Graph Transformation

Contests > ICPC Asia West Continent Finals 2021 > Graph Transformation

DESCRIPTION SUBMISSIONS EDITORIAL

For a simple undirected graph G with N nodes numbered 1, 2, ..., N , define the transformation F(G) to be the graph G' , that has 2N nodes numbered 1, 2, ..., 2N , and has edges (u, v + N) and (u + N, v) for each edge (u, v) of G .

Also, define T(k, G) to be the graph obtained on applying the transformation k times in succession to G . So, T(0, G) = G , T(1, G) = F(G) , T(2, G) = F(F(G)) , and so on.

You are given a graph G with N vertices 1, 2, ..., N and M edges. You have to process Q queries of the form:

- 1 a b : Add the edge (a, b) to G
- 2 k : Print the number of connected components in T(k, G) .

Input Format

- The first line contains three integers, N , M and Q .
- Each of the next M lines contains two integers u and v denoting an edge between u and v
- Then the Q queries follow. Each query is of the form 1 a b or 2 k , as described in the statement

Output Format

Process the queries and print the required number of connected components for each query of type 2 .

Constraints

- 1 ≤ N, M, Q ≤ 300,000
- 1 ≤ M ≤ N * (N - 1) / 2
- 1 ≤ u, v, a, b ≤ N
- 0 ≤ k ≤ 30
- The graph remains simple throughout the queries, i.e. never contains any self loops or multiple edges.

Sample

Input

```
5 1 5
1 2
2 0
1 2 3
1 3 4
1 1 3
2 1
```

Output

```
4
3
```

Explanation

- In the first query of type 2 , k = 0 and we simply have to print the number of connected components in G , which is 4
- In the second query of type 2 , k = 1 , so we have to print the number of components in F(G) , which is 3 .

Time Limit
1000 ms

Memory Limit
262144 KiB

Language: C++

Theme: Light

Key map: Default

Font size: Normal

Tab size: 4 Spaces

```
// TLEs on CodeDrills - not sure if error in code or platform
```

```cpp


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

int find(vector<int>& dsu, int i){
    if (dsu[i] != i) dsu[i] = find(dsu, dsu[i]);
    return dsu[i];
}

int join(vector<int>& dsu, vector<int>& sz, vector<int>& bipartite, int i, int j){
    if (find(dsu, i) != find(dsu, j)){
        if (sz[dsu[i]] > sz[dsu[j]]){
            bipartite[dsu[i]] &= bipartite[dsu[j]];
            sz[dsu[i]] += sz[dsu[j]];
            dsu[dsu[j]] = dsu[i];
        } else {
            bipartite[dsu[j]] &= bipartite[dsu[i]];
            sz[dsu[j]] += sz[dsu[i]];
            dsu[dsu[i]] = dsu[j];
```

Ln: 1, Col: 1 Key Map: default

LOAD FILE

□ Custom input

RUN CODE SUBMIT

Discuss Contact Us Campus Chapter

Privacy Policy | Terms of service

Mathematicians And Physicists

Contests > ICPC Asia West Continent Finals 2021 > Mathematicians And Physicists

DESCRIPTION SUBMISSIONS EDITORIAL

As the captain of a spaceship heading to outer space, you need to choose a team of scientists from two groups: N mathematicians and M physicists.

Each mathematician has an IQ value $A[i]$ and an adaptation value $B[i]$. Adaptation value of a mathematician represents the maximum number of physicists that the mathematician is willing to travel with.

Similarly, each physicist has an IQ value $C[i]$ and an adaptation value $D[i]$. Adaptation value of a physicist represents the maximum number of mathematicians that the physicist is willing to travel with.

Your goal is to select a team of scientists such that the sum of their IQ values is as high as possible, while also respecting each scientist's adaptation value. Find out the maximum total IQ of the selected scientists?

Input Format

- The first line contains an integer N - the number of mathematicians.
- The next line contains N integers $A[0], A[1], ..., A[N-1]$ - IQs of the mathematicians.
- The next line contains N integers $B[0], B[1], ..., B[N-1]$ - adaptation of the mathematicians.
- The next line contains an integer M - the number of physicists.
- The next line contains M integers $C[0], C[1], ..., C[M-1]$ - IQs of the physicists.
- The next line contains M integers $D[0], D[1], ..., D[M-1]$ - adaptation of the physicists.

Output Format

Print an integer - the maximum possible sum of IQs.

Constraints

- $1 \le N \le 100$
- $1 \le M \le 10^5$
- $1 \le A[i] \le 10^5$ for all $0 \le i < N$.
- $0 \le B[i] \le 10^5$ for all $0 \le i < N$.
- $1 \le C[i] \le 10^5$ for all $0 \le i < M$.
- $0 \le D[i] \le 10^5$ for all $0 \le i < M$.

Sample 0

Input

```
1
4
0
2
1 2
0 0
```

Output

```
4
```

Explanation

There is only 1 mathematician with an IQ of 4 and an adaptation value of 0 , meaning that they are not willing to travel with any physicists. There are 2 physicists, with IQs of 1 and 2 and adaptation values of 0 and 0 , respectively, meaning that they are also not willing to travel with any mathematicians.

You can either choose the only mathematician (which will give an IQ equal to 4), or the two physicists (which will give an IQ equal to 3), so the answer is 4 .

Sample 1

Input

```
1
4
```

1 2
1 1

Output

6

Explanation

The perfect combination is to choose the only mathematician and the second physicist, and this will give as a sum of IQs equal to `4 + 2 = 6`.

Time Limit
2500 ms

Memory Limit
262144 KiB

Language
C++

Theme
Light

Key map
Default

Font size
Normal

Tab size
4 Spaces

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

int main(){
    int n; cin >> n;
    vector<pair<int, int>> math_tol_int (n);
    for (int i = 0; i < n; i++) cin >> math_tol_int[i].second;
    for (int i = 0; i < n; i++) cin >> math_tol_int[i].first;
    sort(math_tol_int.begin(), math_tol_int.end(), std::greater<pair<int, int>>());

    int m; cin >> m;
    vector<pair<int, int>> phy_int_tol (m);
    for (int i = 0; i < m; i++) cin >> phy_int_tol[i].first;
    for (int i = 0; i < m; i++) cin >> phy_int_tol[i].second;
    sort(phy_int_tol.begin(), phy_int_tol.end(), std::greater<pair<int, int>>()); // sort by desc intelligence

    vector<vector<long long>> phy_tol_intpref(101, vector<long long> (1, 0));
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j <= 100; j++)
            if (phy_int_tol[i].second >= j) // if can tolerate
```

Ln: 1, Col: 1   Key Map: default

LOAD FILE

☐ Custom input

RUN CODE   SUBMIT

Discuss   Contact Us   Campus Chapter

Veristablium

Contests > ICPC Asia West Continent Finals 2021 > Veristablium

DESCRIPTION SUBMISSIONS EDITORIAL

For a sequence `X` of length `n` (where `n` is at least `3`), we define the strength of `X` as the sum of squares of adjacent differences when it is placed on a circle. Specifically, the strength of `X` is given by the following formula:

`strength(X) = sum{ (X[i] - X[(i mod n) + 1])^2 }` over `1 ≤ i ≤ n`

For example, the strength of the sequence `[1, 2, 1, 3]` is `(1-2)^2 + (2-1)^2 + (1-3)^2 + (3-1)^2 = 10`.

We also define the instability of a sequence as the difference between the maximum and minimum possible strengths over all of its permutations. For example, for the sequence `[1, 3, 2, 1]`, the maximum possible strength is `10` (for the permutation `[1, 2, 1, 3]`) and the minimum possible strength is `6` (for the permutation `[1, 3, 2, 1]` itself). Therefore, the instability of this sequence is `10 - 6 = 4`.

Given a sequence `S`, find the sum of the instabilities of all subsequences of `S` of length at least `3`, modulo `998244353`.

Input Format

- The first line contains `T`, the number of testcases.
- Then the testcases follow. For each test case,
  - The first line contains `n`, the length of the sequence `S`.
  - The second line contains `n` space separated integers denoting the sequence.

Output Format

For each testcase, print the sum of the instabilities of all subsequences of `S` of length at least `3`, modulo `998244353`.

Constraints

- `1 ≤ T ≤ 100`
- `3 ≤ n ≤ 200,000`
- `0 ≤ S[i] < 998244353`
- The sum of `n` over all testcases doesn't exceed `200,000`

Sample

Input

```
2
4
1 3 2 1
6
5 9 3 11 0 7
```

Output

```
4
2104
```

Explanation

In the first testcase,

- For any sequence of length `3`, note that any permutation has the same strength, so the instability is `0`.
- For the full sequence, the minimum and maximum strengths are `6` and `10` respectively, as explained in the problem description.

Time Limit
2000 ms

Memory Limit
262144 KiB

Language
C++

Theme
Light

Key map
Default

Font size

Tab size

Normal ▾

4 Spaces ▾

Ln: 1, Col: 1

Key Map: default

LOAD FILE

□ Custom input

RUN CODE

SUBMIT

Discuss Contact Us Campus Chapter

Privacy Policy | Terms of service

Keyboard String

Contests > ICPC Asia West Continent Finals 2021 > Keyboard String

DESCRIPTION SUBMISSIONS EDITORIAL

There are `26` buttons on a keyboard, one for each English letter from `A` to `Z`. When a button is pressed, the corresponding character is printed.

Recently, `M` of the buttons (`B[1], B[2], ..., B[M]`) have begun to malfunction. When button `B[i]` is pressed, it not only prints the corresponding character, but also presses another button `C[i]` automatically.

Your task is to determine whether a given string `S` of length `N` was printed using the malfunctioning keyboard.

Consider an example: Let there be two malfunctioning buttons: `H` and `E`.

- When the `H` button is pressed, it not only prints out the letter `H`, but also automatically presses the button `E`.
- Similarly, when the `E` button is pressed, it not only prints out the letter `E`, but also automatically presses the button `L`.

We can print string `HELLO` using this keyboard. Here is the corrected breakdown of the button presses, with the buttons that are manually pressed listed in bold and the buttons that are automatically pressed listed in italics:

- Press **H** (manually): This prints out `H` and also presses the button `E` (automatically).
- Press *E* (automatically): This prints out `E` and also presses the button `L` (automatically).
- Press *L* (automatically): This prints out `L`.
- Press **L** (manually): This prints out `L`.
- Press **O** (manually): This prints out `O`.

However, we can't print string `HI` using this keyboard. This is because if you press `H`, then the character `E` is automatically pressed next.

Input

- First line contains `T`, the number of test cases.
- Then test cases follow. For each test case,
  - The first line contains an integer `N` representing the length of the string.
  - The second line contains the string `S`.
  - The third line contains an integer `M` - the number of malfunctioning buttons.
  - `M` lines follow, where the `i`-th line contains two space-separated uppercase characters representing `B[i]` and `C[i]`.

Output

- For each test case, print `YES` if the string `S` was printed using the keyboard, otherwise print `NO`.

Constraints

- `1 ≤ T ≤ 100`
- `1 ≤ N ≤ 10^5`
- `S` contains uppercase English characters only.
- `1 ≤ M ≤ 26`
- Each of `B[i]` and `C[i]` represent an uppercase English character.
- `B[i]` is unique across all `M` malfunctioned buttons.
- Sum of `N` across all testcases does not exceed `10^6`.

Sample

Input

```
2
4
ABCD
2
A B
C D
4
ABCD
[illegible]
A B
```

```
B C
C D
D E
```

Output

```
YES
NO
```

Explanation

- Test case 1: `A` can be pressed first, due to which `A` gets printed and `B` gets pressed. Since `B` is not malfunctioning, it only prints `B`. Next, `C` can be pressed which leads to `C` getting printed and `D` getting pressed. Since `D` is not malfunctioning, only `D` gets printed. Hence, `ABCD` gets printed.
- Test case 2: If `A` is pressed, it would print `A` and press `B`. `B` is malfunctioning, hence it prints `B` and presses `C`. Similarly, `C` and `D` are also malfunctioning, which means `C` gets printed, then `D` gets pressed, `D` gets printed and eventually `E` gets pressed. Overall, `ABCDE` would get printed.

Time Limit
1000 ms

Memory Limit
262144 KiB

Language
C++

Theme
Light

Key map
Default

Font size
Normal

Tab size
4 Spaces

```
for _ in range(int(input())):
    n = int(input())
    s = input()
    m = int(input())
    malf = {}
    for i in range(m):
        a, b = input().split()
        malf[a] = b
    cur = 0
    while cur < len(s):
        if s[cur] in malf:
            if cur+1 < len(s) and malf[s[cur]] == s[cur+1]:
                cur += 2
            else:
                print("NO")
                break
        else:
            cur += 1
    else:
        print("YES")

```

Ln: 1, Col: 1 Key Map: default

LOAD FILE

□ Custom input

RUN CODE SUBMIT

New Zoo Construction

Contests > ICPC Asia West Continent Finals 2021 > New Zoo Construction

DESCRIPTION SUBMISSIONS EDITORIAL

You and your friend Igor live in an infinite 2D matrix `A` . `A[i][j]` is the cell of matrix `A` at the intersection of the `i` -th row and the `j` -th column for any integers `i` and `j` such that `i, j` in `(-infinity, infinity)` .

Igor is going to build a zoo and has already designed a building plan for it. The building plan includes building `n` barriers, where the `i` -th barrier `(1 <= i <= n)` can be represented by four integers: `x[i][1]` , `y[i][1]` , `x[i][2]` , and `y[i][2]` . These integers have the following properties:

- For any integer `i (1 <= i <= n)` , `x[i][1] <= x[i][2]` and `y[i][1] <= y[i][2]` .
- For any integer `i (1 <= i <= n)` , either `x[i][1] = x[i][2]` or `y[i][1] = y[i][2]` (but not both at the same time). This means that the the barrier must be either a vertical line or a horizontal line, but not both.
- For any integers `i` and `j` such that `1 <= i < j <= n` and `x[i][1] = x[i][2] = x[j][1] = x[j][2]` , either `y[i][2] < y[j][1] - 1` or `y[j][2] < y[i][1] - 1` . This means that if two barriers are both vertical lines and share the same x-coordinate, then the y-coordinates of the two barriers must be such that there is at least one cell of space between them.
- For any integers `i` and `j` such that `1 <= i < j <= n` and `y[i][1] = y[i][2] = y[j][1] = y[j][2]` , either `x[i][2] < x[j][1] - 1` or `x[j][2] < x[i][1] - 1` . This means that if two barriers are both horizontal lines and share the same y-coordinate, then the x-coordinates of the two barriers must be such that there is at least one cell of space between them.

Originally, there is a one-step passage between each cell `A[i][j]` and its neighbor cells `A[i+1][j]` , `A[i][j+1]` , `A[i-1][j]` , and `A[i][j-1]` . In other words, there is a one-step passage between each cell and the cells immediately adjacent to it (above, below, to the left, and to the right).

We will say that there is a path between `A[i][j]` and `A[p][q]` if there is a sequence of cells `b[1], b[2], ..., b[m]` such that `b[1] = A[i][j]` , `b[m] = A[p][q]` , and there is a one-step passage between cells `b[i]` and `b[i+1]` for any integer `i (1 <= i <= m-1)` .

Igor wants to label each cell in the matrix with a number, called the aviary number, based on the paths that are possible between cells. The aviary number of a cell is determined by the following rules:

- If there is a path between two cells `A[i][j]` and `A[p][q]` , then the aviary numbers of both cells (`A[i][j]` and `A[p][q]`) must be the same.
- If there is no path between two cells `A[i][j]` and `A[p][q]` , then the aviary numbers of the cells (`A[i][j]` and `A[p][q]`) must be different.

The number of aviaries is the total number of unique aviary numbers that are used to label the cells in the matrix. Igor wants to know this number, because it represents the maximum number of animals that can be in the zoo. In other words, the cells in the matrix represent enclosures for the animals, and the barriers represent the walls between the enclosures. The aviary numbers represent the different enclosures, and the number of aviaries is the total number of enclosures in the zoo.

Find out the total number of aviaries in the zoo.

Input Format

- The first line contains the number of test cases `t` .
- Description of the test cases follows. For each test case,
  - The first line contains single integer `n` .
  - Each of the next `n` lines contains `4` integers `x[i][1]` , `y[i][1]` , `x[i][2]` and `y[i][2]` .

Output Format

For each test case print one integer - number of existing aviaries in the zoo.

Constraints

- `1 <= t <= 2*10^5`
- `0 <= n <= 2*10^5`
- `-10^9 <= x[i][1], y[i][1], x[i][2], y[i][2] <= 10^9` for all valid `i` .
- It is guaranteed that the sum of `n` over all test cases does not exceed `2*10^5` .

nple 0

Input

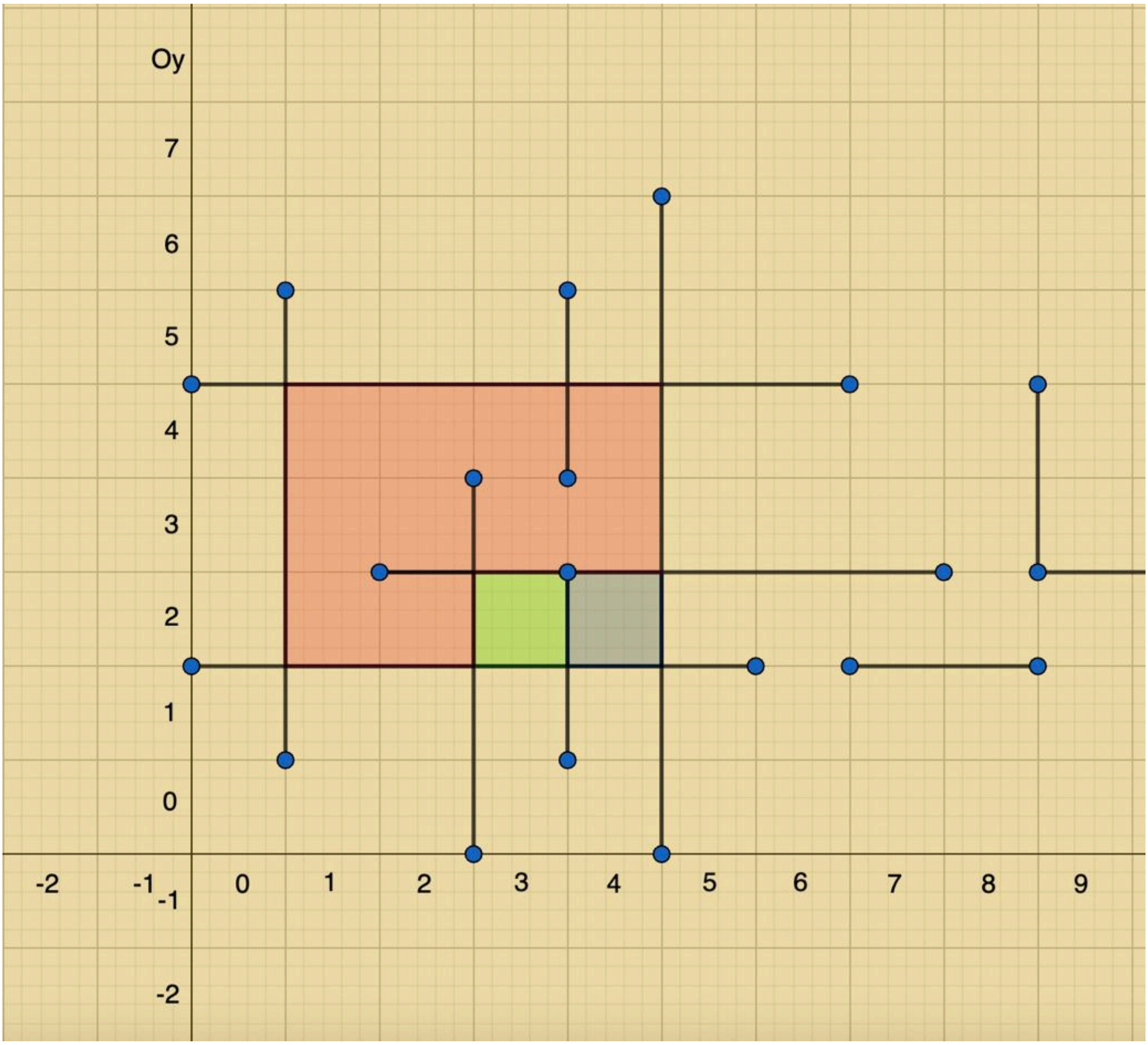
```
6
1
0 0 1 0
4
0 1 0 2
1 2 2 2
2 1 2 2
1 0 2 0
6
0 1 0 3
1 1 1 3
2 1 2 3
1 0 3 0
1 1 3 1
1 2 3 2
0
2
-1000000000 0 1000000000 0
0 -1000000000 0 1000000000
11
0 1 0 5
0 4 6 4
4 0 4 6
0 1 5 1
2 2 7 2
2 0 2 3
3 1 3 2
3 4 3 5
7 1 8 1
8 3 8 4
9 2 10 2
```

Output

```
1
2
5
1
1
4
```

Explanation

Matrix `A` in the last test case after adding barriers (the same color two cells mean that these cells have the same number of aviary to which they belongs):



Time Limit
8000 ms

Memory Limit
524288 KiB

Language: C++

Theme: Light

Key map: Default

Font size: Normal

Tab size: 4 Spaces

Ln: 1, Col: 1 Key Map: default

LOAD FILE

□ Custom input

RUN CODE SUBMIT

Discuss Contact Us Campus Chapter

Privacy Policy | Terms of service

Graph Reconstruction

Contests > ICPC Asia West Continent Finals 2021 > Graph Reconstruction

DESCRIPTION SUBMISSIONS EDITORIAL

There is a simple undirected graph `G` with `N` vertices and `M` edges. For each `1 ≤ L ≤ R ≤ N`, you are told `C[L][R]`, the number of connected components in the subgraph of `G` induced by the vertices in the range `[L, R]`, i.e. the number of connected components in the graph with vertices `L, L + 1, ..., R` and the edges of `G` with both endpoints in the range `[L, R]`.

Find if there exists such a graph. If it does, print the edges of any such graph.

Input Format

- The first line contains `T`, the number of testcases.
- Then the testcases follow. For each test case,
  - The first line contains two integers, `N` and `M`
  - `i-th` of the next `N` lines contains `N - i + 1` integers, `C[i][i], C[i][i + 1], ..., C[i][N - 1], C[i][N]`

Output Format

For each testcase,

- If there exists no valid graph, print `-1` on a new line.
- Else, print `M` lines each containing the endpoints of an edge of the graph. The graph described by these `M` edges must be simple (no self loops or multi-edges).

Constraints

- `1 ≤ T ≤ 300`
- `1 ≤ N ≤ 300`
- `1 ≤ M ≤ N * (N-1) / 2`
- `1 ≤ C[L][R] ≤ N`
- The sum of `N` over all testcases doesn't exceed `300`

Sample

Input

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

Output

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

Explanation

- In the first testcase, you can verify that the graph with edges `(1, 3)` and `(2, 3)` satisfies all the conditions.
- Since `N = 3, M = 2` and the graph is simple, it must be connected, but `C[1][3] = 2`, thus no such graph exists.
- This is the same as the above with the exception of `M = 1`.

Time Limit
1000 ms

Memory Limit
262144 KiB

C++

Theme: Light

Key map: Default

Font size: Normal

Tab size: 4 Spaces

```
#include <iostream>
#include <vector>
#include <set>
#include <algorithm>

using namespace std;

int find(vector<int>& dsu, int i) {
    if (dsu[i] != i) {
        dsu[i] = find(dsu, dsu[i]);
    }
    return dsu[i];
}

bool join(vector<int>& dsu, vector<int>& sz, int i, int j) {
    int a = find(dsu, i);
    int b = find(dsu, j);
    if (a != b) {
        if (sz[a] < sz[b]) swap(a, b);
        sz[a] += sz[b];
        dsu[b] = a;
```

Ln: 1, Col: 1   Key Map: default

LOAD FILE

□ Custom input

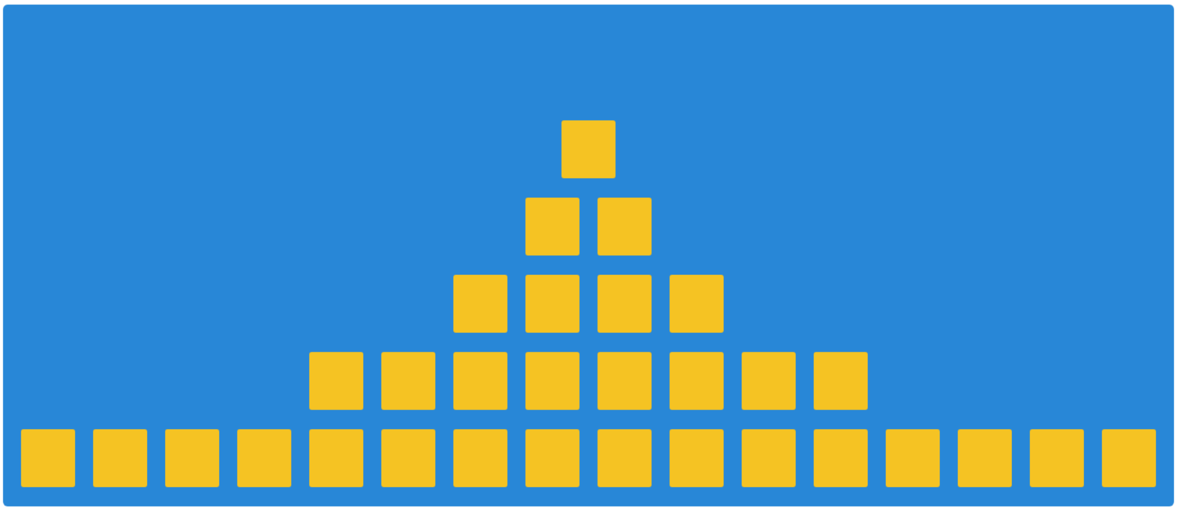
RUN CODE   SUBMIT

Discuss   Contact Us   Campus Chapter

Privacy Policy | Terms of service

Reviving Old Mughal architecture

The Mughal emperors took great interest in constructing beautiful buildings, though unfortunately few have survived to this day. You have been given the task of helping a tourist visit a building that is being reconstructed based on the designs of the Mughal emperors. The building has `N` floors, numbered `1` through `N` from top to bottom, and the `i`-th floor has `2^(i-1)` rooms, for example: the top floor has one room, the next floor has `2` rooms, the next one has `4` rooms, and so on.



The tourist has an enjoyment level that starts at `X`. When they visit the `i`-th floor, their enjoyment level increases by `E[i]`. The value of `E[i]` can be negative as well. The tourist must visit the building starting from the top floor and going down, and they may skip one or more floors (including the top floor as well). However, the tourist's enjoyment level **must not** be **negative at the end** of the visit, even if it was *negative at intermediate points* during the visit.

Your goal is to help the tourist choose which floors to visit in order to maximize the number of rooms visited, while also ensuring that their enjoyment level does not end up negative.

Input Format

- The first line contains an integer `T`, representing the number of test cases.
- Then test cases follow:
  - First line contains two integers `N` and `X` denoting the number of floors in the building and initial amount of excitement the visitor has.
  - Next line contains `N` integers `E[1], E[2],…, E[N]`. Here `E[i]` is the excitement, the visitor gains after visiting the `i`-th floor.

Output Format

For each test case output two lines as follows:

- A line containing a single integer `K` specifying the number of floors they should visit.
- Next line containing `K` space-separated integers, the numbers of floors they should visit. The floor numbers must be printed in increasing order

If there are multiple optimal solutions possible, print any of them.

Constraints

- `1 ≤ T ≤ 10^4`
- `1 ≤ N ≤ 10^5`
- `1 ≤ X ≤ 10^9`
- `-10^9 ≤ E[i] ≤ 10^9` for `1 ≤ i ≤ N`
- Sum of `N` across all testcases is `≤ 10^5`

nple

Input

```
3
5 3
-10 1 2 3 4
5 1
-10 -1 -22 3 -4
7 1
-10 2 -5 -6 2 2 2
```

Output

```
5
1 2 3 4 5
2
4 5
5
2 4 5 6 7
```

Explanation

- In the first query, the visitor has visited all the `5 floors`. They visit `1 + 2 + 4 + 8 + 16 = 31` rooms which is maximum possible, while `E[3] >= 0`.
- But in the second query they only preferred to visit the **fourth** and **fifth** floors from the top. The number of rooms visited is now `8 + 16 = 24` which is maximum possible, while `E[0] >= 0`.

Time Limit
1000 ms

Memory Limit
262144 KiB

Language
C++

Theme
Light

Key map
Default

Font size
Normal

Tab size
4 Spaces

Ln: 1, Col: 1

Key Map: default

LOAD FILE

□ Custom input

RUN CODE SUBMIT

Binary Occurences

Given an array `A` of length `N`, count the number of *good* subarrays in `A`. A subarray is considered *good* if for each element that appears in the subarray, it appears a power of two number of times (i.e. 1, 2, 4, 8 and so on).

Input Format

- The first line contains `T`, the number of testcases.
- Then the testcases follow. For each test case,
  - The first line contains an integer `N` - the size of the array.
  - The next line contains `N` integers - the array elements.

Output Format

For each testcase, print a single line containing the answer.

Constraints

- `1 ≤ T ≤ 10^5`
- `1 ≤ N ≤ 10^5`
- `1 ≤ A[i] ≤ N` for each `0 ≤ i < N`.
- It's guaranteed that the sum of `N` over all testcases doesn't exceed `10 ^ 5`.

Sample 0

Input

```
5
4
1 2 1 1
3
1 1 1
2
1 2
3
2 2 3
6
3 6 6 2 4 2
```

Output

```
9
5
3
6
21
```

Explanation

Let's look at the second testcase where the array is `[1, 1, 1]`, then we can see that the good subarrays are:

1. `sub(1,1)`
2. `sub(1,2)`
3. `sub(2,2)`
4. `sub(2,3)`
5. `sub(3,3)`

Where `sub(l,r)` is the subarray that starts at `l` and ends at `r`. And so, since the number of these subarrays is `5`, the answer is equal to `5`.

Time Limit
[illegible]0 ms

Memory Limit
512000 KiB

Language

Language
C++

Theme
Light

Key map
Default

Font size
Normal

Tab size
4 Spaces

Ln: 1, Col: 1 Key Map: default

LOAD FILE

□ Custom input

RUN CODE SUBMIT

Discuss Contact Us Campus Chapter

Zebra Submatrices

You are given a grid with `N` rows and `N` columns, containing cells that are either white or black. A sub-matrix is called a *zebra* sub-matrix if each row of the sub-matrix has all cells of the same color, and no two consecutive rows have the same color. An example of a valid zebra sub-matrix is:

```
WWW
BBB
WWW
```

A sub-matrix is called a **maximally zebra** sub-matrix if it is a zebra sub-matrix and it is not contained inside another zebra sub-matrix. Your task is to count the number of **maximally zebra** sub-matrices in the grid.

Note that a sub-matrix of a matrix is a matrix formed by selecting a contiguous block of rows and columns from the matrix. The rows and columns of the sub-matrix are consecutive, meaning that they are all adjacent to each other and there are no gaps between them. You can also think a sub-matrix as an axis-parallel rectangle on the given matrix.

Input

- First line contains `T`, the number of test cases.
- Then test cases follow. For each test case,
  - The first line contains `N`, the number of rows and columns in the grid.
  - The `i-th` of the next `N` lines contains the `i-th` row of the grid

Output

- For each test case, output a single line consisting of an integer corresponding to the count of **maximally zebra** sub-matrices.

Constraints

- `1 ≤ T ≤ 5`
- `1 ≤ N ≤ 300`
- Each cell of the grid contains either `W` or `B`

Sample

Input

```
2
3
WBW
WWW
BBB
5
BWBBW
BWBWW
WWWBW
BBBWW
WWWBB
```

Output

```
4
12
```

Explanation

In the first testcase, there are `4` *maximally-zebra* matrices:

- `[1, 1] × [1, 1]`
- `[1, 3] × [2, 2]`
- `[1, 1] × [3, 3]`
- `2, 3] × [1, 3]`

e `[xl, xr] × [yl, yr]` denotes the submatrix containing cells `(x, y)` with `xl ≤ x ≤ xr`, and `yl ≤ y ≤ yr`.

Time Limit
2000 ms

Memory Limit
262144 KiB

Language
C++ ▾

Theme
Light ▾

Key map
Default ▾

Font size
Normal ▾

Tab size
4 Spaces ▾

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

int main(){
    int t;
    cin >> t;
    while (t--){
        int n;
        cin >> n;
        vector<vector<int>> row_errors(n); // error = i means i, doesn't match with i-1
        vector<vector<int>> col_errors(n); // error = i means i, matches with i-1
        vector<vector<int>> grid (n, vector<int> (n));
        for (int i = 0; i < n; i++){
            for (int j = 0; j < n; j++)
            {
                char x;
                cin >> x;
                grid[i][j] = (x == 'W') ? 1 : 0;
                if (j >= 1 && grid[i][j] != grid[i][j-1]) row_errors[i].push_back(j);
                if (i >= 1 && grid[i][j] == grid[i-1][j]) col_errors[j].push_back(i);
            }
```

Ln: 1, Col: 1 Key Map: default

LOAD FILE

□ Custom input

RUN CODE SUBMIT

Discuss Contact Us Campus Chapter

Privacy Policy | Terms of service

White Black Grid

Given a grid `C` of size `N * N`. Each cell is either white or black. Print `max(1, floor((N^2) / 7 - 35))` corner vertex disjoint axis parallel rectangles where all its corners are of the same colour. i.e

- rectangle's sides must be parallel to axis
- no two rectangles can share a common corner. (their sides can overlap)
- All corners of a rectangle must be of the same color.
- A rectangle must have its corner vertices located in different cells. This means that the height and width of each rectangle should be at least 2.
- Note that a 'corner' refers to a cell.
- Additionally, it is guaranteed that a solution exists within the bounds specified in the problem statement."

Input format

- First line contains `T` the number of test cases.
- For each test case:
  - First line contains `N`: the size of the grid.
  - Next `N` lines each containing `N`-size strings.
  - The `j`'th character of `i`'th string denotes the color of cell `C[i][j]`.

Output format

- You need to print `max(1, floor((N^2) / 7 - 35))` lines.
- Each line representing `4` points of the rectangle.
- On each line print `8` integers `x1 y1 x2 y2 x3 y3 x4 y4`. Here `(x1, y1)`, `(x2, y2)`, `(x3, y3)` and `(x4, y4)` are the 4 points of the rectangle in any order.

Constraints

- `1 ≤ T ≤ 1000`
- `7 ≤ N ≤ 1000`

  Sum of `N^2` over all test cases doesn't exceed `10^6`.

Sample 1

Input

```
1
10
WWBWWBBWWB
WBWBBWWBBW
BWWBBWBBBW
WBBBBWWBWB
BBBWBBWBBW
BBBWBBBWBW
BBBWBWBWBW
WWBWBWBWWB
BBBWWWWWBW
BBWBWBWBBW
```

Output

```
2 8 2 9 3 8 3 9
```

Explanation

It forms a rectangle with all its corners as `B`.

Time Limit
1000 ms

Memory Limit
262144 KiB

Language
C++ ▾

Theme
▾

Key map
Default ▾

Font size

Tab size

Normal ▾ 4 Spaces ▾

```
def solve(grid):
    lim = max(1, (len(grid)*len(grid))//7 - 35)
    if n <= 8:
        for i in range(n):
            for j in range(i):
                for l in range(n-1):
                    for k in range(1, n-l):
                        if grid[i][l] == grid[j][l] == grid[i][l+k] == grid[j][l+k]:
                            print(i+1, l+1, j+1, l+1, i+1, l+k+1, j+1, l+k+1)
    else:
        ans = []
        mapper = {0:(0, 1), 1: (1, 2), 2: (0, 2), 3:(0, 1), 4:(0, 1), 5:(0, 2), 6:(1, 2), 7:(0, 1), 8:(0, 1)}
        for i in range(0, n-3, 3):
            prev = {}
            for j in range(n):
                cur = grid[i][j] + grid[i+1][j]*2 + grid[i+2][j]*4
                if cur in prev:
                    ans.append((prev[cur], j))
                    del prev[cur]
                else:
                    prev[cur] = (j, [i + x for x in mapper[cur]])
```

Ln: 1, Col: 1 Key Map: default

LOAD FILE

□ Custom input

RUN CODE SUBMIT

Discuss Contact Us Campus Chapter

Privacy Policy | Terms of service