# Codeforces Round #780 (Div. 3)

A. Vasya and Coins

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

Vasya decided to go to the grocery store. He found in his wallet aa coins of 11 burle and bb coins of 22 burles. He does not yet know the total cost of all goods, so help him find out ss (s>0s>0): the minimum positive integer amount of money he cannot pay without change or pay at all using only his coins.

For example, if a=1a=1 and b=1b=1 (he has one 11-burle coin and one 22-burle coin), then:

he can pay 11 burle without change, paying with one 11-burle coin,

he can pay 22 burle without change, paying with one 22-burle coin,

he can pay 33 burle without change by paying with one 11-burle coin and one 22-burle coin,

he cannot pay 44 burle without change (moreover, he cannot pay this amount at all).

So for a=1a=1 and b=1b=1 the answer is s=4s=4.

Input

The first line of the input contains an integer tt (1≤t≤1041≤t≤104) — the number of test cases in the test.

The description of each test case consists of one line containing two integers aiai and bibi (0≤ai,bi≤1080≤ai,bi≤108) — the number of 11-burle coins and 22-burles coins Vasya has respectively.

Output

For each test case, on a separate line print one integer ss (s>0s>0): the minimum positive integer amount of money that Vasya cannot pay without change or pay at all.

Example

input

Copy

5

1 1

4 0

0 2

0 0

2314 2374

output

Copy

4

5

1

1

7063

Note

The first test case of the example is clarified into the main part of the statement.

In the second test case, Vasya has only 11 burle coins, and he can collect either any amount from 11 to 44, but 55 can't.

In the second test case, Vasya has only 22 burle coins, and he cannot pay 11 burle without change.

In the fourth test case you don't have any coins, and he can't even pay 11 burle.

B. Vlad and Candies

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

Not so long ago, Vlad had a birthday, for which he was presented with a package of candies. There were nn types of candies, there are aiai candies of the type ii (1≤i≤n1≤i≤n).

Vlad decided to eat exactly one candy every time, choosing any of the candies of a type that is currently the most frequent (if there are several such types, he can choose any of them). To get the maximum pleasure from eating, Vlad does not want to eat two candies of the same type in a row.

Help him figure out if he can eat all the candies without eating two identical candies in a row.

Input

The first line of input data contains an integer tt (1≤t≤1041≤t≤104) — the number of input test cases.

The following is a description of tt test cases of input, two lines for each.

The first line of the case contains the single number nn (1≤n≤2⋅1051≤n≤2⋅105) — the number of types of candies in the package.

The second line of the case contains nn integers aiai (1≤ai≤1091≤ai≤109) — the number of candies of the type ii.

It is guaranteed that the sum of nn for all cases does not exceed 2⋅1052⋅105.

Output

Output tt lines, each of which contains the answer to the corresponding test case of input. As an answer, output "YES" if Vlad can eat candy as planned, and "NO" otherwise.

You can output the answer in any case (for example, the strings "yEs", "yes", "Yes" and "YES" will be recognized as a positive answer).

Example

input

Copy

6

2

2 3

1

2

5

1 6 2 4 3

4

2 2 2 1

3

1 1000000000 999999999

1

1

output

Copy

YES

NO

NO

YES

YES

YES

Note

In the first example, it is necessary to eat sweets in this order:

a candy of the type 22, it is the most frequent, now a=[2,2]a=[2,2];

a candy of the type 11, there are the same number of candies of the type 22, but we just ate one, now a=[1,2]a=[1,2];

a candy of the type 22, it is the most frequent, now a=[1,1]a=[1,1];

a candy of the type 11, now a=[0,1]a=[0,1];

a candy of the type 22, now a=[0,0]a=[0,0] and the candy has run out.

In the second example, all the candies are of the same type and it is impossible to eat them without eating two identical ones in a row.

In the third example, first of all, a candy of the type 22 will be eaten, after which this kind will remain the only kind that is the most frequent, and you will have to eat a candy of the type 22 again.

C. Get an Even String

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

A string a=a1a2…ana=a1a2…an is called even if it consists of a concatenation (joining) of strings of length 22 consisting of the same characters. In other words, a string aa is even if two conditions are satisfied at the same time:

its length nn is even;

for all odd ii (1≤i≤n−11≤i≤n−1), ai=ai+1ai=ai+1 is satisfied.

For example, the following strings are even: "" (empty string), "tt", "aabb", "oooo", and "ttrrrroouuuuuuuukk". The following strings are not even: "aaa", "abab" and "abba".

Given a string ss consisting of lowercase Latin letters. Find the minimum number of characters to remove from the string ss to make it even. The deleted characters do not have to be consecutive.

Input

The first line of input data contains an integer tt (1≤t≤1041≤t≤104) —the number of test cases in the test.

The descriptions of the test cases follow.

Each test case consists of one string ss (1≤|s|≤2⋅1051≤|s|≤2⋅105), where |s||s| — the length of the string ss. The string consists of lowercase Latin letters.

It is guaranteed that the sum of |s||s| on all test cases does not exceed 2⋅1052⋅105.

Output

For each test case, print a single number — the minimum number of characters that must be removed to make ss even.

Example

input

Copy

6

aabbdabdccc

zyx

aaababbb

aabbcc

oaoaaaoo

bmefbmuyw

output

Copy

3

3

2

0

2

7

Note

In the first test case you can remove the characters with indices 66, 77, and 99 to get an even string "aabbddcc".

In the second test case, each character occurs exactly once, so in order to get an even string, you must remove all characters from the string.

In the third test case, you can get an even string "aaaabb" by removing, for example, 44-th and 66-th characters, or a string "aabbbb" by removing the 55-th character and any of the first three.

D. Maximum Product Strikes Back

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

You are given an array aa consisting of nn integers. For each ii (1≤i≤n1≤i≤n) the following inequality is true: −2≤ai≤2−2≤ai≤2.

You can remove any number (possibly 00) of elements from the beginning of the array and any number (possibly 00) of elements from the end of the array. You are allowed to delete the whole array.

You need to answer the question: how many elements should be removed from the beginning of the array, and how many elements should be removed from the end of the array, so that the result will be an array whose product (multiplication) of elements is maximal. If there is more than one way to get an array with the maximum product of elements on it, you are allowed to output any of them.

The product of elements of an empty array (array of length 00) should be assumed to be 11.

Input

The first line of input data contains an integer tt (1≤t≤1041≤t≤104) —the number of test cases in the test.

Then the descriptions of the input test cases follow.

The first line of each test case description contains an integer nn (1≤n≤2⋅1051≤n≤2⋅105) —the length of array aa.

The next line contains nn integers a1,a2,…,ana1,a2,…,an (|ai|≤2|ai|≤2) — elements of array aa.

It is guaranteed that the sum of nn over all test cases does not exceed 2⋅1052⋅105.

Output

For each test case, output two non-negative numbers xx and yy (0≤x+y≤n0≤x+y≤n) — such that the product (multiplication) of the array numbers, after removing xx elements from the beginning and yy elements from the end, is maximal.

If there is more than one way to get the maximal product, it is allowed to output any of them. Consider the product of numbers on empty array to be 11.

Example

input

Copy

5

4

1 2 -1 2

3

1 1 -2

5

2 0 -2 2 -1

3

-2 -1 -1

3

-1 -2 -2

output

Copy

0 2

3 0

2 0

0 1

1 0

Note

In the first case, the maximal value of the product is 22. Thus, we can either delete the first three elements (obtain array [2][2]), or the last two and one first element (also obtain array [2][2]), or the last two elements (obtain array [1,2][1,2]). Thus, in the first case, the answers fit: "3 0", or "1 2", or "0 2".

In the second case, the maximum value of the product is 11. Then we can remove all elements from the array because the value of the product on the empty array will be 11. So the answer is "3 0", but there are other possible answers.

In the third case, we can remove the first two elements of the array. Then we get the array: [−2,2,−1][−2,2,−1]. The product of the elements of the resulting array is (−2)⋅2⋅(−1)=4(−2)⋅2⋅(−1)=4. This value is the maximum possible value that can be obtained. Thus, for this case the answer is: "2 0".

E. Matrix and Shifts

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

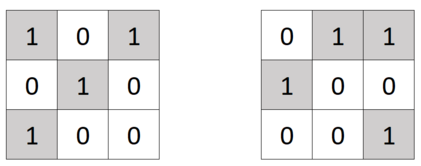
You are given a binary matrix AA of size n×nn×n. Rows are numbered from top to bottom from 11 to nn, columns are numbered from left to right from 11 to nn. The element located at the intersection of row ii and column jj is called AijAij. Consider a set of 44 operations:

Cyclically shift all rows up. The row with index ii will be written in place of the row i−1i−1 (2≤i≤n2≤i≤n), the row with index 11 will be written in place of the row nn.

Cyclically shift all rows down. The row with index ii will be written in place of the row i+1i+1 (1≤i≤n−11≤i≤n−1), the row with index nn will be written in place of the row 11.

Cyclically shift all columns to the left. The column with index jj will be written in place of the column j−1j−1 (2≤j≤n2≤j≤n), the column with index 11 will be written in place of the column nn.

Cyclically shift all columns to the right. The column with index jj will be written in place of the column j+1j+1 (1≤j≤n−11≤j≤n−1), the column with index nn will be written in place of the column 11.

The 3×33×3 matrix is shown on the left before the 33-rd operation is applied to it, on the right — after.

You can perform an arbitrary (possibly zero) number of operations on the matrix; the operations can be performed in any order.

After that, you can perform an arbitrary (possibly zero) number of new xor-operations:

Select any element AijAij and assign it with new value Aij⊕1Aij⊕1. In other words, the value of (Aij+1)mod2(Aij+1)mod2 will have to be written into element AijAij.

Each application of this xor-operation costs one burl. Note that the 44 shift operations — are free. These 44 operations can only be performed before xor-operations are performed.

Output the minimum number of burles you would have to pay to make the AA matrix unitary. A unitary matrix is a matrix with ones on the main diagonal and the rest of its elements are zeros (that is, Aij=1Aij=1 if i=ji=j and Aij=0Aij=0 otherwise).

Input

The first line of the input contains an integer tt (1≤t≤1041≤t≤104) —the number of test cases in the test.

The descriptions of the test cases follow. Before each test case, an empty line is written in the input.

The first line of each test case contains a single number nn (1≤n≤20001≤n≤2000)

This is followed by nn lines, each containing exactly nn characters and consisting only of zeros and ones. These lines describe the values in the elements of the matrix.

It is guaranteed that the sum of n2n2 values over all test cases does not exceed 4⋅1064⋅106.

Output

For each test case, output the minimum number of burles you would have to pay to make the AA matrix unitary. In other words, print the minimum number of xor-operations it will take after applying cyclic shifts to the matrix for the AA matrix to become unitary.

Example

input

Copy

4

3

010

011

100

5

00010

00001

10000

01000

00100

2

10

10

4

1111

1011

1111

1111

output

Copy

1

0

2

11

Note

In the first test case, you can do the following: first, shift all the rows down cyclically, then the main diagonal of the matrix will contain only "1". Then it will be necessary to apply xor-operation to the only "1" that is not on the main diagonal.

In the second test case, you can make a unitary matrix by applying the operation 22 — cyclic shift of rows upward twice to the matrix.

F1. Promising String (easy version)

time limit per test

3 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

This is the easy version of Problem F. The only difference between the easy version and the hard version is the constraints.

We will call a non-empty string balanced if it contains the same number of plus and minus signs. For example: strings "+--+" and "++-+--" are balanced, and strings "+--", "--" and "" are not balanced.

We will call a string promising if the string can be made balanced by several (possibly zero) uses of the following operation:

replace two adjacent minus signs with one plus sign.

In particular, every balanced string is promising. However, the converse is not true: not every promising string is balanced.

For example, the string "-+---" is promising, because you can replace two adjacent minuses with plus and get a balanced string "-++-", or get another balanced string "-+-+".

How many non-empty substrings of the given string ss are promising? Each non-empty promising substring must be counted in the answer as many times as it occurs in string ss.

Recall that a substring is a sequence of consecutive characters of the string. For example, for string "+-+" its substring are: "+-", "-+", "+", "+-+" (the string is a substring of itself) and some others. But the following strings are not its substring: "--", "++", "-++".

Input

The first line of the input contains an integer tt (1≤t≤5001≤t≤500) —the number of test cases in the test.

Then the descriptions of test cases follow.

Each test case of input data consists of two lines. The first line consists of the number nn (1≤n≤30001≤n≤3000): the length of ss.

The second line of the test case contains the string ss of length nn, consisting only of characters "+" and "-".

It is guaranteed that the sum of values nn over all test cases does not exceed 30003000.

Output

For each test case, print a single number: the number of the promising non-empty substrings of string ss. Each non-empty promising substring must be counted in the answer as many times as it occurs in string ss.

Example

input

Copy

5

3

+-+

5

-+---

4

----

7

--+---+

6

+++---

output

Copy

2

4

2

7

4

Note

The following are the promising substrings for the first three test cases in the example:

s[1…2]s[1…2]="+-", s[2…3]s[2…3]="-+";

s[1…2]s[1…2]="-+", s[2…3]s[2…3]="+-", s[1…5]s[1…5]="-+---", s[3…5]s[3…5]="---";

s[1…3]s[1…3]="---", s[2…4]s[2…4]="---".

F2. Promising String (hard version)

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

This is the hard version of Problem F. The only difference between the easy version and the hard version is the constraints.

We will call a non-empty string balanced if it contains the same number of plus and minus signs. For example: strings "+--+" and "++-+--" are balanced, and strings "+--", "--" and "" are not balanced.

We will call a string promising if the string can be made balanced by several (possibly zero) uses of the following operation:

replace two adjacent minus signs with one plus sign.

In particular, every balanced string is promising. However, the converse is not true: not every promising string is balanced.

For example, the string "-+---" is promising, because you can replace two adjacent minuses with plus and get a balanced string "-++-", or get another balanced string "-+-+".

How many non-empty substrings of the given string ss are promising? Each non-empty promising substring must be counted in the answer as many times as it occurs in string ss.

Recall that a substring is a sequence of consecutive characters of the string. For example, for string "+-+" its substring are: "+-", "-+", "+", "+-+" (the string is a substring of itself) and some others. But the following strings are not its substring: "--", "++", "-++".

Input

The first line of the input contains an integer tt (1≤t≤1041≤t≤104) —the number of test cases in the test.

Then the descriptions of test cases follow.

Each test case of input data consists of two lines. The first line consists of the number nn (1≤n≤2⋅1051≤n≤2⋅105): the length of ss.

The second line of the test case contains the string ss of length nn, consisting only of characters "+" and "-".

It is guaranteed that the sum of values nn over all test cases does not exceed 2⋅1052⋅105.

Output

For each test case, print a single number: the number of the promising non-empty substrings of string ss. Each non-empty promising substring must be counted in the answer as many times as it occurs in string ss.

Example

input

Copy

5

3

+-+

5

-+---

4

----

7

--+---+

6

+++---

output

Copy

2

4

2

7

4

Note

The following are the promising substrings for the first three test cases in the example:

s[1…2]s[1…2]="+-", s[2…3]s[2…3]="-+";

s[1…2]s[1…2]="-+", s[2…3]s[2…3]="+-", s[1…5]s[1…5]="-+---", s[3…5]s[3…5]="---";

s[1…3]s[1…3]="---", s[2…4]s[2…4]="---".