Codeforces Round #721 (Div. 2)

A. And Then There Were K

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Given an integer n, find the maximum value of integer k such that the following condition holds:

$$n \,\&\, (n-1) \,\&\, (n-2) \,\&\, (n-3) \,\&\, \ldots \,(k) = 0$$

where & denotes the bitwise AND operation.

Input

The first line contains a single integer t ($1 \le t \le 3 \cdot 10^4$). Then t test cases follow.

The first line of each test case contains a single integer n ($1 \le n \le 10^9$).

Output

For each test case, output a single integer — the required integer k.

Example

input
3 2 5 17
output
1 3 15

Note

In the first testcase, the maximum value for which the continuous & operation gives 0 value, is 1.

In the second testcase, the maximum value for which the continuous & operation gives 0 value, is 3. No value greater then 3, say for example 4, will give the & sum 0.

- $5 \,\&\, 4 \neq 0$,
- $5 \,\&\, 4 \,\&\, 3 = 0$.

Hence, 3 is the answer.

B1. Palindrome Game (easy version)

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

The only difference between the easy and hard versions is that the given string s in the easy version is initially a palindrome, this condition is not always true for the hard version.

A palindrome is a string that reads the same left to right and right to left. For example, "101101" is a palindrome, while "0101" is not.

Alice and Bob are playing a game on a string s **(which is initially a palindrome in this version)** of length n consisting of the characters '0' and '1'. Both players take alternate turns with Alice going first.

In each turn, the player can perform one of the following operations:

1. Choose any i ($1 \le i \le n$), where $s[i] =$ '0' and change $s[i]$ to '1'. Pay 1 dollar.
2. Reverse the whole string, pay 0 dollars. This operation is only allowed if the string is currently **not** a palindrome, and the last operation was not reverse. That is, if Alice reverses the string, then Bob can't reverse in the next move, and vice versa.

Reversing a string means reordering its letters from the last to the first. For example, "01001" becomes "10010" after reversing.

The game ends when every character of string becomes '1'. The player who spends minimum dollars till this point wins the game and it is a draw if both spend equal dollars. If both players play optimally, output whether Alice wins, Bob wins, or if it is a draw.

Input
The first line contains a single integer t ($1 \le t \le 10^3$). Then t test cases follow.

The first line of each test case contains a single integer n ($1 \le n \le 10^3$).

The second line of each test case contains the string s of length n, consisting of the characters '0' and '1'. It is guaranteed that the string s is a palindrome and contains at least one '0'.

Note that there is no limit on the sum of n over test cases.

Output
For each test case print a single word in a new line:

- "ALICE", if Alice will win the game,
- "BOB", if Bob will win the game,
- "DRAW", if the game ends in a draw.

Example

input
2 4 1001 1 0
output
BOB BOB

Note
In the first test case of the example,

- in the 1-st move Alice has to perform the 1-st operation, since the string is currently a palindrome.
- in the 2-nd move Bob reverses the string.
- in the 3-rd move Alice again has to perform the 1-st operation. All characters of the string are '1', game over.

Alice spends 2 dollars while Bob spends 0 dollars. Hence, Bob always wins.

B2. Palindrome Game (hard version)

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

The only difference between the easy and hard versions is that the given string s in the easy version is initially a palindrome, this condition is not always true for the hard version.

A palindrome is a string that reads the same left to right and right to left. For example, "101101" is a palindrome, while "0101" is not.

Alice and Bob are playing a game on a string s of length n consisting of the characters '0' and '1'. Both players take alternate turns with Alice going first.

In each turn, the player can perform one of the following operations:

1. Choose any i ($1 \le i \le n$), where $s[i] =$ '0' and change $s[i]$ to '1'. Pay 1 dollar.
2. Reverse the whole string, pay 0 dollars. This operation is only allowed if the string is currently **not** a palindrome, and the last operation was not reverse. That is, if Alice reverses the string, then Bob can't reverse in the next move, and vice versa.

Reversing a string means reordering its letters from the last to the first. For example, "01001" becomes "10010" after reversing.

The game ends when every character of string becomes '1'. The player who spends minimum dollars till this point wins the game and it is a draw if both spend equal dollars. If both players play optimally, output whether Alice wins, Bob wins, or if it is a draw.

Input
The first line contains a single integer t ($1 \le t \le 10^3$). Then t test cases follow.

The first line of each test case contains a single integer n ($1 \le n \le 10^3$).

The second line of each test case contains the string s of length n, consisting of the characters '0' and '1'. It is guaranteed that the string s contains at least one '0'.

Note that there is no limit on the sum of n over test cases.

Output
For each test case print a single word in a new line:

- "ALICE", if Alice will win the game,
- "BOB", if Bob will win the game,
- "DRAW", if the game ends in a draw.

Example

input
3 3 110 2 00 4 1010
output
ALICE BOB ALICE

Note
In the first test case of example,

- in the 1-st move, Alice will use the 2-nd operation to reverse the string, since doing the 1-st operation will result in her loss anyway. This also forces Bob to use the 1-st operation.
- in the 2-nd move, Bob has to perform the 1-st operation, since the 2-nd operation cannot be performed twice in a row. All characters of the string are '1', game over.

Alice spends 0 dollars while Bob spends 1 dollar. Hence, Alice wins.
In the second test case of example,

- in the 1-st move Alice has to perform the 1-st operation, since the string is currently a palindrome.
- in the 2-nd move Bob reverses the string.
- in the 3-rd move Alice again has to perform the 1-st operation. All characters of the string are '1', game over.

Alice spends 2 dollars while Bob spends 0 dollars. Hence, Bob wins.

C. Sequence Pair Weight

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The *weight* of a sequence is defined as the number of unordered pairs of indexes (i, j) (here $i < j$) with same value ($a_i = a_j$). For example, the weight of sequence $a = [1, 1, 2, 2, 1]$ is 4. The set of unordered pairs of indexes with same value are $(1, 2)$, $(1, 5)$, $(2, 5)$, and $(3, 4)$.

You are given a sequence a of n integers. Print the sum of the weight of all subsegments of a.

A sequence b is a subsegment of a sequence a if b can be obtained from a by deletion of several (possibly, zero or all) elements from the beginning and several (possibly, zero or all) elements from the end.

Input
Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^5$). Description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 10^5$).

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$).

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output
For each test case, print a single integer — the sum of the weight of all subsegments of a.

Example

input
2 4 1 2 1 1 4 1 2 3 4
output
6 0

Note

- In test case 1, all possible subsegments of sequence $[1, 2, 1, 1]$ having size more than 1 are:

  1. $[1, 2]$ having 0 valid unordered pairs;
  2. $[2, 1]$ having 0 valid unordered pairs;
  3. $[1, 1]$ having 1 valid unordered pair;
  4. $[1, 2, 1]$ having 1 valid unordered pairs;
  5. $[2, 1, 1]$ having 1 valid unordered pair;
  6. $[1, 2, 1, 1]$ having 3 valid unordered pairs.

  Answer is 6.
- In test case 2, all elements of the sequence are distinct. So, there is no valid unordered pair with the same value for any subarray. Answer is 0.

D. MEX Tree

time limit per test: 1.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a tree with n nodes, numerated from 0 to $n-1$. For each k between 0 and n, inclusive, you have to count the number of unordered pairs (u, v), $u \neq v$, such that the **MEX** of all the node labels in the shortest path from u to v (including end points) is k.

The **MEX** of a sequence of integers is the smallest non-negative integer that does not belong to the sequence.

Input

The first line contains a single integer t ($1 \leq t \leq 10^4$) — the number of test cases.

The first line of each test case contains a single integer n ($2 \leq n \leq 2 \cdot 10^5$).

The next $n-1$ lines of each test case describe the tree that has to be constructed. These lines contain two integers u and v ($0 \leq u, v \leq n-1$) denoting an edge between u and v ($u \neq v$).

It is guaranteed that the given edges form a tree.

It is also guaranteed that the sum of n for all test cases does not exceed $2 \cdot 10^5$.

Output

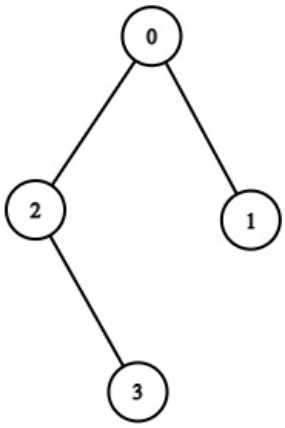
For each test case, print $n+1$ integers: the number of paths in the tree, such that the MEX of all the node labels in that path is k for each k from 0 to n.

Example

input
2 4 0 1 0 2 2 3 2 1 0
output
1 2 1 1 1 0 0 1

Note

1. In example case 1,



- For $k = 0$, there is 1 path that is from 2 to 3 as $MEX([2, 3]) = 0$.
- For $k = 1$, there are 2 paths that is from 0 to 2 as $MEX([0, 2]) = 1$ and 0 to 3 as $MEX([0, 2, 3]) = 1$.
- For $k = 2$, there is 1 path that is from 0 to 1 as $MEX([0, 1]) = 2$.
- For $k = 3$, there is 1 path that is from 1 to 2 as $MEX([1, 0, 2]) = 3$
- For $k = 4$, there is 1 path that is from 1 to 3 as $MEX([1, 0, 2, 3]) = 4$.

2. In example case 2,

- For $k = 0$, there are no such paths.
- For $k = 1$, there are no such paths.
- For $k = 2$, there is 1 path that is from 0 to 1 as $MEX([0, 1]) = 2$.

E. Partition Game

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given an array a of n integers. Define the cost of some array t as follows:

$$cost(t) = \sum_{x \in set(t)} last(x) - first(x),$$

where $set(t)$ is the set of all values in t without repetitions, $first(x)$, and $last(x)$ are the indices of the first and last occurrence of x in t, respectively. In other words, we compute the distance between the first and last occurrences for each distinct element and sum them up.

You need to split the array a into k consecutive segments such that each element of a belongs to exactly one segment and the sum of the cost of individual segments is minimum.

Input

The first line contains two integers n, k ($1 \le n \le 35\,000$, $1 \le k \le \min(n, 100)$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$).

Output

Output the minimum sum of the cost of individual segments.

Examples

input
7 2 1 6 6 4 6 6 6
output
3

input
7 4 5 5 5 5 2 3 3
output
1

Note

In the first example, we can divide the array into $[1, 6, 6, 4]$ and $[6, 6, 6]$. Cost of $[1, 6, 6, 4]$ will be $(1 - 1) + (3 - 2) + (4 - 4) = 1$ and cost of $[6, 6, 6]$ will be $3 - 1 = 2$. Total cost would be $1 + 2 = 3$.

In the second example, divide the array into $[5, 5]$, $[5]$, $[5, 2, 3]$ and $[3]$. Total Cost would be $1 + 0 + 0 + 0 = 1$.