Problem A
AB to C

Time limit: 1 second
Memory limit: 1.5 Gigabytes

Problem Description

You are given a string consisting only of characters A, B, or C. You can do the following operation as many times as you like:

- Choose $1 \leq i < j \leq |S|$ such that $S_i \neq S_j$.
  - Delete S_i and S_j.
  - Insert the complementary character of (S_i, S_j) back into the string at **any** position of your choice.

The complementary character of X is defined as below:

- If $X = (A, B)$ or (B, A), the complementary character is C.
- If $X = (B, C)$ or (C, B), the complementary character is A.
- If $X = (A, C)$ or (C, A), the complementary character is B.

You are allowed to do as many operations on S as you want. Find the lexicographically minimal[†] final string possible.

[†]String A is said to be lexicographically smaller than B if either of the following conditions hold:

- There exists i such that $1 \leq i \leq \min(|A|, |B|)$ and $A_i \neq B_i$. Consider X to be the smallest such i. $A_X < B_X$ holds.
- $|A| < |B|$ and for all $1 \leq i \leq |A|$, $A_i = B_i$.

That is, either A is a prefix of B, or at the first position where A and B differ, A's character is smaller.

Input Format

- The first line of input will contain a single integer T, denoting the number of test cases.
- Each test case consists of multiple lines of input:
  - The first line of each test case contains a single integer N - the length of the string.
  - The second line of each test case contains a string S.

Output Format

For each test case, output on a new line the lexicographically minimal string possible after applying operations on S.

Constraints

- $1 \leq T \leq 10^5$
- $1 \leq N \leq 2 \cdot 10^5$
- $S_i \in \{A, B, C\}$
- $|S| = N$
- The sum of N over all test cases does not exceed 10^6.

Samples

Sample Input 1

```
3
2
BC
3
AAA
3
CAA
```

Sample Output 1

```
A
AAA
AB
```

Sample Explanation

Test Case 1 : We use one operation with $i = 1, j = 2$. We delete S_1 and S_2 making the string empty temporarily. The complementary character is A, and we insert that into the string's 1st position. Thus, we end up with A.

Test Case 2 : No operations are possible.

Problem B
Bob Learns to Read

Time limit: 1 second
Memory limit: 1.5 Gigabytes

Problem Description

Bob is just learning to read. He has difficulty reading some complex words, so you want to help him.

Bob is given 2 words A and B, which he will read back to back (read A completely then read B completely). The difficulty for Bob arises whenever he reads a character that's not the same as the previous character he read.

For example, if Bob is reading only the word *icpc*, he will find it to have a difficulty of 3, while reading *iccc* only has a difficulty of 1.

We want to make sure the difficulty for Bob is as small as possible. To achieve this, we can arbitrarily rearrange the characters in both the words A and B. Find the minimum difficulty possible.

Input Format

- The first line of input will contain a single integer T, denoting the number of test cases.
- Each test case consists of two lines of input:
  - The first line of each test case contains a single word A – the first word Bob has to read.
  - The second line of each test case contains a single word B – the second word Bob has to read.

Output Format

For each test case, output the minimum difficulty possible.

Constraints

- $1 \leq T \leq 100$
- $1 \leq |A| \leq 20$
- $1 \leq |B| \leq 20$
- Each of the characters in A and B is a lower-case English character.

Samples

Sample Input 1

```
4
a
b
icpc
programming
kanpur
regional
alice
bob
```

Sample Output 1

```
1
9
12
6
```

Sample Explanation

Test Case 1: There is no rearrangement possible because both strings are of length 1. The difficulty is 1 as Bob will have to switch from a to b.

Test Case 4: The optimal rearrangement is `alice` and `obb`. Here the difficulty is 6 because there are 6 changes of characters.

Problem C
Check Good

Time limit: 5 seconds
Memory limit: 1.5 Gigabytes

Problem Description

An array B of length M is considered **good** if there exists a permutation P of $[1, 2, \ldots, M]$ such that for every i $(1 \leq i \leq M)$, the following condition holds:

$$B_i = \max(P_1, P_2, \ldots, P_i) - \min(P_1, P_2, \ldots, P_i)$$

You are given an array A of length N. For each integer i $(1 \leq i \leq N)$, determine whether there exists a **subarray**[†] X of A of length i such that X can be rearranged (shuffled) to form a **good** array.

[†] B is said to be a subarray of C if we can obtain B by deleting some prefix and some suffix of C (the prefix/suffix may be empty). For example, $[1, 4]$, $[1, 4, 2]$ and $[2]$ are all subarrays of $[1, 4, 2]$ but $[1, 2]$ is not.

Input Format

- The first line of input will contain a single integer T, denoting the number of test cases.
- Each test case consists of two lines of input:
  - The first line of each test case contains N - the size of the array.
  - The second line of each test case contains N integers - $A_1, A_2, \ldots, A_N$.

Output Format

For each test case, output a binary string S of length N. The i-th character of S (S_i) should be 1 if there exists a good subarray of length i, and 0 otherwise.

Constraints

- $1 \leq T \leq 10^4$
- $1 \leq N \leq 5 \cdot 10^5$
- $0 \leq A_i \leq N$
- It is guaranteed that the sum of N over all test cases does not exceed $5 \cdot 10^5$.

Samples

Sample Input 1

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

Sample Output 1

```
11
10011
1001
11000
000
```

Sample Explanation

Test Case 2: We can find length 1, 4, and 5 subarrays. Here are the examples for 1 and 4:

- We choose the subarray $C = [0]$, and $B = [0]$ as its rearrangement. Then B is good with respect to $P = [1]$.
- We choose the subarray $C = [3, 3, 0, 2]$, and $B = [0, 2, 3, 3]$ as its rearrangement. Then B is good with respect to $P = [1, 3, 4, 2]$.

Problem D
Hard Counting Problem

Time limit: 1 second
Memory limit: 1.5 Gigabytes

Problem Description

Alice had an array A of N elements, but for no reason whatsoever, a monster has started attacking the array. (Why? How? We don't know either!)

The monster does not like non-fixed indices (where a fixed index means that $A_i = i$). Thus, every second starting from 1, the monster deletes all elements simultaneously from the array at indices i satisfying $A_i \neq i$.
The remaining elements are then re-indexed starting from 1 without changing their order.
Then, the monster will again attack next second if there are more elements it can delete, and so on.

Alice wants to know when the monster will stop attacking her array. The monster will stop attacking when there are no more elements it can delete. The state of the array is called stable then. Note that an empty array is said to be stable as well.

Let $f(A)$ denote the time taken for the array to become stable. If the array was stable from the beginning, $f(A) = 0$.

Given integers N and M, find the sum of $f(A)$ over all M^N integer arrays satisfying $|A| = N$ and $1 \leq A_i \leq M$ (Array length is N and all elements are integers between 1 and M). Since the answer may be large, output it modulo 998244353.

Input Format

- The first line of input will contain a single integer T, denoting the number of test cases.
- The first and only line of input for each test case contains 2 integers N and M.

Output Format

For each test case, output on a new line the sum of $f(A)$ over all arrays of length N and elements bounded by M, modulo 998244353.

Constraints

- $1 \leq T \leq 10^4$
- $1 \leq N, M \leq 2 \cdot 10^5$
- The sum of N and the sum of M both do not exceed $2 \cdot 10^5$.

Samples

Sample Input 1

```
5
2 2
3 1
4 2
3 4
343 343
```

Sample Output 1

```
4
1
20
87
262200258
```

Sample Explanation

Test Case 1: There are 4 possible arrays, here are the computations:

- $A = [1, 2]$: This array is already stable. Hence, $f([1, 2]) = 0$.
- $A = [1, 1]$: Here, the monster will first delete the 2nd index because $A_2 = 1 \neq 2$. Then, the array becomes stable. Hence $f([1, 1]) = 1$.
- $A = [2, 1]$: Both the elements get deleted on the first attack. Thus, $f([2, 1]) = 1$.
- $A = [2, 2]$: First, the monster attacks the 1st index only. Then, the array becomes $[2]$ which is still not stable. The monster attacks the 1st index again, to finally get an empty array which is stable. Hence $f([2, 2]) = 2$.

The sum is $0 + 1 + 1 + 2 = 4$.

Problem E
Largest K

Time limit: 1 second
Memory limit: 1.5 Gigabytes

Problem Description

You have N 0's and N 1's with you. You want to construct a set of non-empty binary strings (say of size K) satisfying the following conditions:

- All the K strings are distinct.
- The total number of 0's in all strings is $\leq N$.
- The total number of 1's in all strings is $\leq N$.

Find the maximum K for which a valid set of strings exists.

Input Format

- The first line of input will contain a single integer T, denoting the number of test cases.
- The first and only line of input for each test case contains a single integer N.

Output Format

For each test case, output on a new line the maximum value of K for which a set of strings satisfying all constraints exists.

Constraints

- $1 \leq T \leq 10^3$
- $1 \leq N \leq 10^{18}$

Samples

Sample Input 1

```
3
1
2
5
```

Sample Output 1

```
2
3
6
```

Sample Explanation

Test Case 1: We can take the strings 0 and 1 to form a set of 2 distinct strings, using exactly one 0 and one 1.

Test Case 2: We can construct 3 strings by choosing $\{0, 1, 01\}$.

Problem F
Lexicographic Raffle

Time limit: 2 seconds
Memory limit: 1.5 Gigabytes

Problem Description

You have a string S of length N consisting of lower-case English characters.
The process `raffle`(L, R) for $1 \leq L < R \leq N$ is defined as follows:

- **Step 1**: Let X be the substring $S_L S_{L+1} \ldots S_{R-1}$ and Y be the substring $S_{L+1} S_{L+2} \ldots S_R$.
- **Step 2**: If Y is **lexicographically smaller**[†] than X, increment L by 1. Otherwise, decrement R by 1.
- **Step 3**: If $L = R$, terminate the process and return L as the result of the process; otherwise, go back to Step 1.

You are given Q queries, each of which contains two integers L and R $(1 \leq L \leq R \leq N)$. For each query, find the final value of L in the process `raffle`(L, R) defined as above.

[†] String A is said to be lexicographically smaller than B if either of the following conditions hold:

- There exists i such that $1 \leq i \leq \min(|A|, |B|)$ and $A_i \neq B_i$. Consider X to be the smallest such i. $A_X < B_X$ holds.
- $|A| < |B|$ and for all $1 \leq i \leq |A|$, $A_i = B_i$.

That is, either A is a prefix of B, or at the first position where A and B differ, A's character is smaller.

Input Format

- The first line of input will contain a single integer T, denoting the number of test cases.
- Each test case consists of multiple lines of input:
  - The first line of each test case contains N and Q - the length of the string and the number of queries.
  - The second line of each test case contains S - a string of size N.
  - The next Q lines each contain 2 integers L and R - representing a query.

Output Format

For each test case, for each query, output on a new line the final value of L after the process `raffle`(L, R).

Constraints

- $1 \leq T \leq 10^4$
- $2 \leq N \leq 2 \cdot 10^5$
- $1 \leq Q \leq 2 \cdot 10^5$
- $|S| = N$
- S contains only lower-case English characters.
- $1 \leq L < R \leq N$
- The sum of N and the sum of Q over all test cases both do not exceed $2 \cdot 10^5$.

Samples

Sample Input 1

```
4
6 3
kanpur
1 6
4 6
5 6
10 2
adccbabbab
1 10
2 9
5 3
accaa
1 5
2 4
3 5
5 1
jddda
3 4
```

Sample Output 1

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

Sample Explanation

Test Case 1: Here are the explanations for the first query:

- **Query 1**:
  - Initially, $L = 1, R = 6$; $X =$ `kanpu`, $Y =$ `anpur`. Since Y is lexicographically smaller, we increment L to 2. The process does not terminate here since $L \neq R$.
  - Now, $L = 2, R = 6$; $X =$ `anpu`, $Y =$ `npur`. Since Y is **not** lexicographically smaller, we decrement R to 5. The process does not terminate here since $L \neq R$.
  - Now, $L = 2, R = 5$; $X =$ `anp`, $Y =$ `npu`. Since Y is **not** lexicographically smaller, we decrement R to 4. The process does not terminate here since $L \neq R$.
  - Now, $L = 2, R = 4$; $X =$ `an`, $Y =$ `np`. Since Y is **not** lexicographically smaller, we decrement R to 3. The process does not terminate here since $L \neq R$.
  - Now, $L = 2, R = 3$; $X =$ `a`, $Y =$ `n`. Since Y is **not** lexicographically smaller, we decrement R to 2. The process terminates here as $L = R$.

Therefore, the final values are $L = R = 2$. Hence, the answer is 2.

Problem G
Majority Voters

Time limit: 1 second
Memory limit: 1.5 Gigabytes

Problem Description

There are N voters in a line numbered from 1 to N. They are voting in an election between 2 candidates, Alice and Bob. In order from 1 to N, they will each vote for either Alice or Bob.

The voters' preferences are represented by the string S: $S_i = A$ if the i-th voter was going to vote for Alice, and $S_i = B$ if the i-th voter was going to vote for Bob.

You want to rig the elections in favor of Alice. To do this, you have gained a special power. You can persuade people to change their stance to "majority voter". A majority voter numbered X will vote for the candidate who has already received strictly more votes (from previous voters 1 to $X - 1$). In case both candidates have an equal number of votes, a majority voter will not cast their vote at all.

Alice wins the election if and only if the number of voters for her is strictly greater than the number of voters for Bob.
Let $f(S)$ denote the minimum number of people whom you have to transform to majority voters, for Alice to win the election. If it is impossible for Alice to win no matter what, $f(S) = -1$.

Now, coming to the problem. You are given a binary string S of length N, and Q queries of the form:

- Given L and R, find $f(S[L, R])$ where $S[L, R]$ denotes the substring $S_L S_{L+1} \dots S_R$.

Input Format

- The first line of input will contain a single integer T, denoting the number of test cases.
- Each test case consists of multiple lines of input:
  - The first line of each test case contains 2 integers, N and Q - the length of the string and the number of queries.
  - The second line contains a binary string S of size N.
  - The next Q lines each contain 2 integers L and R - the parameters of each query.

Output Format

For each test case, for each query, output on a new line the value of $f(S[L, R])$.

Constraints

- $1 \leq T \leq 10^4$
- $1 \leq N, Q \leq 2 \cdot 10^5$
- $S_i \in \{A, B\}$
- $|S| = N$

- $1 \leq L \leq R \leq N$
- The sum of N and the sum of Q both do not exceed $2 \cdot 10^5$.

Samples

Sample Input 1

```
2
4 4
AABA
1 4
2 3
3 3
3 4
5 2
BBBBA
1 5
5 5
```

Sample Output 1

```
0
1
-1
1
4
0
```

Sample Explanation

Test Case 1: Here are the explanations for each query.

- **Query 1**: Alice is already winning as she has 3 voters voting for her while Bob has only 1.
- **Query 2**: Alice and Bob are tied with 1 vote each. If you change the Bob voter to a majority voter, Alice ends up winning because the majority voter will also vote for her (since the previous voter had voted for Alice).
- **Query 3**: Best you can do is change the only Bob voter to a majority voter, who will not vote for anyone at all; but that's not sufficient to make Alice win. Hence, the answer is -1.
- **Query 4**: Alice and Bob have one vote each. Changing the one Bob voter to a majority voter will result in Alice's victory, since the majority voter won't vote for anyone while Alice will still have one voter.

Problem H
P to Q

Time limit: 1 second
Memory limit: 1.5 Gigabytes

Problem Description

You are given 2 permutations P and Q of the integers 1 to N. Your score S is initialized as $\texttt{inversions}(P)^{\dagger}$.

You can do the following operation at most $10 \cdot N$ times:

- Select any integer x satisfying $1 \leq x \leq N$. Delete x from P, and then insert x back into P in any desired location. Replace S with $\max(S, \texttt{inversions}(P))$.

Your goal is to change P into Q while minimizing the final value of S. Also, print the operations performed.
Note that you **do not have to minimize** the number of operations performed.

---

† $\texttt{inversions}(P)$ represents the number of pairs (i, j) satisfying $1 \leq i < j \leq N$ and $P_i > P_j$.

Input Format

- The first line of input will contain a single integer T, denoting the number of test cases.
- Each test case consists of three lines of input:
  - The first line of each test case contains N - the permutation length.
  - The second line contains N integers - $P_1, P_2, \ldots, P_N$ representing the permutation P.
  - The third line contains N integers - $Q_1, Q_2, \ldots, Q_N$ representing the permutation Q.

Output Format

For each test case, the output is as follows:

- Firstly, on a new line, output K $(0 \leq K \leq 10 \cdot N)$ - the number of operations you wish to perform.
- Each of the next K lines must contain 2 space-separated integers:
  - x $(1 \leq x \leq N)$: the number we choose to delete and reinsert in this operation; and
  - pos $(1 \leq pos \leq N)$: the position where we reinsert x. Note there are $N - 1$ elements left after deleting x. $pos = i$ means we will reinsert x right before the i-th element, and $pos = N$ means that we will insert it at the end.

Constraints

- $1 \leq T \leq 100$
- $1 \leq N \leq 500$
- $1 \leq P_i, Q_i \leq N$
- $P_i \neq P_j$ for all $i \neq j$
- $Q_i \neq Q_j$ for all $i \neq j$
- The sum of N over all test cases does not exceed 500.

Samples

Sample Input 1

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

Sample Output 1

```
1
1 2
0
2
1 1
3 1
```

Sample Explanation

Test Case 1: Initially, $S = 0$ because $\texttt{inversions}([1, 2]) = 0$.
We perform one operation, which is to delete 1 and re-insert at the 2nd position.
This changes the permutation to $[2, 1]$ and S gets updated to $\max(0, \texttt{inversions}([2, 1])) = 1$.

We have successfully converted P to Q, and it can be proven that it is impossible to have a lower score.

Test Case 3: We show the steps:

- Initial conditions: $P = [2, 1, 3]$, $S = 1$.
- Delete 1 and reinsert at position 1: $P = [1, 2, 3]$, $S = 1$.
- Delete 3 and reinsert at position 1: $P = [3, 1, 2]$, $S = 2$.

Problem I
Prime Difference Graph

Time limit: 1 second
Memory limit: 1.5 Gigabytes

Problem Description

An array A consisting of N integers is said to be *good* if it satisfies each of the following conditions:

- All elements are positive integers, i.e., $A_i > 0$ for all $1 \leq i \leq N$.
- The array is strictly increasing, i.e., $A_i > A_{i-1}$ for all $2 \leq i \leq N$.
- Consider an undirected graph G on N nodes, where we add an edge between X and Y ($X \neq Y$) if and only if $|A_X - A_Y|$ is prime. G must be connected.

Find an array A satisfying all of the above conditions.
If multiple arrays exist, minimize $A_1 + A_2 + \ldots + A_N$. If multiple arrays still exist, any of them will be accepted.
It can be proven that at least one valid array A exists.

Input Format

- The first line of input will contain a single integer T, denoting the number of test cases.
- The first and only line of each test case contains N — the size of the array.

Output Format

For each test case, output on a new line N integers $A_1, A_2, \ldots, A_N$ satisfying all the conditions.

If multiple valid arrays with minimum sum exist, any of them will be accepted.

Constraints

- $1 \leq T \leq 50$
- $1 \leq N \leq 1000$
- The sum of N over all test cases does not exceed 1000.

Samples

Sample Input 1

```
2
1
2
```

Sample Output 1

```
1
1 3
```

Sample Explanation

Test Case 1: In the first test case, all arrays with $A_1 > 0$ are trivially valid, because an array of size 1 is always increasing, and a size 1 graph is connected. To minimize A_1, we choose $A_1 = 1$.

Test Case 2: The first 2 conditions are clearly satisfied. Also, an edge $(1, 2)$ exists in the graph G created because $|A_1 - A_2| = 2$ which is prime. Note that $[1, 2]$ does not satisfy the connected graph condition.

Problem J
Score Sum

Time limit: 1 second
Memory limit: 1.5 Gigabytes

Problem Description

Suppose we have an array A of N elements. We will define a *score* function on the array A in the following way.

Let $d(L, R)$ denote the number of distinct elements in the subarray $[A_L, A_{L+1}, \ldots, A_R]$.
Define $f(L, R) = (R - L + 1) - d(L, R)$, i.e., the length of the subarray from L to R minus the number of distinct elements in it.

Consider the pair (L, R) satisfying $1 \leq L \leq R \leq N$ with the maximum value of $f(L, R)$. If there are multiple such pairs, pick the one with the **minimum** value of $(R - L + 1)$. If there are still multiple such pairs, pick any.

Finally, we define $score(A)$ to be $(R - L + 1)$. That is, the length of the subarray which has the maximum f value and the minimum length.

Since finding the score was too easy, we added sum over all subarrays as a harder task.

Formally, given an array A of N elements, find the value

$$\sum_{L=1}^{N} \sum_{R=L}^{N} score(A[L, R])$$

where $A([L, R])$ represents the subarray $[A_L, A_{L+1}, \ldots, A_R]$.

Input Format

- The first line of input will contain a single integer T, denoting the number of test cases.
- Each test case consists of two lines of input:
  - The first line of each test case contains N - the size of the array A.
  - The second line of each test case contains N integers - $A_1, A_2, \ldots, A_N$.

Output Format

For each test case, output on a new line the sum of scores of all subarrays of A.

Constraints

- $1 \leq T \leq 10^4$
- $1 \leq N \leq 2 \cdot 10^5$
- $1 \leq A_i \leq N$
- The sum of N over all test cases does not exceed $2 \cdot 10^5$.

Samples

Sample Input 1

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

Sample Output 1

```
4
8
26
62
```

Sample Explanation

Test Case 1: There are 3 total subarrays - 2 repeated occurrences of [1] and 1 occurrence of $[1, 1]$.

- [1]: $score([1])$ is clearly just 1, because there is only one pair to choose $L = 1, R = 1$.
- $[1, 1]$: $L = 1, R = 2$ is the unique interval with the largest value of f. Hence, $score([1, 1]) = 2$.

The sum is $1 + 1 + 2 = 4$.

Problem K
Stalin Sort

Time limit: 1 second
Memory limit: 1.5 Gigabytes

Problem Description

You: Mom, can we have Stalin Sort?
Mom: We have Stalin Sort at home.
Stalin Sort at home:

Your Stalin Sort at home is even worse than the original Stalin Sort. Here is the algorithm:

- While the array is not sorted, delete a random element from it (equiprobably from all the elements).

You have a permutation P of length N. You want to apply this version of Stalin Sort to it. Find the expected length of the final array $\pmod{998244353}$.

Formally, it can be proven that the answer can be represented as $\frac{P}{Q}$ where P, Q are both integers and $Q \bmod 998244353 \neq 0$. Output the unique integer R in the range $[0, 998244353)$ such that $R \cdot Q \equiv P \pmod{998244353}$.

Input Format

- The first line of input will contain a single integer T, denoting the number of test cases.
- Each test case consists of two lines of input:
  - The first line of each test case contains a single integer N - the length of the initial array.
  - The second line contains N integers - $P_1, P_2, \ldots, P_N$, the initial permutation.

Output Format

For each test case, output on a new line the expected length of the final array modulo 998244353.

Constraints

- $1 \leq T \leq 50$
- $2 \leq N \leq 50$
- $1 \leq P_i \leq N$
- $P_i \neq P_j$ for all $i \neq j$
- The sum of N^2 over all test cases does not exceed 2500.

Samples

Sample Input 1

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

Sample Output 1

```
2
1
665496237
665496237
249561090
898419919
```

Sample Explanation

Test Case 1: The given array is already sorted. Hence, the expected length is simply the initial array length which is 2.

Test Case 2: The given array is unsorted; but after deleting either element, it becomes sorted. Hence, the length of the final sorted array is always 1.

Test Case 3: There are the following cases:

- Delete 1 in initial array: array becomes $[2, 3]$ which is sorted, length is 2.
- Delete 2 in initial array: array becomes $[1, 3]$ which is sorted, length is 2.
- Delete 3 in initial array: array becomes $[2, 1]$ which is not sorted. After one more step, it becomes sorted. Thus, length of final sorted array is 1.

Hence, the expected length is $\frac{(2+2+1)}{3} = \frac{5}{3}$, which is 665496237 mod 998244353.

Problem L
Yet Another MST Problem

Time limit: 1 second
Memory limit: 1.5 Gigabytes

Problem Description

You are given a connected undirected graph G with N nodes and M edges. Each edge has weight 1.
Some nodes are marked, and the other nodes are unmarked. You are given a binary string S where $S_i = 1$ if and only if the node i is marked.

Let X be the set of all marked nodes. It is guaranteed that X is non-empty. Construct a complete weighted graph H on the subset X. The weight of the edge between u and v is $\texttt{dist}(u, v)^{\dagger}$, where the distance is measured in the original graph G.

Find the total sum of weights of a minimum spanning tree in H.

---

† $\texttt{dist}(u, v)$ is the length of the shortest path between u and v. The length of a path is measured by the number of edges in the path.

Input Format

- The first line of input will contain a single integer T, denoting the number of test cases.
- Each test case consists of multiple lines of input:
  - The first line of each test case contains N and M - the number of nodes and the number of edges.
  - The second line contains a binary string S of size N.
  - Each of the next M lines contains 2 integers, u and v, representing an edge (u, v) in the graph G.

Output Format

For each test case, output on a new line the sum of weights of the minimum spanning tree of the constructed graph H.

Constraints

- $1 \leq T \leq 10^4$
- $2 \leq N \leq 2 \cdot 10^5$
- $(N - 1) \leq M \leq 2 \cdot 10^5$
- $S_i \in \{0, 1\}$
- $|S| = N$
- There exists at least one i such that $S_i = 1$
- $1 \leq u, v \leq N$
- $u \neq v$
- All the M pairs (u, v) are distinct.
- The given graph G is connected.
- The sum of N and the sum of M both do not exceed $2 \cdot 10^5$.

Samples

Sample Input 1

```
7
3 3
101
1 2
2 3
1 3
6 5
100101
1 2
2 3
3 4
3 5
5 6
4 3
0111
1 2
1 3
1 4
7 7
1110111
1 7
1 4
2 4
3 4
4 5
4 6
4 7
2 1
10
1 2
6 9
100111
2 5
4 3
3 5
5 1
1 6
4 2
4 5
6 3
2 3
6 8
110110
3 5
4 2
2 6
5 1
1 6
6 4
4 3
4 5
```

Sample Output 1

```
1
6
4
9
0
3
3
```

Sample Explanation

Test Case 1: There are 2 marked nodes 1 and 3. $\texttt{dist}(1,3) = 1$ as there is a direct edge $(1,3)$. Hence, the weight of the minimum spanning tree is simply 1.

Test Case 2: There are 3 marked nodes 1, 4, and 6. The weights of each edge are listed below:

- $\texttt{dist}(1,4) = 3$
- $\texttt{dist}(1,6) = 4$
- $\texttt{dist}(4,6) = 3$

The minimum spanning tree contains the 1st and the 3rd edges. Hence, the sum of weights is 6.