 189 points away from the gold level for your problem solving badge. 50% 661/850'. Below this is a 'Next Challenge' button. A list of test cases is shown, with 'Test case 0' selected. The 'Expected Output' for the selected test case is:

```
1 12.0
2 8.0
3 5.0
4 4.5
5 5.0
6 6.0
```

find running median

File: [./data_structure/hard/2/README.md](#)

min average time

—Асуулт—“english” : Tieu owns a pizza restaurant and he manages it in his own way. While in a normal restaurant, a customer is served by following the first-come, first-served rule, Tieu simply minimizes the average waiting time of his customers. So he gets to decide who is served first, regardless of how sooner or later a person comes.

Different kinds of pizzas take different amounts of time to cook. Also, once he starts cooking a pizza, he cannot cook another pizza until the first pizza is completely cooked. Let's say we have three customers who come at time $t=0$, $t=1$, & $t=2$ respectively, and the time needed to cook their pizzas is 3, 9, & 6 respectively. If Tieu applies first-come, first-served rule, then the waiting time of three customers is 3, 11, & 16 respectively. The average waiting time in this case is $(3 + 11 + 16) / 3 = 10$. This is not an optimized solution. After serving the first customer at time $t=3$, Tieu can choose to serve the third customer. In that case, the waiting time will be 3, 7, & 17 respectively. Hence the average waiting time is $(3 + 7 + 17) / 3 = 9$.

Help Tieu achieve the minimum average waiting time. For the sake of simplicity, just find the integer part of the minimum average waiting time.

Input Format

The first line contains an integer N , which is the number of customers. In the next N lines, the i th line contains two space separated numbers T_i and L_i . T_i is the time when i th customer order a pizza, and L_i is the time required to cook that pizza.

The customer is not the customer arriving at the arrival time.

Output Format

Display the integer part of the minimum average waiting time. Constraints

$1 \leq N \leq 105$ $0 \leq T_i \leq 109$ $1 \leq L_i \leq 109$ Note

The waiting time is calculated as the difference between the time a customer orders pizza (the time at which they enter the shop) and the time she is served.

Cook does not know about the future orders.

Sample Input #00

3 0 3 1 9 2 6 Sample Output #00

9 Sample Input #01

3 0 3 1 9 2 5 Sample Output #01

8 Explanation #01

Let's call the person ordering at time = 0 as A, time = 1 as B and time = 2 as C. By delivering pizza for A, C and B we get the minimum average wait time to be

$$(3 + 6 + 16)/3 = 25/3 = 8.33$$

—Асуулт—“Монголоор” : Tieu пиццаны ресторан ажиллуулдаг бөгөөд тэрээр үүнийг өөрийнхөөрөө удирддаг. Хэвийн рестораны хувьд үйлчлүүлэгчдийг ирсэн дарааллаар нь үйлчилдэг бол, Tieu үйлчлүүлэгчдийн дундаж хүлээлгийн хугацааг багасгахыг зорьдог. Тиймээс хэн түрүүлж ирсэн нь хамаагүй, хэн түрүүлж үйлчлүүлэхээ Tieu өөрөө шийддэг.

Янз бүрийн төрлийн пицца хийхэд янз бүрийн хугацаа шаардагддаг. Мөн нэг пиццар хийж эхэлсний дараа, тэр пицца бүрэн бэлэн болтол өөр пицца хийх боломжгүй. Жишээлбэл, гурван үйлчлүүлэгч $t = 0$, $t = 1$, $t = 2$ үед ирдэг гэж үзье. Тэдний пицца хийх хугацаа нь тус бүртээ 3, 9, 6 минут байна. Хэрвээ Tieu ирсэн дарааллаар үйлчилбэл, үйлчлүүлэгчдийн хүлээлгийн хугацаа нь 3, 11, 16 минут болно. Энэ тохиолдолд дундаж хүлээлгийн хугацаа нь $(3 + 11 + 16) / 3 = 10$ минут болно. Гэхдээ энэ нь хамгийн оновчтой шийдэл биш. Эхний үйлчлүүлэгчийг $t = 3$ үед дуусгасны дараа Tieu гурав дахь үйлчлүүлэгчид үйлчилбэл, тэдний хүлээлгийн хугацаа нь 3, 7, 17 минут болно. Ингэснээр дундаж хүлээлгийн хугацаа нь $(3 + 7 + 17) / 3 = 9$ минут болно.

Tieu-д үйлчлүүлэгчдийн дундаж хүлээлгийн хугацааг хамгийн бага байлгахад нь туслаарай. Хялбар болгохын тулд дундаж хугацааны зөвхөн бүхэл хэсгийг ол.

Оролтын формат: Эхний мөрөнд N бүхэл тоо өгөгдөнө — үйлчлүүлэгчдийн тоо. Дараагийн N мөр бүрт хоёр бүхэл тоо өгөгдөнө:

T_i — i дахь үйлчлүүлэгчийн захиалга өгсөн цаг

L_i — тухайн пиццаны хийгдэх хугацаа

Жич: i дахь мөр дээр байгаа үйлчлүүлэгч нь T_i хугацаанд ирсэн үйлчлүүлэгчтэй заавал таарах албагүй.

Гаралтын формат: Хүлээлгийн дундаж хугацааны бүхэл хэсгийг хэвлэ.

Жишээ оролт #00: 3 0 3 1 9 2 6 Жишээ гаралт #00: 9 Жишээ оролт #01: 3 0 3 1 9 2 5
Жишээ гаралт #01: 8 Тайлбар #01: 0 үед захиалсан үйлчлүүлэгчийг А, 1 үед захиалсан үйлчлүүлэгчийг В, 2 үед захиалсан үйлчлүүлэгчийг С гэж нэрлэе. Хэрвээ $A \rightarrow C \rightarrow B$ дарааллаар үйлчилбэл хүлээлгийн хугацаа нь:

А: 3 минут

С: 6 минут (тэр 2-р минут ирсэн ч 3-аас эхэлж 5 минут хүлээнэ $\rightarrow 3+5 = 8 \rightarrow 8 - 2 = 6$)

В: 16 минут Ингээд дундаж нь $(3 + 6 + 16) / 3 = 8.33$ бөгөөд бүхэл хэсэг нь 8 болно. — Код—“PYTHON3” `#!/bin/python3`

```
import math
import os
import random
import re
import sys
```

Complete the ‘minimumAverage’ function below.

The function is expected to return an INTEGER.

The function accepts 2D_INTEGER_ARRAY customers as parameter.

```
import heapq

def minimumAverage(customers):
    customers.sort()
    n = len(customers)
    i = 0
    current_time = 0
    waiting_time = 0
    h = []

    while i < n or h:
        # Add all customers with arrival time less or equal to current time so
        # that no customer that arrives later is processed, as that is not valid.
        while i < n and customers[i][0] <= current_time:
            heapq.heappush(h, (customers[i][1], customers[i][0]))
```



```

        i += 1

    # From the clients whose arrival time is less than the current time,
    extract the one whose pizza has the least preparation time.
    if h:
        prep_time, arrival_time = heapq.heappop(h)
        current_time += prep_time
        waiting_time += current_time - arrival_time

    # In case there are no customers to process, move to next customer. :)
    else:
        current_time = customers[i][0]

return waiting_time // n

if __name__ == '__main__': fptr = open(os.environ['OUTPUT_PATH'], 'w')

n = int(input().strip())

customers = []

for _ in range(n):
    customers.append(list(map(int, input().rstrip().split())))

result = minimumAverage(customers)

fptr.write(str(result) + '\n')

fptr.close()

```

—Тайлбар—

HackerRank Prepare > Data Structures > Heap > Minimum Average Waiting Time

Problem

Tieu owns a pizza restaurant and he manages it in his own way. While in a normal restaurant, a customer is served by following the first-come, first-served rule, Tieu simply minimizes the average waiting time of his customers. So he gets to decide who is served first, regardless of how sooner or later a person comes.

Different kinds of pizzas take different amounts of time to cook. Also, once he starts cooking a pizza, he cannot cook another pizza until the first pizza is completely cooked. Let's say we have three customers who come at time $t=0, t=1$, & $t=2$ respectively, and the time needed to cook their pizzas is 3, 9, & 6 respectively. If Tieu applies first-come, first-served rule, then the waiting time of three customers is 3, 11, & 16 respectively. The average waiting time in this case is $(3 + 11 + 16) / 3 = 10$. This is not an optimized solution. After serving the first customer at time $t=3$, Tieu can choose to serve the third customer. In that case, the waiting time will be 3, 7, & 17 respectively. Hence the average waiting time is $(3 + 7 + 17) / 3 = 9$.

Help Tieu achieve the minimum average waiting time. For the sake of simplicity, just find the integer part of the minimum average waiting time.

Input Format

- The first line contains an integer N , which is the number of customers.
- In the next N lines, the i^{th} line contains two space separated numbers T_i and L_i . T_i is the time when i^{th} customer order a pizza, and L_i is the time required to cook that pizza.
- The i^{th} customer is not the customer arriving at the i^{th} arrival time.

Output Format

- Display the integer part of the minimum average waiting time.

Constraints

- $1 \leq N \leq 10^5$
- $0 \leq T_i \leq 10^9$
- $1 \leq L_i \leq 10^9$

Test Cases

- Test case 0
- Test case 1
- Test case 2
- Test case 3
- Test case 4
- Test case 5
- Test case 6
- Hidden Test Case

Success

You have earned 80.00 points!
You are now 109 points away from the gold level for your problem solving badge.

77% 74/850

Congratulations

You solved this challenge. Would you like to challenge your friends?

[Next Challenge](#)

Hidden Test Case

Unlock this test case for 5 hacks.

[Unlock](#)

File: ./data_structure/hard/3/README.md

truck-tour

—Асуулт—“english” : Suppose there is a circle. There are petrol pumps on that circle. Petrol pumps are numbered to (both inclusive). You have two pieces of information corresponding to each of the petrol pump: (1) the amount of petrol that particular petrol pump will give, and (2) the distance from that petrol pump to the next petrol pump.

Initially, you have a tank of infinite capacity carrying no petrol. You can start the tour at any of the petrol pumps. Calculate the first point from where the truck will be able to complete the circle. Consider that the truck will stop at each of the petrol pumps. The truck will move one kilometer for each litre of the petrol.

Input Format

The first line will contain the value of n . The next lines will contain a pair of integers each, i.e. the amount of petrol that petrol pump will give and the distance between that petrol pump and the next petrol pump.

Constraints:

Output Format

An integer which will be the smallest index of the petrol pump from which we can start the tour.

Sample Input

3 1 5 10 3 3 4

Sample Output

1

Explanation

—Асуулт—“Монголоор” : Гэрлэн тойрог байна гэж төсөөлье. Тэр тойрог дээр шатахуун түгээх станцууд байрлана. Шатахуун түгээх станцуудыг 0-аас $n-1$ хүртэл дугаарласан байдаг (хоёуланг нь оролцуулна). Шатахуун түгээх станц бүрийн хувьд танд хоёр төрлийн мэдээлэл байна: (1) тухайн шатахуун түгээх станц хэр хэмжээний шатахуун өгч чадах вэ, (2) тэр станцаас дараагийн станц хүртэлх зай. Эхэндээ таньд хязгааргүй багтаамжтай, гэхдээ шатахуунгүй сав байгаа. Та аяллаа аль ч шатахуун түгээх станцаас эхлүүлж болно. Ачааны машин тойргийг бүрэн тойрон явах боломжтой хамгийн эхний цэгийг ол. Ачааны машин шатахуун түгээх станц бүр дээр зогсож явна гэж үзнэ. Ачааны машин шатахууны нэг литрт нэг километр явна. Эхний мөрөнд n -ийн утга өгөгдөнө. Дараагийн n мөр тус бүрт хоёр бүхэл тоо өгөгдөх ба, эхнийх нь тухайн станц хэр их шатахуун өгөхийг, хоёр дахь нь дараагийн станц хүртэлх зайг илэрхийлнэ. Аяллаа эхэлж болох хамгийн бага дугаартай шатахуун

түгээх станцын индексийг хэвлэнэ. Бид аяллаа хоёр дахь шатахуун түгээх станцаас эхлүүлж чадна.

—Код—
#!/bin/python3

```
import math import os import random import re import sys
```

Complete the 'truckTour' function below.

The function is expected to return an INTEGER.

The function accepts 2D_INTEGER_ARRAY petrolpumps as parameter.

```
def truckTour(petrol_pumps): start_index = 0 total_liters = 0 total_distance = 0
for i in range(len(petrol_pumps)):
    liters, distance = petrol_pumps[i]
    total_liters += liters
    total_distance += distance
    # Arriving to next petrol pump

    if total_liters - total_distance < 0:
        # Initialize start index before restarting in start index
        start_index = i + 1
        total_liters = 0
        total_distance = 0

return start_index

if name == 'main': fptr = open(os.environ['OUTPUT_PATH'], 'w')
n = int(input().strip())

petrolpumps = []

for _ in range(n):
```

```

petrolpumps.append(list(map(int, input().rstrip().split())))

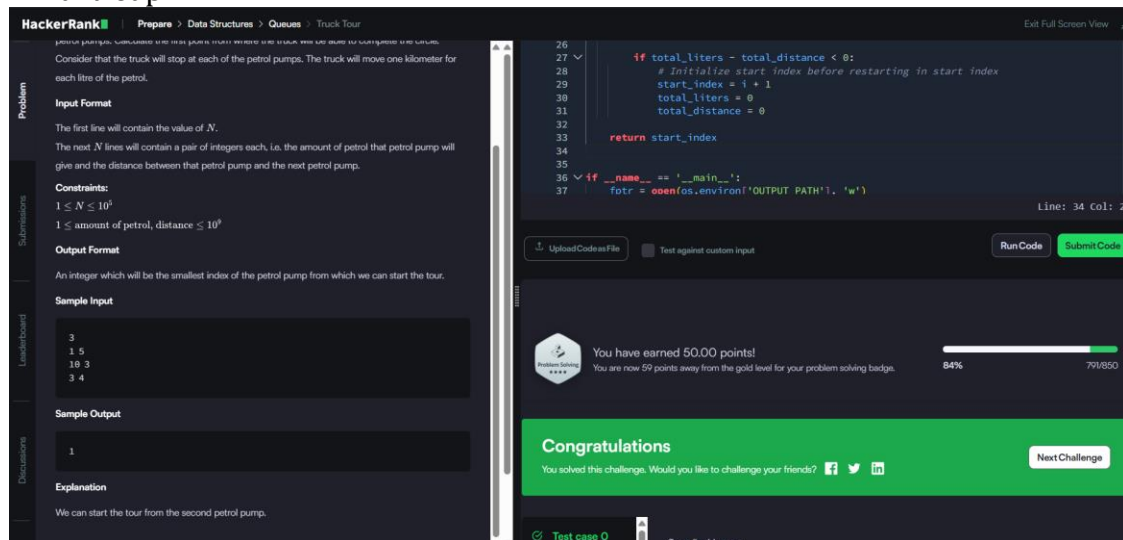
result = truckTour(petrolpumps)

fptr.write(str(result) + '\n')

fptr.close()

```

—Тайлбар



ap—

File: [./data_structure/hard/4/README.md](#)

queries-with-fixed-length

—Асуулт—“english” : Consider an n -integer sequence, a . We perform a query on a by using an integer, k , to calculate the result of the following expression:

In other words, if we let a_i , then you need to calculate $\max_{i=1}^n \min_{j=i}^{i+k-1} a_j$.

Given a and q queries, return a list of answers to each query.

Example

The first query uses all of the subarrays of length $k=3$. The maxima of the subarrays are $[1, 5, 10]$. The minimum of these is 1 .

The second query uses all of the subarrays of length $k=2$. The maxima of the subarrays are $[1, 5, 10, 10]$. The minimum of these is 1 .

Return ans .

Function Description

Complete the solve function below.

solve has the following parameter(s):

int arr[n]: an array of integers int queries[q]: the lengths of subarrays to query Returns

int[q]: the answers to each query Input Format

The first line consists of two space-separated integers, n and q . The second line consists of n space-separated integers, the elements of arr . Each of the subsequent lines contains a single integer denoting the value of $ans[i]$ for that query.

Constraints

Sample Input 0

5 5 33 11 44 11 55 1 2 3 4 5 Sample Output 0

11 33 44 44 55 Explanation 0

For $q=1$, the answer is 11.

For $q=2$, the answer is 33. For $q=3$, the answer is 44. For $q=4$, the answer is 44. For $q=5$, the answer is 55. Sample Input 1

5 5 1 2 3 4 5 1 2 3 4 5 Sample Output 1

1 2 3 4 5 Explanation 1

For each query, the “prefix” has the least maximum value among the consecutive subsequences of the same size.

—Асуулт—“Монголоор” : n бүхэл тооноос бүрдсэн дараалал өгөгдсөн гэж үзье. Бид уг дараалалд нэг бүхэл тоо болох d ашиглан дараах илэрхийллийг бодох асуулт тавина:

$\min(\text{all subarray maxima of length } d)$ Өөрөөр хэлбэл, урт нь d тэнцүү бүх дэд массивуудын хамгийн их утгуудыг олж, тэдгээрийн дундаас хамгийн бага утгыг авна.

Танд arr массив болон $queries$ гэсэн d утгуудын жагсаалт өгөгдөнө. Танд асуулт бүрийн хариуг жагсаалтаар буцаах хэрэгтэй.

Жишээ:

Эхний асуулт нь урт нь 1 тэнцүү бүх дэд массивуудыг авна: [33], [11], [44], [11], [55]. Эдгээрийн хамгийн их утгууд: 33, 11, 44, 11, 55. Эдгээрийн хамгийн бага нь: 11.

Хоёр дахь асуулт нь урт нь 2 тэнцүү дэд массивуудыг авна: [33 11], [11 44], [44 11], [11 55]. Хамгийн их утгууд: 33, 44, 44, 55. Эдгээрийн хамгийн бага нь: 33.

Ингээд үр дүн нь: [11, 33, 44, 44, 55].

Функцийн тайлбар:

solve функц дараах параметруудтэй:

int arr[n]: бүхэл тоон дараалал

int queries[q]: дэд массивын уртуудыг илэрхийлэх бүхэл тоонуудын жагсаалт

Буцаах утга:

int[q]: асуулт бүрийн хариу

Оролтын формат:

Эхний мөрөнд хоёр бүхэл тоо: n ба q

Хоёр дахь мөрөнд n ширхэг бүхэл тоо (arr)

Дараагийн q мөрөнд тус бүр нэг бүхэл тоо байгаа ба энэ нь d утга —Код—” ” #include <bits/stdc++.h>

using namespace std;

string ltrim(const string &); string rtrim(const string &); vector split(const string &);

```
/ Complete the 'solve' function below. The function is expected to return an
INTEGER_ARRAY. * The function accepts following parameters: * 1. INTEGER_ARRAY arr *
2. INTEGER_ARRAY queries */ void add(deque& q, int v){ while(!q.empty() && q.back() <
v) q.pop_back(); q.push_back(v);
}
```

```
int getMax(deque& q){ return q.front(); }
```

```
void pop(deque& q, int rv){ if(!q.empty() && rv == q.front()) q.pop_front(); }
```

```
int computeMinOfSubarrays(vector maxs){ int ans = INT_MAX; for(auto m : maxs) ans =
min(ans, m); return ans;
}
```

```
vector solve(vector arr, vector queries) { vector mins; for(auto subArrayLength : queries){
deque q; vector maxs; int size = 0; int toBeRemoved = 0; for(int i=0; i<arr.size(); ++i){
add(q, arr[i]); size++; if(size == subArrayLength){ size -= 1; maxs.push_back(getMax(q));
pop(q, arr[toBeRemoved++]); } } mins.push_back(computeMinOfSubarrays(maxs)); }
return mins; } int main() { ofstream fout(getenv("OUTPUT_PATH"));
```

```
string first_multiple_input_temp;
getline(cin, first_multiple_input_temp);
```

```
vector<string> first_multiple_input =
split(rtrim(first_multiple_input_temp));
```

```
int n = stoi(first_multiple_input[0]);
```

```
int q = stoi(first_multiple_input[1]);
```

```

string arr_temp_temp;
getline(cin, arr_temp_temp);

vector<string> arr_temp = split(rtrim(arr_temp_temp));

vector<int> arr(n);

for (int i = 0; i < n; i++) {
    int arr_item = stoi(arr_temp[i]);

    arr[i] = arr_item;
}

vector<int> queries(q);

for (int i = 0; i < q; i++) {
    string queries_item_temp;
    getline(cin, queries_item_temp);

    int queries_item = stoi(ltrim(rtrim(queries_item_temp)));

    queries[i] = queries_item;
}

vector<int> result = solve(arr, queries);

for (size_t i = 0; i < result.size(); i++) {
    fout << result[i];

    if (i != result.size() - 1) {
        fout << "\n";
    }
}

fout << "\n";

fout.close();

return 0;
}

string ltrim(const string &str) { string s(str);
    s.erase(
        s.begin(),
        find_if(s.begin(), s.end(), not1(ptr_fun<int, int>(isspace)))
    );
};

```

```

return s;
}

string rtrim(const string &str) { string s(str);

s.erase(
    find_if(s.rbegin(), s.rend(), not1(ptr_fun<int, int>(isspace))).base(),
    s.end()
);

return s;
}

vector split(const string &str) { vector tokens;

string::size_type start = 0;
string::size_type end = 0;

while ((end = str.find(" ", start)) != string::npos) {
    tokens.push_back(str.substr(start, end - start));

    start = end + 1;
}

tokens.push_back(str.substr(start));

return tokens;
}

```

—Тайлбар—

HackerRank | Prepare > Data Structures > Queues > Queries with Fixed Length | Exit Full Screen View

For $d = 4$, the answer is
 $\min(\max(a_0, a_1, a_2, a_3), \max(a_1, a_2, a_3, a_4)) = 44$

For $d = 5$, the answer is
 $\min(\max(a_0, a_1, a_2, a_3, a_4)) = 55$

Sample Input 1

```

5 5
1 2 3 4 5
1
2
3
4
5

```

Sample Output 1

```

1
2
3
4
5

```

Explanation 1

For each query, the "prefix" has the least maximum value among the consecutive subsequences of the same size.

You have earned 50.00 points!
 You are now 9 points away from the gold level for your problem solving badge. 99% 841/850

Congratulations
 You solved this challenge. Would you like to challenge your friends? [f](#) [t](#) [in](#) [Next Challenge](#)

Test case 0 Compiler Message
 Test case 1 Success
 Test case 2
 Test case 3
 Test case 4
 Test case 5
 Test case 6

Hidden Test Case
 Unlock this test case for 5 hacks.
 Unlock

File: ./data_structure/hard/5/README.md

merging

—Асуулт—“english” : People connect with each other in a social network. A connection between Person and Person is represented as . When two persons belonging to different communities connect, the net effect is the merge the communities which and belong to.

At the beginning, there are people representing communities. Suppose person and connected and later and connected, then „ and will belong to the same community.

There are two types of queries:

communities containing persons and are merged if they belong to different communities.

print the size of the community to which person belongs.

Input Format

The first line of input contains 2 space-separated integers and , the number of people and the number of queries. The next lines will contain the queries.

Constraints

Output Format

The output of the queries.

Sample Input

STDIN Function --- 3 6 n = 3, q = 6 Q 1 print the size of the community containing person 1 M 1 2 merge the communities containing persons 1 and 2 ... Q 2 M 2 3 Q 3 Q 2

Sample Output

1 2 3 3 Explanation

Initial size of each of the community is .

—Асуулт—“Монголоор” : Хүмүүс нийгмийн сүлжээнд хоорондоо холбогддог. Хүн а болон хүн b хоёрын хоорондын холбоог (a, b) гэж тэмдэглэнэ. Хэрвээ хоёр хүн өөр өөр нийгэмлэгт харьяалагддаг бөгөөд тэд холбогдвол, тухайн хоёрын харьяалагдаж байсан нийгэмлэгүүд нэгдэх болно.

Эхэндээ n хүн байна гэж үзвэл, бүгд тус тусдаа n ширхэг нийгэмлэгт харьяалагдана. Хэрвээ хүн 1 ба 2 холбогдож, дараа нь хүн 2 ба 3 холбогдвол, 1, 2, 3 бүгд нэг нийгэмлэгт харьяалагдах болно.

Хоёр төрлийн асуулт байна:

$M a b$ — хэрвээ хүн a ба хүн b өөр өөр нийгэмлэгт харьяалагдаж байвал тэдний нийгэмлэгүүдийг нэгтгэнэ.

$Q a$ — хүн a харьяалагдаж буй нийгэмлэгийн хэмжээг хэвлэнэ.

Оролтын формат:

Эхний мөрөнд 2 бүхэл тоо өгөгдөнө: n (хүмүүсийн тоо) ба q (асуултын тоо). Дараагийн q мөр тус бүрт нэг асуулт байна.

Гаралтын формат:

“Асуулт бүрийн хариуг” хэвлэнэ.

Жишээ оролт:

3 6

Q 1

M 1 2

Q 2

M 2 3

Q 3

Q 2 Жишээ гаралт:

1

2

3

```
3 —Код—” ” def union_join(member1, member2): if people_community_dict[member1] >
people_community_dict[member2]: people_community_dict[member2] +=
people_community_dict[member1] people_community_dict[member1] = member2 else:
people_community_dict[member1] += people_community_dict[member2]
people_community_dict[member2] = member1
```

```
def union_find(member): memberList = [] # only when member is parent node id it will
break it while people_community_dict[member] > 0: # looping to find the big parent (As
parent ) memberList.append(member) member = people_community_dict[member] #
redirect all the members when loop through it to the parent for m in memberList:
people_community_dict[m] = member return member
```

```
if name == 'main': num_of_people, num_of_query = map(int, input().split()) # use -1 as init,
only community root having -num and the abs(num) is the size of community
people_community_dict = {i: -1 for i in range(1, num_of_people + 1)}
```

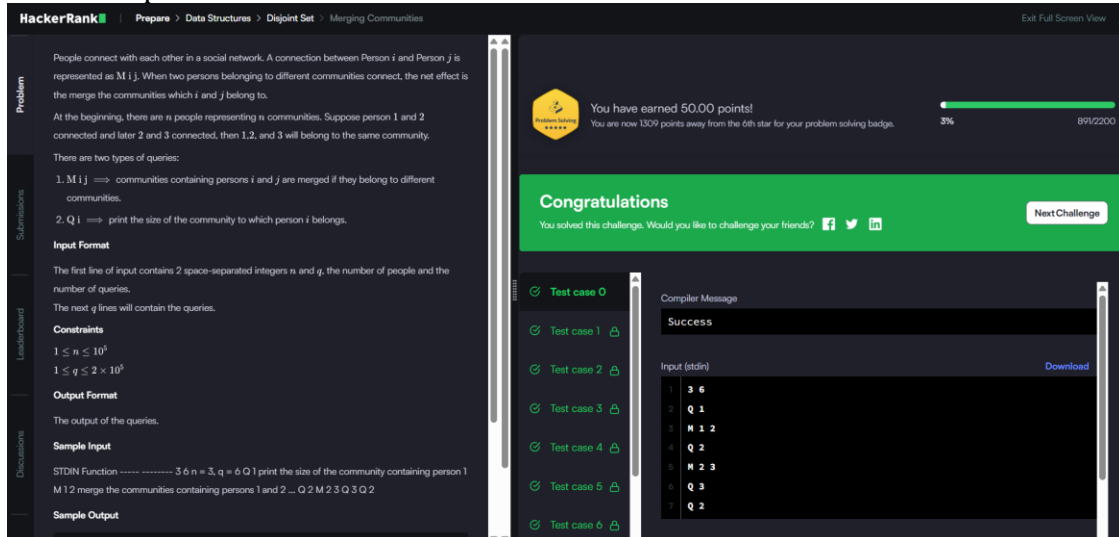
```
for _ in range(num_of_query):
    cur_query = input().split()
    if cur_query[0] == "Q":
        member_id = int(cur_query[1])
        community_root = union_find(member_id)
        print(abs(people_community_dict[community_root]))
    else:
```

```

member_id1, member_id2 = int(cur_query[1]), int(cur_query[2])
root_m1, root_m2 = union_find(member_id1), union_find(member_id2)
if root_m1 != root_m2:
    union_join(root_m1, root_m2)

```

—Тайлбар—



File: ./data_structure/medium/1/README.md

waiter

—Асуулт—“english” :

You are given a list of numbers. For each element at position i , we define $P(i)$ and $N(i)$ as:

$P(i)$ = closest index j such that $j < i$ and $a[j] > a[i]$. If no such j exists then $P(i) = 0$.

$N(i)$ = closest index k such that $k > i$ and $a[k] > a[i]$. If no such k exists then $N(i) = 0$.

We define $S(i) = P(i) * N(i)$. You need to find out the maximum $S(i)$ among all i .

Input Format: The first line contains an integer n , the number of integers. The next line contains the n integers describing the list $a[1..n]$.

Constraints

Output Format: Output the maximum $S(i)$ among all indices from 1 to n .

Sample Input: 5 5 4 3 4 5

Sample Output: 8

Explanation: We can compute the following: The largest of these is 8, so it is the answer.

—Асуулт—“Монголоор” :

Танд тоонуудын жагсаалт өгөгдсөн. Жагсаалтын i байрлал дахь элементийн хувьд $P(i)$ ба $N(i)$ -г дараах байдлаар тодорхойлно:

$P(i) = \sum_{j < i} [a[j] > a[i]]$ байх хамгийн ойр индекс j . Хэрвээ ийм j байхгүй бол $P(i) = 0$.

$N(i) = \sum_{k > i} [a[k] > a[i]]$ байх хамгийн ойр индекс k . Хэрвээ ийм k байхгүй бол $N(i) = 0$.

$S(i) = P(i) * N(i)$ гэж тодорхойлъё. Бүх i -гийн хувьд хамгийн их $S(i)$ -г ол.

Оролтын формат: Эхний мөрөнд n бүхэл тоо байна — тоонуудын тоо. Дараагийн мөрөнд жагсаалтыг илэрхийлэх n ширхэг бүхэл тоо өгөгдөнө $a[1..n]$.

Гаралтын формат: 1-ээс n хүртэлх бүх индексийн хувьд хамгийн их $S(i)$ -г хэвлэ.

Жишээ оролт: 5 5 4 3 4 5

Жишээ гаралт: 8 # —Код—”” #include <bits/stdc++.h>

```
using namespace std;
```

```
string ltrim(const string &); string rtrim(const string &); vector split(const string &);
```

```
long long solve(vector arr) { vector prevBigger(arr.size() + 1); vector nextBigger(arr.size() + 1); for (int i=2; i<= arr.size(); i++) { if (arr[i - 1] < arr[i - 2]) { prevBigger[i] = i - 1; } else { long it = prevBigger[i - 1];
```

```
    while (arr[it - 1] < arr[i - 1] && it != 0) {
        it = prevBigger[it];
    }
```

```
    prevBigger[i] = prevBigger[it];
}
}
```

```
for (int i = arr.size() - 1; i >= 1; i--) {
    if (arr[i] > arr[i - 1]) {
        nextBigger[i] = i + 1;
    } else {
```

```

        long it = nextBigger[i+1];
        while (it != 0 && arr[i - 1] > arr[it - 1]) {
            it = nextBigger[it];
        }
        nextBigger[i] = nextBigger[it];
    }
}
long long res = 0;
for (int i = 1; i <= arr.size(); i++) {
    long long t = prevBigger[i] * nextBigger[i];
    if (t > res) {
        res = t;
    }
}
return res;
}

int main() { // std::ifstream in("input.txt"); // std::streambuf *cinbuf = std::cin.rdbuf();
//save old buf // std::cin.rdbuf(in.rdbuf()); //redirect std::cin to in.txt!

// std::ofstream fout("output.txt");
ofstream fout(getenv("OUTPUT_PATH"));

string arr_count_temp;
getline(cin, arr_count_temp);

int arr_count = stoi(ltrim(rtrim(arr_count_temp)));

string arr_temp_temp;
getline(cin, arr_temp_temp);

vector<string> arr_temp = split(rtrim(arr_temp_temp));

vector<long> arr(arr_count);

for (int i = 0; i < arr_count; i++) {
    long arr_item = stol(arr_temp[i]);

    arr[i] = arr_item;
}

long long result = solve(arr);

fout << result << "\n";

fout.close();

return 0;

```

```

}

string ltrim(const string &str) { string s(str);
    s.erase(
        s.begin(),
        find_if(s.begin(), s.end(), not1(ptr_fun<int, int>(isspace)))
    );
    return s;
}

string rtrim(const string &str) { string s(str);
    s.erase(
        find_if(s.rbegin(), s.rend(), not1(ptr_fun<int, int>(isspace))).base(),
        s.end()
    );
    return s;
}

vector split(const string &str) { vector tokens;

    string::size_type start = 0;
    string::size_type end = 0;

    while ((end = str.find(" ", start)) != string::npos) {
        tokens.push_back(str.substr(start, end - start));

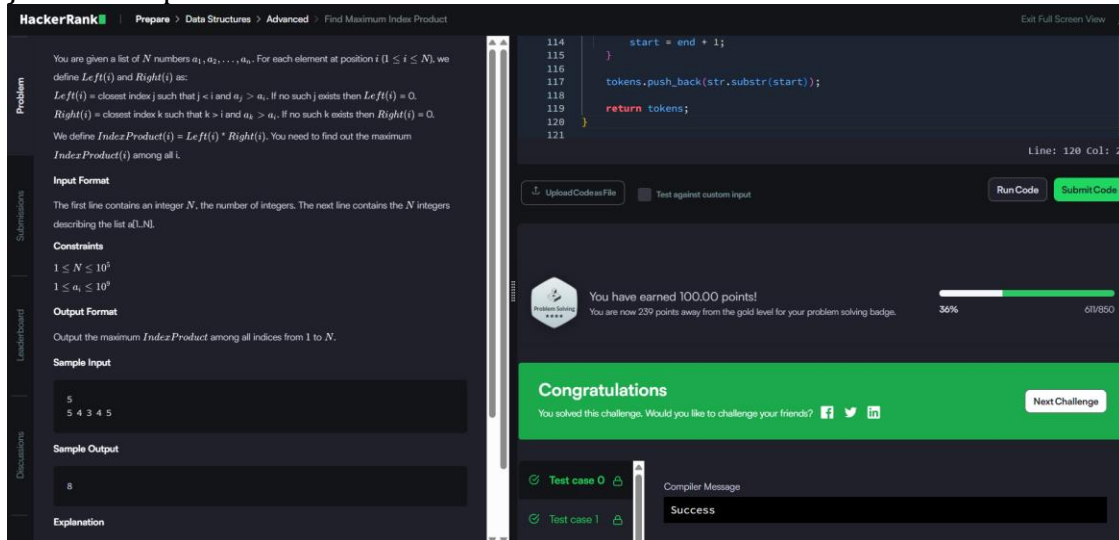
        start = end + 1;
    }

    tokens.push_back(str.substr(start));

    return tokens;
}

```

—Тайлбар—



File: [./data_structure/medium/binary-tree/README.md](#)

Binary tree

—Асуулт—“english” : For the purposes of this challenge, we define a binary tree to be a binary search tree with the following ordering requirements:

The value of every node in a node’s left subtree is less than the data value of that node. The value of every node in a node’s right subtree is greater than the data value of that node. Given the root node of a binary tree, can you determine if it’s also a binary search tree?

Complete the function in your editor below, which has parameter: a pointer to the root of a binary tree. It must return a boolean denoting whether or not the binary tree is a binary search tree. You may have to write one or more helper functions to complete this challenge.

Input Format

You are not responsible for reading any input from stdin. Hidden code stubs will assemble a binary tree and pass its root node to your function as an argument.

Constraints

Output Format

You are not responsible for printing any output to stdout. Your function must return true if the tree is a binary search tree; otherwise, it must return false. Hidden code stubs will print this result as a Yes or No answer on a new line.

Sample Input

tree

Sample Output

No

—Асуулт—“Монголоор” : Сорилын зорилгоор бид хоёртын модыг дараах эрэмбэлэлтийн шаардлагатай хоёртын хайлтын мод (binary search tree) гэж тодорхойлно:

Аль ч зангилааны зүүн дэд модонд байгаа бүх зангилаануудын утга тухайн зангилааны утгаас бага байх ёстой.

Харин баруун дэд модонд байгаа бүх зангилаануудын утга тухайн зангилааны утгаас их байх ёстой.

Чи хоёртын модын үндсэн зангилааг (root) өгөгдсөн гэж үзвэл, тухайн мод нь хоёртын хайлтын мод мөн эсэхийг тодорхойлж чадах уу?

Доорх функцийн биеийг гүйцээ. Энэ функц 1 параметр авна: хоёртын модны үндсэн зангилаа руу заах заагч. Энэ нь тухайн мод хоёртын хайлтын мод мөн эсэхийг илэрхийлэх boolean (үнэн/худал) утга буцаах ёстой.

Энэ сорилын турш функцэд helper функц бичих шаардлагатай байж болох юм.

Оролтын формат: Чи stdin-оос ямар ч оролт унших шаардлагагүй. Нууц код нь хоёртын модыг угсарч, үндсэн зангилааг функц руу дамжуулна.

Хязгаарлалт: (Тодорхой хязгаарлалт өгөгдөөгүй байна, гэхдээ ямар нэг онцгой нөхцөл байж болно.)

Гаралтын формат: Чи ямар ч гаралт хэвлэх шаардлагагүй. Таны функц true (үнэн) утга буцаавал, нууц код нь “Yes”, харин false (худал) бол “No” гэж шугам дээр хэвлэнэ.

Жишээ оролт: tree Жишээ гаралт: No —Код—“Python3” “""" Node is defined as class node: def **init**(self, data): self.data = data self.left = None self.right = None """ def check_binary_search_tree_(root): def is_valid_bst(node, min_val, max_val): if node is None: return True if not (min_val < node.data < max_val): return False return (is_valid_bst(node.left, min_val, node.data) and is_valid_bst(node.right, node.data, max_val))

```
return is_valid_bst(root, float('-inf'), float('inf'))
```

—Тайлбар— Програм эхэндээ check_binary_search_tree_ нэртэй үндсэн функцтэй. Энэ функц нь модны үндсэн root нүдийг авч, туслах функц болох is_valid_bst-ийг дуудаж шалгалт хийдэг.

is_valid_bst туслах функцийн гол үүрэг: Хэрэв тухайн нүд хоосон байвал (мөчир байхгүй бол), энэ нь BST-д саад болохгүй тул True гэж үзнэ.

Харин тухайн нүдний утга нь тодорхой заасан доод болон дээд хязгаарын хооронд байгаа эсэхийг шалгана. Хэрэв зөрчвөл буруу гэж үзээд False буцаана.

Хэрэв зөв бол тухайн нүдний зүүн хүүхдийн модыг дахин доод хязгаар болон энэ нүдний утга хоёрын хооронд байгааг шалгана.

Мөн баруун хүүхдийн модыг энэ нүдний утга болон дээд хязгаар хоёрын хооронд шалгана.

Ийнхүү бүх модыг дотор нь гүнзгий орж шалгаснаар BST-ийн нөхцөл зөрчигдөж байгаа эсэхийг тодорхойлдог.

File: [./data_structure/medium/bodlogo6/README.md](#)

Encryption

Англи текстийг дараах нууцлалын аргаар шифрлэх хэрэгтэй. Эхлээд текстээс бүх зай (space)-г устгана. Дараа нь уг текстийн урт L гэж авъя. Энэ текстийн тэмдэгтүүдийг нэгэн хүснэгтэд бичих бөгөөд тухайн хүснэгтийн мөр ба баганын тоо дараах нөхцлийг хангасан байх ёстой:

$\lceil \sqrt{L} \rceil \leq \text{row} \leq \lceil \sqrt{L} \rceil$, where $\lfloor x \rfloor$ is floor function and $\lceil x \rceil$ is ceil function

Таслалгүй бичвэрийн урт нь 54 тэмдэгт. $\sqrt{54}$ нь 7 ба 8-ын хооронд, тиймээс бичвэрийг 7 мөр ба 8 баганын сүлжээнд байрлуулна. if man was

meant to

say on the

ground go

down would have

vegetables

and roots

- $\text{rows} \times \text{columns} \geq L$ нөхцөлийг хангасан байх ёстой (энд L нь тэмдэгтийн нийт тоо)
- Хэрвээ олон сүлжээ энэ нөхцөлийг хангаж байвал хамгийн бага талбайтай ($\text{rows} \times \text{columns}$ бага) сүлжээг сонгоно.

Кодчилсон мессежийг баганын дагуу тэмдэгтүүдийг уншиж, баганануудын хооронд зай тавьж гаргана. Дээрх сүлжээний кодчилсон мессеж дараах байдалтай байна:

imtg dvs fearwer mayoo go anouuio ntnnlvt wtddes aohghn sseoau

Функц үүсгэх даалгавар: Функцийн тодорхойлолт: Доорх encryption нэртэй функцийг кодыг гүйцээ.

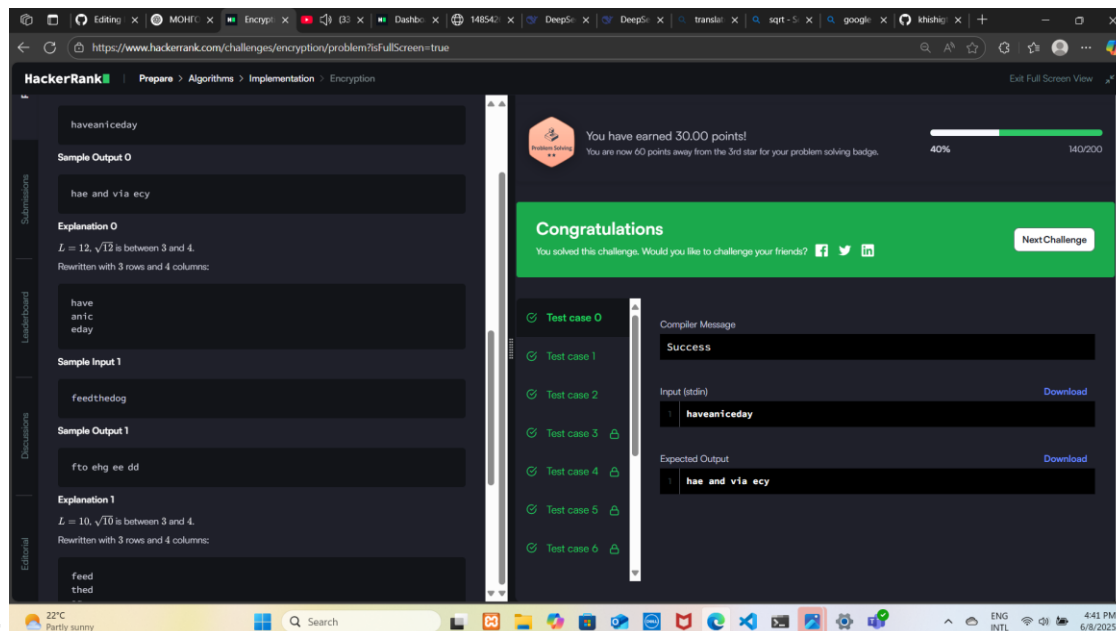
encryption функц дараах параметртэй: string s: кодлох тэмдэгт мөр

Буцаах утга: string: кодлогдсон мессеж

Оролтын формат: Нэг мөрөнд тэмдэгт мөр өгөгдөнө: s

Хязгаарлалт: $1 \leq \text{length of } s \leq 81$

Оролт: haveaniceday Гаралт: hae and via ecy



Үр дүн:

Бодлогын холбоос: <https://www.hackerrank.com/challenges/encryption/problem>

File: `./data_structure/medium/bracket/README.md`

Bracket

—Асуулт—“english” : A bracket is considered to be any one of the following characters: (,), {, }, [, or].

Two brackets are considered to be a matched pair if the an opening bracket (i.e., (, [, or {) occurs to the left of a closing bracket (i.e.,),], or }) of the exact same type. There are three types of matched pairs of brackets: [], {}, and ().

A matching pair of brackets is not balanced if the set of brackets it encloses are not matched. For example, `{[([])}` is not balanced because the contents in between { and } are not balanced. The pair of square brackets encloses a single, unbalanced opening bracket, (, and the pair of parentheses encloses a single, unbalanced closing square bracket,].

By this logic, we say a sequence of brackets is balanced if the following conditions are met:

It contains no unmatched brackets. The subset of brackets enclosed within the confines of a matched pair of brackets is also a matched pair of brackets. Given strings of brackets, determine whether each sequence of brackets is balanced. If a string is balanced, return YES. Otherwise, return NO.

Function Description

Complete the function `isBalanced` in the editor below.

`isBalanced` has the following parameter(s):

string `s`: a string of brackets Returns

string: either YES or NO Input Format

The first line contains a single integer `n`, the number of strings. Each of the next `n` lines contains a single string `s`, a sequence of brackets.

Constraints

`n`, where `n` is the length of the sequence. All characters in the sequences $\in \{ \{, \}, (,), [,] \}$. Output Format

For each string, return YES or NO.

Sample Input

STDIN	Function
3	<code>n = 3</code>
<code>{[()]}</code>	first <code>s = '{[()]}'</code>
<code>{[(())]}</code>	second <code>s = '{[(())]}'</code>
<code>{{[[(())]]}}</code>	third <code>s = '{{[[(())]]}'</code>

Sample Output

YES NO YES Explanation

The string `{[()]}` meets both criteria for being a balanced string. The string `{[(())]}` is not balanced because the brackets enclosed by the matched pair `{` and `}` are not balanced: `[(())]`. The string `{{[[(())]]}}` meets both criteria for being a balanced string.

—Асуулт—“Монголоор” : Тасалгааны хаалтын тэмдэгт гэдэг нь дараах аль нэг тэмдэгтүүд байж болно: `(,), {, }, [,]`, эсвэл `]`.

Хоёр тасалгааны хаалтын тэмдэгт нь таарч байгаа хос гэж үздэг бол нээлттэй тасалгааны хаалтын тэмдэгт `(, [, {)` баруун талд яг адилхан хаалтын тэмдэгттэй `(,], }` байгаа тохиолдол юм. Ийм таарч байгаа гурван төрлийн хос байна: `[], {}, ()`.

Гэхдээ таарч байгаа хос тасалгааны хаалт бүрэн тэнцвэртэй гэж тооцогдохгүй, учир нь түүний доторх тасалгааны хаалт таарч байгаа эсэхийг ч мөн шалгана. Жишээ нь, `{[(())]}` гэдэг нь тэнцвэргүй, учир нь дотор нь байгаа тасалгааны хаалтууд таарч, зөв

дарааллаар байрлаагүй байна. Талын тасалгааны хаалт дотор нэг нээлттэй тасалгааны хаалт () байна, мөн тасалгааны хаалтны дотор хаалттай тасалгааны хаалт () байна.

Иймээс тасалгааны хаалтын дараалал тэнцвэртэй байх нөхцөл нь:

Ямар ч таараагүй тасалгааны хаалт байхгүй байх.

Тэнцвэртэй байгаа хосын доторх тасалгааны хаалт бас тэнцвэртэй байх.

Тэгэхээр, n тооны тасалгааны хаалтын дараалал өгөгдсөн байхад тус бүрийн тэнцвэртэй эсэхийг шалгаад, тэнцвэртэй бол “YES”, тэнцвэргүй бол “NO” гэж хэвлэнэ.

Функцийн тайлбар

isBalanced функцэд нэг параметр орно:

s — тасалгааны хаалтын тэмдэгтүүдийн дараалал.

Функц нь “YES” эсвэл “NO” гэсэн тэмдэгт буцаана.

Оролтын формат

Эхний мөрөнд n — дарааллуудын тоо байна.

Дараагийн n мөрөнд тасалгааны хаалтын дараалал өгөгдөнө.

Жишээ оролт

```
3 {[()]} {[()]} {[[(())]]}
```

Жишээ гаралт

```
YES NO YES
```

Энэ нь ямар учиртай вэ гэвэл:

{[()]} — бүх тасалгааны хаалтууд зөв хослоод, доторх тасалгаанууд нь ч мөн зөв тэнцвэртэй байна.

{[(())]} — доторх тасалгааны хаалтууд таарахгүй байгаа тул тэнцвэргүй.

{[[(())]]} — бүгд зөв дарааллаар таарч тэнцвэртэй. —Код—” ” #include <bits/stdc++.h>

```
using namespace std;
```

```
string ltrim(const string &); string rtrim(const string &);
```

```
/ Complete the 'isBalanced' function below. The function is expected to return a STRING. *  
The function accepts STRING s as parameter. */
```

```
string isBalanced(string s) { stack st; for (char c : s) {
```

```

        if (c == '(' || c == '[' || c == '{') {
            st.push(c);
        } else {
            if (st.empty()) return "NO";

            char top = st.top();
            st.pop();
            if ((c == ')' && top != '(') ||
                (c == ']' && top != '[') ||
                (c == '}' && top != '{')) {
                return "NO";
            }
        }
    }

    return st.empty() ? "YES" : "NO";
}

int main() { ofstream fout(getenv("OUTPUT_PATH"));

    string t_temp;
    getline(cin, t_temp);

    int t = stoi(ltrim(rtrim(t_temp)));

    for (int t_itr = 0; t_itr < t; t_itr++) {
        string s;
        getline(cin, s);

        string result = isBalanced(s);

        fout << result << "\n";
    }

    fout.close();

    return 0;
}

string ltrim(const string &str) { string s(str);

    s.erase(
        s.begin(),
        find_if(s.begin(), s.end(), not1(ptr_fun<int, int>(isspace)))
    );

    return s;
}

```

—Тайлбар— Стек (stack) ашигласан нь учиртай: Тасалгааны хаалтууд нь хамгийн сүүлд нээгдсэн нь хамгийн түрүүнд хаагдах ёстой тул LIFO (Last-In-First-Out) буюу сүүлд орсон нь эхэнд гарна гэсэн шинж чанартай байдаг. Стек энэ зарчмыг яг тохирохуйц өгөгдлийн бүтэц юм.

Хэрвээ с нь нээлттэй хаалт байвал st.push(c) гэж стек дээр хадгална.

Стек хоосон бол шууд буцаана (тэнцвэргүй).

Тохирох хаалт эсэхийг шалгана.

Эцэст нь стек хоосон байвал бүх тасалгааны хаалт таарсан байна, “YES” гэж буцаана. Үгүй бол “NO”.

simple-text-editor

Implement a simple text editor. The editor initially contains an empty string, . Perform operations of the following types: append - Append string to the end of . delete - Delete the last characters of . print - Print the character of . undo - Undo the last (not previously undone) operation of type or , reverting to the state it was in prior to that operation. Example operation index S ops[index] explanation
 --- 0 abcde 1 fg
 append fg 1 abcdefg 3 6 print the 6th letter - f 2 abcdefg 2 5 delete the last 5 letters 3 ab 4
 undo the last operation, index 2 4 abcdefg 3 7 print the 7th character - g 5 abcdefg 4 undo
 the last operation, index 0 6 abcde 3 4 print the 4th character - d The results should be

printed as: f g d Input Format The first line contains an integer, , denoting the number of operations. Each line of the subsequent lines (where) defines an operation to be performed. Each operation starts with a single integer, (where), denoting a type of operation as defined in the Problem Statement above. If the operation requires an argument, is followed by its space-separated argument. For example, if and , line will be 1 abcd.

Constraints The sum of the lengths of all in the input . The sum of over all delete operations . All input characters are lowercase English letters. It is guaranteed that the sequence of operations given as input is possible to perform. Output Format

Each operation of type must print the character on a new line.

Sample Input

STDIN	Function
8	Q = 8
1 abc	ops[0] = '1 abc'
3 3	ops[1] = '3 3'
2 3	...
1 xy	
3 2	
4	
4	
3 1	

Sample Output

c
y
a

Explanation

Initially, is empty. The following sequence of operations are described below:

. We append to , so .

Print the character on a new line. Currently, the character is c.

Delete the last characters in (), so .

Append to , so .

Print the character on a new line. Currently, the character is y.

Undo the last update to , making empty again (i.e.,).

Undo the next to last update to (the deletion of the last characters), making .

Print the character on a new line. Currently, the character is a.

—asuult—“mongoloor” : Энгийн текст засварлагч хэрэгжүүл Засварлагч нь анх хоосон тэмдэгт мөр ("") агуулсан байна. Дараах төрлийн Q тооны үйлдлийг гүйцэтгэнэ:

Үйлдлийн төрлүүд: append s — Тэмдэгт мөр s-ийг одоо байгаа мөрийн төгсгөлд нэмнэ. delete k — Одоо байгаа мөрийн төгсгөлийн k ширхэг тэмдэгтийг устгана. print k — Одоо байгаа мөрийн k-дахь (1-ээс эхэлдэг) тэмдэгтийг хэвлэнэ. undo — Өмнө нь хийсэн хамгийн сүүлийн (буюу өмнө нь “болих” хийгдээгүй) append эсвэл delete үйлдлийг буцааж, тухайн үеийн байдалд сэргээнэ.

Жишээ index Мөрийн утга Үйлдэл Тайлбар 0 abcde 1 fg fg-г нэмнэ 1 abcdefg 3 6 6 дахь үсэг — f 2 abcdefg 2 5 Сүүлийн 5 үсгийг устгана 3 ab 4 Сүүлчийн үйлдлийг болино 4 abcdefg 3 7 7 дахь үсэг — g 5 abcdefg 4 Үйлдэл 0-ийг болино 6 abcde 3 4 4 дэх үсэг — d

Хэвлэгдэх үр дүн: f

g

d Оролтын формат: Эхний мөрөнд Q — нийт үйлдлийн тоо. Дараагийн Q мөрөнд тус бүрийн үйлдлийг оруулна. Хэрэв 1 буюу append бол “1 s” — тэмдэгт мөр. Хэрэв 2 буюу delete бол “2 k” — устгах тэмдэгтийн тоо. Хэрэв 3 буюу print бол “3 k” — хэвлэх индекс. Хэрэв 4 буюу undo бол “4” л байна. Хязгаарлалт: Бүх оролт зөв байна.

Оролтод орсон бүх тэмдэгтүүд жижиг латин үсэг байна. s-ийн нийт урт нь 10^6 -аас хэтрэхгүй. delete-д заагдсан нийт тэмдэгтийн тоо 10^6 -аас ихгүй. Жишээ оролт: 8 1 abc 3 3 2 3 1 ху 3 2 4 4 3 1 Жишээ гаралт: c

у

а

File: ./data_structure/medium/waiter/README.md

waiter

—Асуулт—“english” : You are a waiter at a party. There is a pile of numbered plates. Create an empty array. At each iteration, , remove each plate from the top of the stack in order. Determine if the number on the plate is evenly divisible by the prime number. If it is, stack it in pile . Otherwise, stack it in stack . Store the values in from top to bottom in . In the next iteration, do the same with the values in stack . Once the required number of iterations is complete, store the remaining values in in , again from top to bottom. Return the array.

Example An abbreviated list of primes is . Stack the plates in reverse order. Begin iterations. On the first iteration, check if items are divisible by . Move elements to . On the second iteration, test if elements are divisible by . Move elements to . And on the third iteration, test if elements are divisible by . Move elements to . All iterations are complete, so move the remaining elements in , from top to bottom, to . Return this list. Function Description Complete the waiter function in the editor below. waiter has the following parameters: int number[n]: the numbers on the plates int q: the number of iterations Returns int[n]: the numbers on the plates after processing Input Format The first line

contains two space separated integers, and . The next line contains space separated integers representing the initial pile of plates, i.e., . Constraints Sample Input 0 5 1 3 4 7 6 5 Sample Output 0 4 6 3 7 5 Explanation Initially: = [3, 4, 7, 6, 5]<-TOP

After 1 iteration (divide by 2, the 1st prime number): = [5, 7, 3]<-TOP = [6, 4]<-TOP Move elements to . All iterations are complete, so move elements to . Sample Input 1 5 2 3 3 4 4 9 Sample Output 1 4 4 9 3 3 Explanation 1 Initially: = [3, 3, 4, 4, 9]<-TOP After iteration (divide by 2): = [9, 3, 3]<-TOP = [4, 4]<-TOP Move to . After iteration (divide by 3): = []<-TOP = [3, 3, 9]<-TOP Move elements to . There are no values remaining in . —Асуулт—
“Монголоор” : Нэг багц дугаарласан таваг байна. Эхлээд хоосон массив (list) үүсгэ. Итерац бүрд дараах алхмуудыг хий: stack-ийн орой талаас тавагнуудыг дарааллаар нь ав. Таваган дээрх тоо нь тухайн үеийн i-дах анхны тоонд жигд хуваагдаж байгаа эсэхийг шалга. Хэрвээ жигд хуваагддаг бол, B_i багц руу хий. Үгүй бол A_i багц руу хий. B_i багцын утгуудыг дээдээс нь доош result массив руу хадгал. Дараагийн итерацид, A_i дээрээ яг энэ үйлдлүүдийг давтана. q итерац дуусмагц, A багцад үлдсэн утгуудыг ч мөн адил дээд талаас нь доош result руу нэмнэ. Анхны анхны тоонууд: [2, 3, 5, 7, 11, ...] Тавагнуудыг эсрэг дарааллаар жагсааж байрлуулна (хамгийн сүүлд бичигдсэн таваг хамгийн дээр байна). Итерац эхэлнэ: Эхний итерац: бүгдийг 2-т хуваагдаж байгаа эсэхийг шалгана Хуваагдаж байвал → B1 Үгүй бол → A1 B1-г дээдээс доош result-д нэмнэ Хоёр дахь итерац: A1-ээс авч, 3-т хуваагдах эсэхийг шалгана Хуваагдаж байвал → B2 Үгүй бол → A2 B2-г result-д нэмнэ Гурав дахь итерац: A2 дээр 5-т хуваагдах эсэхийг шалгана Хуваагдвал → B3 Үгүй бол → A3 B3-г result-д нэмнэ Ингээд бүх итерац дууссан бол, үлдсэн A3-ийн утгуудыг ч бас result-д нэмж дуусгана. def waiter(number, q): Оролт: number[n]: Тавагнууд дээрх тоонууд (хамгийн сүүлд бичигдсэн нь дээр байрлана) q: Итерацийн тоо (мөн q ширхэг анхны тоо ашиглана) Гаралт: Боловсруулсан дарааллаар гарсан бүх тавагны дугаарууд (шинэ мөр бүрт нэг утга) Оролтын формат Эхний мөр: n ба q гэсэн хоёр бүхэл тоо (таслалаар салгасан) Хоёр дахь мөр: n ширхэг бүхэл тоо (анхны тавагнуудын дугаар) Жишээ Оролт 1 5 1 3 4 7 6 5 Үйл явц: Эхлээд: A = [3, 4, 7, 6, 5] <- TOP 1-р итерац (анхны тоо 2): Тавганд байгаа тоо 2-т хуваагдах уу Ямар багцад очих вэ 5 Үгүй A1 6 Тийм B1 7 Үгүй A1 4 Тийм B1 3 Үгүй A1 B1 = [6, 4] <- TOP → [4, 6] → result руу орно A1 = [3, 7, 5] <- TOP → [5, 7, 3] → мөн result руу орно Гаралт: 4 6 3 7 5 Жишээ Оролт 2 5 2 3 3 4 4 9 Эхлээд: A = [3, 3, 4, 4, 9] <- TOP Итерац 1 (анхны тоо: 2): 9 → A1 4 → B1 4 → B1 3 → A1 3 → A1 B1 = [4, 4] → result руу A1 = [3, 3, 9] Итерац 2 (анхны тоо: 3): 9 → B2 3 → B2 3 → B2 B2 = [3, 3, 9] → result руу A2 = [] — юу ч үлдээгүй Гаралт: 4 4 9 3 3 —Код—“c++”
#include <bits/stdc++.h> using namespace std;

```
string ltrim(const string &); string rtrim(const string &); vector split(const string &);
```

```
vector waiter(vector number, int q) {
```

```
vector<int> v{};
stack<int> snumber;
vector<int> output;
```

```
for(auto a : number)
{
    cout<<a<<" ";
```

```

        snumber.push(a);
    }
    auto getprimes = [](int q){
        deque<int> primes{};
        int num = 2;
        while (static_cast<int>(primes.size()) < q) {
            bool is_prime = true;
            for (int i = 2; i <= std::sqrt(num); ++i) {
                if (num % i == 0) {
                    is_prime = false;
                    break;
                }
            }
            if (is_prime) {
                primes.push_back(num);
            }
            num++;
        }
        return primes;
    };
    deque<int> nprimes = getprimes(q);
    while (!nprimes.empty()) {
        int prime = nprimes.front();
        nprimes.pop_front();
        stack<int> a;
        stack<int> b;
        while(!snumber.empty())
        {
            int top = snumber.top();
            snumber.pop();
            if(top%prime==0)
                b.push(top);
            else
                a.push(top);
        }
        while(!b.empty())
        {
            int top = b.top();
            b.pop();
            output.push_back(top);
        }
        snumber=a;
    }

    while(!snumber.empty())
    {
        int top = snumber.top();
        snumber.pop();
        output.push_back(top);
    }

```

```

}
return output;
}

int main() { ofstream fout(getenv("OUTPUT_PATH"));

string first_multiple_input_temp;
getline(cin, first_multiple_input_temp);

vector<string> first_multiple_input =
split(rtrim(first_multiple_input_temp));
int n = stoi(first_multiple_input[0]);
int q = stoi(first_multiple_input[1]);

string number_temp_temp;
getline(cin, number_temp_temp);

vector<string> number_temp = split(rtrim(number_temp_temp));
vector<int> number(n);

for (int i = 0; i < n; i++) {
    number[i] = stoi(number_temp[i]);
}

vector<int> result = waiter(number, q);

for (size_t i = 0; i < result.size(); i++) {
    fout << result[i];
    if (i != result.size() - 1) {
        fout << "\n";
    }
}
fout << "\n";
fout.close();

return 0;
}

string ltrim(const string &str) { string s(str); s.erase(s.begin(), find_if(s.begin(), s.end(),
not1(ptr_fun<int, int>(isspace)))); return s; }

string rtrim(const string &str) { string s(str); s.erase(find_if(s.rbegin(), s.rend(),
not1(ptr_fun<int, int>(isspace))).base(), s.end()); return s; }

vector split(const string &str) { vector tokens; string::size_type start = 0; string::size_type
end = 0;

while ((end = str.find(" ", start)) != string::npos) {
    tokens.push_back(str.substr(start, end - start));

```

```
        start = end + 1;
    }

    tokens.push_back(str.substr(start));
    return tokens;
}
```

—Тайлбар— Эхлээд number-ийн бүх элементийг snumber stack-д оруулна.

Prime тоог олж авна.

Prime бүрээр snumber дахь тоог шалгаж, prime-д хуваагдах тоонуудыг b stack-д, бусдыг a stack-д хадгална.

b stack-аас pop хийж гаралт руу нэмнэ.

a-г дараагийн гүйлтэнд зориулан snumber болгож өөрчилнө.




Давталтыг prime тоо бүрээр гүйцэтгэнэ.

Эцэст нь үлдсэн snumber-ийн бүх тоог гаралт руу нэмнэ.

40-bodlogo

Congratulations

You solved this challenge. Would you like to challenge your friends?



Next Challenge

Test case 16

Hidden test case

Message

Success

Test case 17

Test case 18

Test case 19

Test case 20

Test case 21

Test case 22

Input (stdin)

1

4 9 2

2

3 5 7

3

8 1 5

Expected Output

1

1

Download

Download

<https://www.hackerrank.com/challenges/magic-square-forming/problem?isFullScreen=true>

Бид “шидэт квадрат” гэж дараах нөхцөлийг хангасан 3x3 хэмжээтэй матрицыг хэлнэ:

1-ээс 9 хүртэлх давтагдахгүй эерэг тоонууд агуулах ба аль ч мөр, багана, диагоналын гурван тооны нийлбэр нь адилхан тогтмол утгатай байна. Үүнийг “шидэт тогтмол” гэж нэрлэнэ.

Танд 1-ээс 9 хүртэлх бүхэл тооноос бүрдэх 3x3 хэмжээтэй хүснэгт өгөгдөнө. Та хүснэгтэд байгаа дурын тоог 1-ээс 9 хүртэлх өөр нэг тоонд өөрчлөх боломжтой, өөрчлөх зардал нь $|a - b|$ буюу хоёр тооны ялгаврын абсолют утга байна.

📌 Зорилго: Өгөгдсөн хүснэгтийг хамгийн бага зардлаар “шидэт квадрат” болгох, дараа нь уг зардлыг хэвлэ.

Жич: Үүссэн шидэт квадрат нь мөн адил 1-9 хүртэлх давтагдахгүй бүх тоог агуулсан байх ёстой. Жишээ: $s = [[5, 3, 4], [1, 5, 8], [6, 4, 2]]$

Тус матрицыг дараах байдлаар өөрчилж болно: $5\ 3\ 4 \rightarrow 8\ 3\ 4$

$1\ 5\ 8 \rightarrow 1\ 5\ 9$

$6\ 4\ 2 \rightarrow 6\ 7\ 2$

3 ширхэг тоог сольсноор нийт зардал $= 1 + 1 + 1 = 3$ болно.

✓Функцийн тайлбар: formingMagicSquare функц дараах параметртэй: `int s[3][3]`:
бүхэл тоонуудын 3x3 хүснэгт

Буцаах утга: `int`: Хүснэгтийг “шидэт квадрат” болгоход шаардагдах **хамгийн бага зардал**

☐ Оролтын формат: 3 мөр бүхий оролт өгөгдөнө. Мөр бүр нь 3 ширхэг бүхэл тооноос бүрдэнэ, тус бүр нь 1-ээс 9 хооронд байна.

Congratulations

You solved this challenge. Would you like to challenge your friends?

Next Challenge

Test case 5

Test case 6

Test case 7

Test case 8

Test case 9

Test case 10

Test case 11

Success

Input (stdin)

17

2100 100 50 40 40 20 10

34

45 25 50 120

Expected Output

16

24

32

41

Download

Download

<https://www.hackerrank.com/challenges/climbing-the-leaderboard/problem?isFullScreen=true>

Тоглоомын тайлбар




Нэгэн тоглоомын тоглогч лидерүүдийн жагсаалтад дээгүүр байр эзлэхийг хүсэж, өөрийн онооны дагуу байраа хянахыг хүсдэг. Тоглоом нь “Нягт үнэлгээтэй” (Dense Ranking) систем ашигладаг. Энэ систем дараах байдлаар ажилладаг: Хамгийн өндөр оноотой тоглогч 1-р байр эзэлнэ. Ижил оноотой тоглогчид ижил байр эзэлнэ. Дараагийн байр дарааллын дагуу үргэлжилнэ (үндсэн байр алгасахгүй). Жишээ: Хэрвээ үнэлгээний оноо дараах байдалтай байвал: 100 100 50 40 40 20 10 Эдгээр тоглогчдын байр дараах байдалтай байна: 1 1 2 3 3 4 5 Харин тоглогчийн оноо: 5 25 50 120 тус бүрийн дараа дараах байр эзэлнэ: 6 4 2 1 Иймээс хариу нь: [6, 4, 2, 1]

✓Функцын тайлбар List climbingLeaderboard(List ranked, List player) Параметрууд: ranked[n]: Лидерүүдийн оноо (ихээс баг руу эрэмбэлэгдсэн) player[m]: Тоглогчийн оноонууд (багаас их рүү эрэмбэлэгдсэн) Буцаах утга: int[m]: Тоглогчийн байрлалын жагсаалт. Тоглолт бүрийн дараа хэдэд эрэмбэлэгдэж байгааг харуулна.


❑ Оролтын формат: Эхний мөрөнд n тоо байна — лидерүүдийн тоо. Дараагийн мөрөнд n ширхэг зайгаар тусгаарлагдсан тоо байна — ranked оноонууд. Дараагийн мөрөнд m тоо байна — тоглогч хэдэн удаа тоглосон. Эцэст нь m ширхэг зайгаар тусгаарлагдсан тоо байна — player оноонууд.


❑ Хязгаарлалт: $1 \leq n, m \leq 2 \times 10^5$ Бүх оноо 0 ба 10^9 -ийн хооронд байна ranked жагсаалт ихээс баг руу эрэмбэлэгдсэн player жагсаалт багаас их рүү эрэмбэлэгдсэн


Congratulations


You solved this challenge. Would you like to challenge your friends?   


Next Challenge


Test case 6 


Test case 7 

Test case 8 

Test case 9 

Test case 10 

Test case 11 

Test case 12 

Compiler Message

Success

Input (stdin)

1

25

Expected Output

1

15511210043330985984000000

Download

Download

<https://www.hackerrank.com/challenges/extra-long-factorials/problem?isFullScreen=true>

✨Бодлогын орчуулга:

n бүхэл тооны факториал буюу $n!$ гэдгийг дараах байдлаар тодорхойлно: $n! = n \times (n - 1) \times (n - 2) \times \dots \times 3 \times 2 \times 1$ Өгөгдсөн бүхэл тооны факториалын утгыг бодож, хэвлэ. Жишээ: Хэрэв $n = 5$ байвал: $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$

❑ Функцийн тайлбар: extraLongFactorials гэдэг функцыг дараах байдлаар гүйцэтгэнэ. Энэхүү функц нь зөвхөн утгыг хэвлэнэ, буцаалт хийх шаардлагагүй.

❑ Параметр: n : нэг ширхэг бүхэл тоо

Тэмдэглэл: Жич: Том факториалын утгуудыг long зэрэг энгийн өгөгдлийн төрлүүдэд хадгалах боломжгүй байдаг. Иймээс BigInteger зэрэг том тоо хадгалах боломжтой өгөгдлийн төрлийг ашиглах шаардлагатай.

Congratulations

You solved this challenge. Would you like to challenge your friends? [f](#) [t](#) [in](#)

[Next Challenge](#)

Test case 10

Test case 11

Test case 12

Test case 13

Test case 14

Test case 15

Test case 16

Compiler Message

Success

Input (stdin)

1	4 3
2	1 7 2 4

Expected Output

1	3
---	---

Download

Download

<https://www.hackerrank.com/challenges/non-divisible-subset/problem?isFullScreen=true>

Даалгаврын орчуулга:




Өгөгдсөн ялгаатай бүхэл тоонуудын олонлогоос тийм хамгийн том дэд олонлогийг (subset) олж, түүний хэмжээг хэвлэ. Тухайн дэд олонлог доторх аль ч хоёр тооны нийлбэр нь өгөгдсөн k тоонд бүхлээр хуваагдах ёсгүй.

Жишээ нь: Хэрэв $S = [1, 7, 2, 4]$ ба $k = 3$ бол: Боломжит нэг дэд олонлог нь $[1, 7, 4]$ Өөр нэг нь $[1, 2, 4]$ Эдгээрээс хамгийн урт боломжит дэд олонлог нь 3 элементийн урттай байна Функцийн тайлбар nonDivisibleSubset гэдэг функцийг дуусгана уу. `int nonDivisibleSubset(int k, List s)` Оролт: `int s[n]`: ялгаатай бүхэл тоонуудын жагсаалт `int k`: хуваагдах тоо (divisor) Гаралт: `int`: `s` олонлогоос бүтэх хуваагдахгүй дэд олонлогийн хамгийн их хэмжээ Оролтын формат Эхний мөрөнд `n` ба `k` — элементийн

тоо болон хуваагч Дараагийн мөрөнд n ширхэг ялгаатай бүхэл тоо байна
Хязгаарлалт: Бүх өгөгдсөн тоонууд ялгаатай байна.

Congratulations

You solved this challenge. Would you like to challenge your friends?



Next Challenge

Test case 15

Test case 16

Test case 17

Test case 18

Test case 19

Test case 20

Test case 21

Compiler Message

Success

Input (stdin)

1	4 0
2	4 4

Download

Expected Output

1	9
---	---

Download

<https://www.hackerrank.com/challenges/queens-attack-2/problem?isFullScreen=true>

Чингэлэгт орчуулга:

Танд нэг хатан хаан болон зарим саадтай дөрвөлжин шатрын талбай өгөгдөнө. Хатан хаан хэдэн эсрэг талбайг довтолж чадахыг ол. Хатан хаан $n \times n$ хэмжээтэй шатрын талбайд зогсож байна. Талбайн мөрүүд нь доошоос дээш 1-ээс n хүртэл дугаарлагдсан байна. Баганууд нь зүүнээс баруун 1-ээс n хүртэл дугаарлагдсан. Талбай бүрийг (r, c) хосоор тодорхойлно, энд r нь мөр, c нь баганы дугаар.

Хатан хаан (r_q, c_q) байрлалд зогсож байна. Нэг хөдөлгөөнөөр хатан хаан баруун, зүүн, дээш, доош болон дөрвөн диагональ чиглэлээр аль нэг эсрэг талбай руу довтолж чадна. Доорх зурагт (r_q, c_q) -оос хатан хаан довтолж чадах бүх эсрэг талбайг ногоон тойрог дүрсэлсэн байна. Талбай дээр зарим саад байрлаж байгаа бөгөөд эдгээр саадууд хатан хааны довтолох замыг саатуулна. Жишээ нь, зураг дээрх саад $(4, 5)$ байрлал нь хатан хааны довтолж чадах талбайн зарим хэсгийг

хааж байна. Хатан хааны байрлал болон бүх саадын байрлалыг мэдэж байхад хатан хаан довтолж чадах эсрэг талбайн тоог олж хэвлэ. Дээрх зураг дээр хатан хаан 9 эсрэг талбай довтолж чадна.

Функцийн тайлбар queensAttack функцыг editor-д бүрэн гүйцэд бичнэ үү.
queensAttack функц дараах параметруудийг хүлээн авна: int n: талбайн мөр ба баганы тоо int k: саадуудын тоо int r_q: хатан хааны байрлалын мөр int c_q: хатан хааны байрлалын багана int obstacles[k][2]: саадуудын координатууд (мөр, багана) Функц буцаана: int: хатан хаан довтолж чадах эсрэг талбайн тоо Оролтын формат Эхний мөрөнд зайгаар тусгаарлагдсан 2 бүхэл тоо n, k - талбайн хэмжээ болон саадуудын тоо Дараагийн мөрөнд зайгаар тусгаарлагдсан 2 бүхэл тоо r_q, c_q q ,c q - хатан хааны байрлал Дараагийн k мөр бүрд зайгаар тусгаарлагдсан 2 бүхэл тоо r, c, c - тус бүр саадны байрлал Хязгаарлалт Нэг эсрэг талбайд хэд хэдэн саад байж болно. Хатан хааны байрлал дээр саад байхгүй.

Congratulations

You solved this challenge. Would you like to challenge your friends?

Next Challenge

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Test case 6

Compiler Message

Success

Input (stdin)

1 2

2 2

3 1 1

4 1 1

5 2

6 0 2

7 1 1

Expected Output

Download

Download

<https://www.hackerrank.com/challenges/organizing-containers-of-balls/problem?isFullScreen=true>

Асуулт:

Давид хэд хэдэн савтай бөгөөд эдгээр сав бүрт төрөл бүрийн бөмбөгнүүд байна. Тэрбээр байгаа бөмбөг бүрийг тус тусын төрөлд нь хуваарилахад яг тохирох тооны савтай. Давид бөмбөгнүүдийг дараах солилцоо хийх аргаар эмхэлж ангилахыг хүсэж байна: Шаардлага: Давид солилцоо хийхдээ дараах нөхцөлийг хангах ёстой: Сав бүр зөвхөн нэг төрлийн бөмбөг агуулна. Ижил төрлийн бөмбөг хоёр өөр саванд байх ёсгүй. Жишээ: Давид 2 сав, 2 төрлийн бөмбөгтэй байг. Сав дахь бөмбөгнүүдийн байршлыг доорх хүснэгтээр үзүүлж болно: 1 1

1 1 Энэ нь: 1-р сав: 1 ногоон, 1 улаан бөмбөг 2-р сав: 1 ногоон, 1 улаан бөмбөг Давид хоёр савнаас бөмбөг солих нэг үйлдэл хийж чадна (өөрөөр хэлбэл, хоёр өөр савнаас бөмбөг солино). Гэвч ийм солилцоогоор бүх ногоон бөмбөгийг нэг саванд, бүх улааныг нөгөөд нь хийж болохгүй. Иймд хариу нь Impossible (боломжгүй) байна.

Ажиллагаа: Та хэд хэдэн container матриц хэлбэртэй асуултуудыг гүйцэтгэнэ.




Матриц бүрт: container[i][j] нь i-р саванд байгаа j-р төрлийн бөмбөгний тоо байна.

Матриц бүрийн хувьд дараах хариуг хэвлэнэ: Possible, хэрэв Давид бүх ижил төрлийн бөмбөгийг нэг саванд байрлуулж чадвал. Impossible, хэрэв боломжгүй бол. Функцийн тайлбар: String organizingContainers(List<List> container) Параметрууд:

container[n][m] – n ширхэг савны мэдээлэлтэй 2D массив. Тус бүрийн элементийн утга нь тухайн саванд байгаа төрлийн бөмбөгний тоо. Буцаах утга: String: “Possible” эсвэл “Impossible” Оролтын формат: Эхний мөр: q — асуултын тоо Дараагийн мөрүүд: тус бүр n × n хэмжээтэй матриц (savny тоо ба төрөл адил). Хязгаарлалт: Бөмбөгний тоо хэтэрхий их байхгүй (n ≤ 100) Сав бүр өөр төрөл бөмбөг агуулахгүй нөхцлийг солилцоогоор хангаж чадах эсэхийг шалгана.

Congratulations

You solved this challenge. Would you like to challenge your friends?



Next Challenge

Test case 6

Test case 7

Test case 8

Test case 9

Test case 10

Test case 11

Test case 12

Compiler Message

Success

Input (stdin)

1 haveaniceday

Expected Output

1 hae and via ecy

Download

Download

<https://www.hackerrank.com/challenges/encryption/problem?isFullScreen=true>

Шифрлэх даалгаврын орчуулга:

Англи текстийг дараах шифрлэх схем ашиглан шифрлэх шаардлагатай. Эхлээд текст доторх бүх зай (space)-г устгана. Дараа нь тэмдэгтүүдийг нэгэн төрлийн мөр, баганын хязгаарлалттай сүлжээнд (grid) байрлуулна. Жишээ: Жишээ нь, доорх текст өгөгдсөн бол: if man was meant to stay on the ground god would have given us roots

Spaces-ийг устгасны дараа:

ifmanwasmeanttostayonthegroundgodwouldhavegivenusroots Энэ нь 54 тэмдэгттэй.

Тэгвэл: $L = 54$ floor(sqrt(L)) = 7 ceil(sqrt(L)) = 8 → $7 \leq \text{rows} \leq \text{columns} \leq 8$ гэсэн нөхцлийг хангах хамгийн бага rows * columns талбайтай grid нь 7 мөр, 8 багана

байна. Grid-д оруулбал: ifmanwas meanttos tayonthe groundgo dwouldha vegivenu sroots Шифрлэгдсэн үр дүн: Баганаар дараалан тэмдэгтүүдийг авч, хооронд нь зайтай бичнэ: imtgdvs fearwer mayoogo anouuio ntnnlvt wtdddes aohghn sseoau ✓
 Функцийн тайлбар: encryption гэдэг функц нь дараах параметртэй: Параметр: string s – шифрлэх англи текст Буцаах утга: string – шифрлэгдсэн текст ✓Оролтын формат: Нэг мөрөнд шифрлэх текст өгөгдөнө. Зөвхөн жижиг үсгүүд (a-z) болон зай (space, ascii 32) агуулагдана.

Congratulations

You solved this challenge. Would you like to challenge your friends?

Next Challenge

✓ Test case 0

1 5

2 ab

3 bb

4 hefg

5 dhkc

6 dkhc

✓ Test case 1

✓ Test case 2

✓ Test case 3

✓ Test case 4

Expected Output

1 ba

2 no answer

3 heg f

4 dhkc



5 hcdk

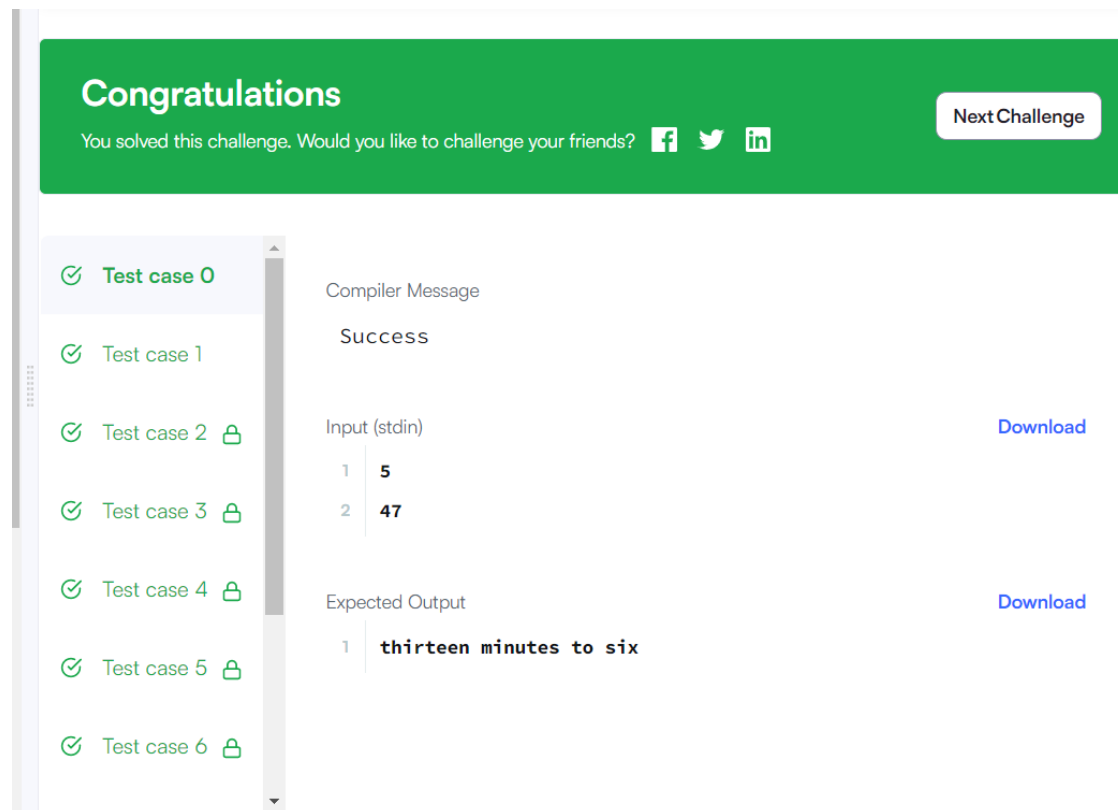
Download

<https://www.hackerrank.com/challenges/bigger-is-greater/problem?isFullScreen=true>

Орчуулга:

Лексикографик дараалал гэдэг нь ихэвчлэн үсгийн дараалал гэж ойлгогддог. Хоёр үгийн аль нь их вэ гэдгийг тодорхойлоход, илүү дараа нь ирдэг үгийг илүү их гэж үзнэ. Танд нэг үг өгөгдөнө. Та энэ үгийн зарим эсвэл бүх үсгийг сольж шинэ үг үүсгэх хэрэгтэй. Шинэ үг дараах хоёр нөхцөлийг хангасан байх ёстой: Шинэ үг нь анхны үгээс их байх ёстой (лексикографикаар дараа нь байх). Тэр их үгнүүдээс хамгийн бага буюу хамгийн ойр дараагийн үг байх ёстой. Жишээ: Жишээ үг: ab Дараагийн хамгийн бага их үг: ba □ Функцийн тайлбар: String biggerIsGreater(String w) Параметр: w: нэг

үг (string) Буцаах утга: Хэрвээ боломжтой бол: анхны үгнээс лексикографикаар их, хамгийн бага боломжит шинэ үгийг буцаана. Боломжгүй бол: “no answer” гэдэг текст буцаана.  Оролтын формат: Эхний мөрөнд: T — тестийн тоо (integer) Дараагийн T мөрөнд: үгнүүд (w) байна  Хязгаарлалт: Үг нь зөвхөн жижиг үсгүүд `ascii[a..z]` агуулсан байна.



The screenshot shows a green banner at the top with the text "Congratulations" and a button "Next Challenge". Below the banner, there is a list of test cases on the left, all marked as "Test case 0" through "Test case 6" with green checkmarks. The main area displays the "Compiler Message" as "Success". Below this, the "Input (stdin)" is shown as two lines: "1 5" and "2 47". To the right of the input is a "Download" button. Below the input, the "Expected Output" is shown as one line: "1 thirteen minutes to six". To the right of the output is another "Download" button.

<https://www.hackerrank.com/challenges/the-time-in-words/problem?isFullScreen=true>

Орчуулга

Цагийг тоон хэлбэрээр өгөгдөхөд үгийг хэлбэрт нь хөрвүүлэх боломжтой, доор үзүүлсэн шиг: Цаг бол o’ clock гэж бичнэ. Минут нь 1-30 хооронд бол past үг ашиглана, харин 31-59 минут бол to үгийг хэрэглэнэ. o’ clock-д байгаа апостроф ба clock хооронд зайтай гэдгийг анхаарна уу. Таны бичих програм нь өгөгдсөн форматаар цагийг үг хэлбэрээр хэвлэх ёстой. Функцийн тодорхойлолт Доорх `timeInWords` функцийг гүйцээ. `timeInWords` функц дараах параметруудийг хүлээн авна: `int h`: Цагны тоо `int m`: Минутын тоо Буцаах утга: `string`: Үгээр илэрхийлсэн цагны хэлбэр Орлын формат Эхний мөрөнд цагны утга `h` орно Хоёрдугаар мөрөнд минутын утга `m` орно.

Congratulations

You solved this challenge. Would you like to challenge your friends? [f](#) [t](#) [in](#)

[Next Challenge](#)

✓ Test case 9

✓ Test case 10

✓ Test case 11

✓ Test case 12

✓ Test case 13

✓ Test case 14

✓ Test case 15

```
13 3 4
14 9505
15 3845
16 3530
17 15 15
18 400453592126560
19 114213133098692
20 474386082879648{-truncated-}
```

[Download to view the full testcase](#)

Expected Output

```
1 YES
2 NO
```

[Download](#)

<https://www.hackerrank.com/challenges/the-grid-search/problem?isFullScreen=true>

Мэдээлэл орчуулга:

Дигит (тоо тэмдэгт) агуулсан мөрүүдийн массив өгөгдөхөд, өгөгдсөн дигитүүдийн хэв маягийг (pattern) хайж олоорой. Grid ба pattern массив бүрийн мөр тус бүр нь grid-ийн нэг мөрийг илэрхийлнэ. Жишээ нь, доорх grid-ийг авч үзье: 1234567890 0987654321

1111111111

1111111111

2222222222

Pattern массив нь: 876543

111111

111111 Энэ pattern нь grid-ийн хоёр дахь мөр, гурав дахь баганаас эхэлж дараагийн хоёр мөрөөр үргэлжилнэ. Иймд pattern нь grid дээр байна гэж үзнэ. Хэрвээ pattern олдвол YES, олдохгүй бол NO гэж буцаана.

Функцийн тайлбар: gridSearch функцыг editor дээр гүйцээнэ үү. Хэрэв pattern grid дээр байгаа бол YES, үгүй бол NO гэж буцаана. gridSearch функцийн параметрууд:

string G[R]: хайх grid string P[r]: хайх pattern Оролтын формат: Эхний мөр нь тестийн тоо T байна. Тэгээд T ширхэг тест тус бүр: Эхний мөр нь хоёр зайгаар тусгаарлагдсан бүхэл тоо R (grid-ийн мөрний тоо), C (мөр бүрийн урт). Дараагийн R мөр бүр нь C урттай digits-ийн мөр (grid). Дараагийн мөр нь хоёр зайгаар тусгаарлагдсан бүхэл тоо r (pattern-ийн мөрний тоо), c (pattern-ийн мөр бүрийн урт). Дараагийн r мөр бүр нь c урттай digits-ийн мөр (pattern). Буцах утга: string: "YES" эсвэл "NO" (pattern grid дээр байвал YES, үгүй бол NO)

Congratulations

You solved this challenge. Would you like to challenge your friends? [f](#) [t](#) [in](#)

Next Challenge

Test case 19

Test case 20

Test case 21

Test case 22

Test case 23

Test case 24

Test case 25

Compiler Message

Success

Input (stdin)

1 1 1

2 1

Expected Output

1 6

Download

Download

<https://www.hackerrank.com/challenges/3d-surface-area/problem?isFullScreen=true>

Орчуулга

Мэдисон бол тоглоомонд дуртай жижигхэн охин. Түүний найз Мэйсон тоглоом үйлдвэрлэдэг үйлдвэрт ажилладаг. Мэйсон-д хэмжээтэй хоёр хэмжээст самбар байдаг, энэ самбар нь мөрүүд ба багануудтай. Самбар нь жижиг эсүүдэд хуваагдсан бөгөөд тус бүрийг координат (i, j)(i,j) гэж тэмдэглэдэг. Эс бүр дээр A[i][j]A[i][j] гэсэн бүхэл тоо бичигдсэн байдаг. Мэйсон тоглоом хийхдээ эс бүр дээр кубыг A[i][j]A[i][j] ширхэгээр дээжээр тавьдаг. Самбарын тайлбараас AA утгуудыг авч, тоглоомын үнийг буюу 3D гадаргын талбайг олж гаргаарай. Оролтын формат: Нэгдүгээр мөрөнд

хоёр зайгаар тусгаарлагдсан бүхэл тоонууд HH ба WW — самбарын өндрийг (мөрүүдийн тоо) болон өргөнийг (багануудын тоо) тус тус өгнө. Дараагийн HH мөр бүрд WW зайгаар тусгаарлагдсан бүхэл тоонууд өгөгдөнө. Эдгээр нь $A[i][j]A[i][j]$ -ийн утгуудыг илэрхийлнэ. Гаралт: Тоглоомын үнийг буюу 3D гадаргын нийт талбайг нэг мөрөнд хэвлэнэ.

Congratulations

You solved this challenge. Would you like to challenge your friends?

Next Challenge

Test case 6

Test case 7

Test case 8

Test case 9

Test case 10

Test case 11

Test case 12

Success

Input (stdin)

13

22 1

33 0

43 2

Expected Output

12 1

21 2 3

3-1

Download

Download

<https://www.hackerrank.com/challenges/absolute-permutation/problem?isFullScreen=true>

ОРчуулна

Бид nn -ийг 1-с эхэлсэн эхний nn бүхэл тооны уртасгасан дараалал гэж тодорхойлно. $pos[i]$ нь i байрлал дээрх утгыг илэрхийлнэ (1-ээс эхлэсэн тоололтой). pos дараалал нь абсолютн уртасгасан дараалал (absolute permutation) гэж тооцогдоно, хэрэв бүх i -ийн хувьд: $|pos[i] - i| = k$ энэ тэгшитгэл биелэгдэнэ гэж үзнэ. Таны өгсөн nn ба k утгын дагуу абсолютн уртасгасан хамгийн эхний (лексикографын хамгийн бага) дарааллыг хэвлэх хэрэгтэй. Хэрвээ ийм дараалал үгүй бол -1 гэж хэвлэнэ. Жишээ: $n=4, k=2$ гэж үзье. 1-ээс 4 хүртэлх тоонуудыг дараалалд оруулна: $pos = [3, 4, 1, 2]$ Энд $|pos[i] - i| = 2$ бүх байрлалд биелнэ: | байрлал i | утга $pos[i]$ | $|pos[i] - i|$ | ————|





———|—————|| 1 | 3 | 2 || 2 | 4 | 2 || 3 | 1 | 2 || 4 | 2 | 2 | Функцийн тайлбар
 absolutePermutation функц нь дараах параметруудтэй: int n: Тоонуудын дээд хязгаар
 (1-с эхэлсэн) int k: Байршил ба утгын абсолютын зөрүү Оролтын формат Эхний
 мөрөнд qq - асуултын тоо байна. Дараа нь qq мөрөнд тус бүр nn ба kk хоёр утга орно.
 Үр дүн nn урттай лексикографын хамгийн бага абсолютын уртасгасан дараалал
 (пермутаци) Хэрвээ байхгүй бол -1 -1 гэж хэвлэнэ.

The screenshot shows a green banner at the top with the text "Congratulations" and "You solved this challenge. Would you like to challenge your friends?" followed by social media icons for Facebook, Twitter, and LinkedIn. A "Next Challenge" button is on the right. Below the banner, on the left, is a list of test cases from 19 to 25, each with a green checkmark and a lock icon. In the center, the "Compiler Message" section shows "Success". Below that, the "Input (stdin)" section displays a 7x7 grid of characters: Row 1: "6 7 3", Row 2: ".....", Row 3: "...0...", Row 4: "...0..", Row 5: ".....", Row 6: "00.....", Row 7: "00.....". To the right of the input is a "Download" button. Below the input, the "Expected Output" section is visible with another "Download" button.




<https://www.hackerrank.com/challenges/bomberman/problem?isFullScreen=true>

Тоглоомын нөхцөл



Bomberman нэг тэгш өнцөгт сүлжээнд (grid) амьдардаг. Энэ сүлжээний тус бүр эс нь дараах 2-ын нэг байдаг: Бөмбөгтэй эс Хоосон эс ☐ Бөмбөгийн дүрэм: Бөмбөгийг дурын эст суулгаж болно. Суулгаснаас хойш яг 3 секундйн дараа дэлбэрдэг. Дэлбэрэх үед: Өөрөө устаж алга болно. Дөрвөн хөрш эстэйгээ (дээр, доор, зүүн, баруун) хамт устгана. Хэрэв хөрш эст бас бөмбөг байвал тэр нь дэлбэрэлгүй устдаг (гинжин дэлбэрэлт үүсэхгүй). ☐ Bomberman өөрөө: Бөмбөгнөөс дархлаатай тул чөлөөтэй хөдөлж чадна. ☐ Bomberman юу хийдэг вэ? Эхэндээ ямар ч байдлаар хэд



хэдэн эсэд бөмбөг суулгана. (Анхны байдал) 1 секунд өнгөрөхөд, юу ч хийхгүй. 2 дахь секунд: Бүх хоосон эст бөмбөг суулгана. Ингэснээр бүх эс бөмбөгтэй болно. (Энэ үед бөмбөг дэлбэрэхгүй) 3 дахь секунд: 3 секундын өмнө суулгасан бөмбөгнүүд дэлбэрнэ. (Анхны бөмбөгнүүд) 3, 4-р алхмыг дараагийн секундүүдэд байнга давтана.  Бүх бөмбөгийг нэгэн зэрэг суулгаж, нэгэн зэрэг дэлбэрүүлдэг!  Таны даалгавар: Танд эхний байдал өгөгдөнө (сүлжээний дүр зураг) — та n секундын дараах сүлжээний байдал-ыг олж мэдэх ёстой.  Оролт (Input Format) Эхний мөр: $r\ c\ n\ r$ — мөрийн тоо c — баганын тоо n — хэдэн секунд симуляци хийх Дараагийн r мөр бүр: сүлжээний байдал “.” — хоосон эс “O” — бөмбөгтэй эс  Гаралт (Output Format) r мөр бүхий тэмдэгт мөрүүд (багц) n секундын дараах сүлжээний байдал



Congratulations



You solved this challenge. Would you like to challenge your friends?   



Next Challenge



 Test case 16 



 Test case 17 

 Test case 18 

 Test case 19 

 Test case 20 

 Test case 21 

 Test case 22 

Compiler message

Success

Input (stdin)

1	5 6
2	GGGGGG
3	GBBBBGB
4	GGGGGG
5	GGBBGB
6	GGGGGG

Expected Output


1	5
---	---

[Download](#)

<https://www.hackerrank.com/challenges/two-pluses/problem?isFullScreen=true>

Эмма квант компьютер бүтээжээ!

Түүний чадварыг шалгахад туслаарай, доорх бодлогыг шийднэ үү:

 Бодлогын тайлбар: Өгөгдсөн $n \times m$ хэмжээтэй тор (grid) байна. Энэ торны бүх нүд G эсвэл B тэмдэгттэй байна: G — сайн (green) нүд, энд plus зурах боломжтой. B — муу

(bad) нүд, энд юу ч зурах боломжгүй. **+**“Plus” гэж юу вэ? Plus (нэмэх тэмдэг) гэдэг нь гол нүд-нээс дээш, доош, зүүн, баруун чиглэлд ижил урттай “гар” сунгасан хэлбэр юм. Жишээлбэл: G GGGGG G Энэ нь голдоо G тэмдэгтэй, тал бүрт 2 урттай гартай size = 2 plus юм. Нийт талбай: $4 * size + 1 = 9$. Зорилго: 2 ширхэг “plus” ол. Эдгээр нь давхцахгүй байх ёстой (аль ч нүд давхардахгүй). Тэдгээрийн талбайн үржвэр хамгийн их байх ёстой. Жишээ: Хоёр plus: нэг нь 9 талбай, нөгөө нь 5 талбай. Үржвэр: $9 * 5 = 45$. Функцийн тодорхойлолт: `int twoPluses(List grid)` Параметрууд: grid: мөрүүдийн жагсаалт. Бүх тэмдэгт G эсвэл B. Оролтын формат: Эхний мөрөнд 2 бүхэл тоо байна — n ба m. Дараагийн n мөр бүрт m ширхэг тэмдэгтэй string байна. Гаралтын формат: Хамгийн том 2 plus-н талбайн үржвэрийг нэг бүхэл тоогоор хэвлэ. Хязгаарлалт: $2 \leq n, m \leq 15$

Congratulations

You solved this challenge. Would you like to challenge your friends?

Next Challenge

Test case 15

Test case 16

Test case 17

Test case 18

Test case 19

Test case 20

Test case 21

Compiler Message

Success

Input (stdin)

1

3

2

3

3

3 1 2

4

4

5

1 3 4 2

6

5

7

1 2 3 5 4

Expected Output

Download

Download




<https://www.hackerrank.com/challenges/larrys-array/problem?isFullScreen=true>

Орчуулга

Ларриг 1-ээс эхэлсэн дарааллын нэгэн орлуулгыг массив хэлбэрээр өгсөн. Тэрээр дараах үйлдлийг хэдэн ч удаа давтаж массивыг эрэмбэлж болох эсэхийг тодорхойлох ёстой: Үргэлжлүүлэн байрласан ямар нэгэн 3 индексийг сонгож, эдгээр гурван

элементүүдийг тойруулан эргүүлэх (rotation) үйлдлийг хийх. Жишээ нь: А эргүүлэлт [1,6,5,2,4,3] [6,5,2] [1,5,2,6,4,3] [5,2,6] [1,2,6,5,4,3] [5,4,3] [1,2,6,3,5,4] [6,3,5] [1,2,3,5,6,4] [5,6,4] [1,2,3,4,5,6] Хэрэв бүх элементүүдийг эрэмбэлж чадвал (sorted array) "YES" гэж хэвлэ, үгүй бол "NO" гэж хэвлэ. Функцийн тайлбар larrysArray функц нь массив А-г параметр болгон авч, "YES" эсвэл "NO" утгыг буцаана. Оролтын формат Эхний мөрөнд туршилтын тоо t байна. Дараагийн t хос мөр бүр: Нэг мөрөнд массивын урт n Дараагийн мөрөнд n ширхэг зайгаар тусгаарлагдсан бүхэл тоон элементүүд өгөгдөнө. Хязгаарлалт Массивын элементүүд 1-ээс n хүртэл тасралтгүй өссөн тоонууд байна. Гаралтын формат Туршилтын бүрд нэг мөрөнд "YES" эсвэл "NO" гэж хэвлэнэ.

Congratulations

You solved this challenge. Would you like to challenge your friends?   

Next Challenge

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Test case 6

Compiler Message

Success

Input (stdin)

1

2

2

4 2

Expected Output

1

yes

2

swap 1 2

Download

Download

<https://www.hackerrank.com/challenges/almost-sorted/problem?isFullScreen=true>

ОРчуулга

Өгөгдсөн бүхэл тооны массивыг зөвхөн дараах нэг үйлдлээр өсөх дарааллаар эрэмбэлж болох эсэхийг тодорхойлно уу: Хоёр элементийг солино. Нэг тасалбар хэсгийг буцааж эргүүлнэ. Үйлдлүүдийн аль нэгийг ашиглан массивыг эрэмбэлж болдог эсэхийг шалгаад, үр дүнг дараах байдлаар гаргана: Хэрвээ массив аль

хэдийнээ эрэмбэлэгдсэн байвал, эхний мөрөнд yes гэж хэвлэ. Дараагийн мөрийг хэвлэх шаардлагагүй. Хэрвээ массивыг зөвхөн нэг үйлдлээр эрэмбэлж болдог бол, эхний мөрөнд yes гэж хэвлээд дараах байдлаар үргэлжлүүлнэ: Хэрвээ зөвхөн хоёр элементийг солих замаар л эрэмбэлж болох бол, хоёр дахь мөрөнд swap l r гэж хэвлэнэ. Энд l ба r нь солино гэж байгаа элементүүдийн индекс бөгөөд массивын индекс 1-ээс эхэлнэ гэж үзнэ. Харин зөвхөн тасалбар хэсгийг буцааж эргүүлэх замаар л эрэмбэлж болох бол, хоёр дахь мөрөнд reverse l r гэж хэвлэнэ. Энд l ба r нь эргүүлэх тасалбар хэсгийн эхний ба сүүлийн элементүүдийн индекс (1-ээс эхлэх) болно. Хэрвээ массивыг солих болон эргүүлэх аль аль үйлдлээр эрэмбэлж болдог бол, заавал солих (swap) үйлдлийг сонгоно. Хэрвээ массивыг эдгээр үйлдлүүдийн аль нэгээр ч эрэмбэлж болдоггүй бол эхний мөрөнд no гэж хэвлэнэ. Жишээ Массивыг эрэмбэлэхийн тулд 3 болон 4-р индекст байгаа элементийг солих буюу эсвэл тэдгээрийг буцааж эргүүлэх боломжтой. Өмнө дурдсанчлан солих үйлдлийг илүүд үздэг тул: Эхний мөрөнд: yes Хоёр дахь мөрөнд: swap 3 4 гэж хэвлэнэ. Функцийн тайлбар almostSorted функц нь дараах параметртэй: arr: бүхэл тооны массив (int arr[n]) Функц нь үр дүнг хэвлэх бөгөөд юу ч буцаахгүй. Оролтын формат Эхний мөрөнд массивын урт n байна. Дараагийн мөрөнд n ширхэг зайгаар тусгаарлагдсан бүхэл тоо байна. Хязгаарлалт Бүх элементүүд өөр хоорондоо ялгаатай байна. Гаралтын формат Хэрвээ массив аль хэдийн эрэмбэлэгдсэн бол, эхний мөрөнд yes гэж хэвлэнэ. Хэрвээ нэг үйлдлээр массивыг эрэмбэлж болдог бол эхний мөрөнд yes гэж, хоёр дахь мөрөнд swap l r эсвэл reverse l r гэж хэвлэнэ. Хэрвээ эрэмбэлж чадахгүй бол no гэж хэвлэнэ.