Codeforces Round 855 (Div. 3)

A. Is It a Cat?

2 seconds, 256 megabytes

You were walking down the street and heard a sound. The sound was described by the string s consisting of lowercase and uppercase Latin characters. Now you want to find out if the sound was a cat meowing.

For the sound to be a meowing, the string can only contain the letters 'm', 'e', 'o' and 'w', in either uppercase or lowercase. Also:

- string must start with non-empty sequence consisting only of characters 'm' or 'M'
- it must be immediately followed by non-empty sequence consisting only of characters 'e' or 'E'
- it must be immediately followed by non-empty sequence consisting only of characters 'o' or 'O'
- it must be immediately followed by non-empty sequence consisting only of characters 'w' or 'W', this sequence ends the string, after it immediately comes the string end

For example, strings "meow", "mmmEeOWww", "MeOooOw" describe a meowing, but strings "Mweo", "MeO", "moew", "MmEW", "meowmeow" do not.

Determine whether the sound you heard was a cat meowing or something else.

Input

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

The description of the test cases follows.

The first line of each test case contains an integer n ($1 \le n \le 50$) — the length of the string describing the sound.

The second line of each test case contains a string s of n characters. The string describes the sound you heard and consists only of lowercase and uppercase Latin letters.

Output

For each test case, output on a separate line:

- YES if the sound was a cat meowing;
- NO otherwise.

You can output YES and NO in any case (for example, strings yEs, yes, Yes and YES will be recognized as positive response).

input

```
7
4
meOw
14
mMmeoOoWWWwwwW
3
mew
7
MmeEeUw
4
MEOW
6
MmyaVW
5
meowA
```

output

```
YES
YES
NO
NO
YES
NO
NO
```

In the first test case, the string consists of a sequence of characters 'm', 'e', 'O', 'w', which satisfies the definition of meowing.

In the second test case, the string consists of a sequence of 3 characters 'm' and 'M', one 'e', a sequence of 3 characters 'o' and 'O' and a sequence of 7 characters 'w' and 'W', which satisfies the definition of meowing.

In the third test case, the string does not describe a meowing because it lacks a sequence of 'o' or 'O' characters between 'e' and 'w'.

In the fourth test case, the string contains the character 'U', so it does not describe a meowing.

B. Count the Number of Pairs

2 seconds, 256 megabytes

Kristina has a string s of length n, consisting only of lowercase and uppercase Latin letters. For each pair of lowercase letter and its matching uppercase letter, Kristina can get 1 burl. However, pairs of characters cannot overlap, so each character can only be in one pair.

For example, if she has the string s = "aAaaBACacbE", she can get a burl for the following character pairs:

- s_1 = "a" and s_2 = "A"
- s_4 = "a" and s_6 = "A"
- s_5 = "B" and s_{10} = "b"
- s_7 = "C" and s_9 = "c"

Kristina wants to get more burles for her string, so she is going to perform no more than k operations on it. In one operation, she can:

- either select the lowercase character s_i ($1 \le i \le n$) and make it uppercase.
- or select uppercase character s_i ($1 \le i \le n$) and make it lowercase.

For example, when k = 2 and s = "aAaaBACacbE" it can perform one operation: choose s_3 = "a" and make it uppercase. Then she will get another pair of s_3 = "A" and s_8 = "a"

Find **maximum** number of burles Kristina can get for her string.

Input

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

The description of the test cases follows.

The first line of each test case contains two integers n ($1 \le n \le 2 \cdot 10^5$) and k ($0 \le k \le n$) — the number of characters in the string and the maximum number of operations that can be performed on it.

The second line of each test case contains a string s of length n, consisting only of lowercase and uppercase Latin letters.

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

Output

For each test case, print exactly one integer on a separate line: the maximum number of burles that Kristina can get for her string s.

input

```
5
11 2
aAaaBACacbE
2 2
ab
4 1
aaBB
6 0
abBAcC
5 3
cbccb
```

output

```
5
0
1
3
2
```

The first test case is explained in the problem statement.

In the second test case, it is not possible to get any pair by performing any number of operations.

C1. Powering the Hero (easy version)

2 seconds, 256 megabytes

This is an easy version of the problem. It differs from the hard one only by constraints on n and t.

There is a deck of n cards, each of which is characterized by its power. There are two types of cards:

- a hero card, the power of such a card is always equal to 0;
- a bonus card, the power of such a card is always positive.

You can do the following with the deck:

- take a card from the top of the deck;
- if this card is a bonus card, you can put it **on top** of your bonus deck or discard;
- if this card is a hero card, then the power of **the top** card from your bonus deck is added to his power (if it is not empty), after that the hero is added to your army, and the used bonus discards.

Your task is to use such actions to gather an army with the maximum possible total power.

Input

The first line of input data contains single integer t ($1 \le t \le 1000$) — the number of test cases in the test.

The first line of each test case contains one integer n ($1 \le n \le 5000$) — the number of cards in the deck.

The second line of each test case contains n integers $s_1, s_2, \ldots, s_n$ ($0 \le s_i \le 10^9$) — card powers in top-down order.

It is guaranteed that the sum of n over all test cases does not exceed 5000.

Output

Output t numbers, each of which is the answer to the corresponding test case — the maximum possible total power of the army that can be achieved.

input

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

output

```
6
6
8
9
4
```

In the first sample, you can take bonuses 1 and 2. Both hero cards will receive 3 power. If you take all the bonuses, one of them will remain unused.

In the second sample, the hero's card on top of the deck cannot be powered up, and the rest can be powered up with 2 and 3 bonuses and get 6 total power.

In the fourth sample, you can take bonuses $1, 2, 3, 5$ and skip the bonus 6, then the hero 4 will be enhanced with a bonus 3 by 5, and the hero 7 with a bonus 5 by 4. $4 + 5 = 9$.

C2. Powering the Hero (hard version)

2 seconds, 256 megabytes

This is a hard version of the problem. It differs from the easy one only by constraints on n and t.

There is a deck of n cards, each of which is characterized by its power. There are two types of cards:

- a hero card, the power of such a card is always equal to 0;
- a bonus card, the power of such a card is always positive.

You can do the following with the deck:

- take a card from the top of the deck;
- if this card is a bonus card, you can put it **on top** of your bonus deck or discard;
- if this card is a hero card, then the power of **the top** card from your bonus deck is added to his power (if it is not empty), after that the hero is added to your army, and the used bonus discards.

Your task is to use such actions to gather an army with the maximum possible total power.

Input

The first line of input data contains single integer t ($1 \le t \le 10^4$) — the number of test cases in the test.

The first line of each test case contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of cards in the deck.

The second line of each test case contains n integers $s_1, s_2, \ldots, s_n$ ($0 \le s_i \le 10^9$) — card powers in top-down order.

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

Output

Output t numbers, each of which is the answer to the corresponding test case — the maximum possible total power of the army that can be achieved.

input

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

output

```
6
6
8
9
4
```

In the first sample, you can take bonuses 1 and 2. Both hero cards will receive 3 power. If you take all the bonuses, one of them will remain unused.

In the second sample, the hero's card on top of the deck cannot be powered up, and the rest can be powered up with 2 and 3 bonuses and get 6 total power.

In the fourth sample, you can take bonuses 1, 2, 3, 5 and skip the bonus 6, then the hero 4 will be enhanced with a bonus 3 by 5, and the hero 7 with a bonus 5 by 4. $4 + 5 = 9$.

D. Remove Two Letters

2 seconds, 256 megabytes

Dmitry has a string s, consisting of lowercase Latin letters.

Dmitry decided to remove two **consecutive** characters from the string s and you are wondering how many different strings can be obtained after such an operation.

For example, Dmitry has a string "aaabcc". You can get the following different strings: "abcc"(by deleting the first two or second and third characters), "aacc"(by deleting the third and fourth characters),"aaac" (by deleting the fourth and the fifth character) and "aaab" (by deleting the last two).

Input

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

The descriptions of the test cases follow.

The first line of the description of each test case contains an integer n ($3 \le n \le 2 \cdot 10^5$).

The second line of the description of each test case contains a string s of length n consisting of lowercase Latin letters.

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

Output

For each test case print one integer — the number of distinct strings that can be obtained by removing two consecutive letters.

input

```
7
6
aaabcc
10
aaaaaaaaaa
6
abcdef
7
abacaba
6
cccfff
4
abba
5
ababa
```

output

```
4
1
5
3
3
3
1
```

The first example is explained in the statement.

In the third example, the following strings are obtained: "cdef", "adef", "abef", "abcf", "abcd".

In the seventh example, any deletion will result in the string "aba".

E1. Unforgivable Curse (easy version)

1 second, 256 megabytes

This is an easy version of the problem. In this version, k is always 3.

The chief wizard of the Wizengamot once caught the evil wizard Drahyrt, but the evil wizard has returned and wants revenge on the chief wizard. So he stole *spell* s from his student Harry.

The *spell* — is a n-length string of lowercase Latin letters.

Drahyrt wants to replace *spell* with an unforgivable curse — string t.

Drahyrt, using ancient magic, can swap letters at a distance k or $k + 1$ in *spell* as many times as he wants. *In this version of the problem, you can swap letters at a distance of* 3 *or* 4. In other words, Drahyrt can change letters in positions i and j in *spell* s if $|i - j| = 3$ or $|i - j| = 4$.

For example, if s = "talant" and t = "atltna", Drahyrt can act as follows:

- swap the letters at positions 1 and 4 to get *spell* "aaltnt".
- swap the letters at positions 2 and 6 to get *spell* "atltna".

You are given *spells* s and t. Can Drahyrt change *spell* s to t?

Input

The first line of input gives a single integer T ($1 \le T \le 10^4$) — the number of test cases in the test.

Descriptions of the test cases are follow.

The first line contains two integers n, k ($1 \le n \le 2 \cdot 10^5$, $k = 3$) — the length *spells* and the number k such that Drahyrt can change letters in a spell at a distance k or $k + 1$.

The second line gives *spell* s — a string of length n consisting of lowercase Latin letters.

The third line gives *spell* t — a string of length n consisting of lowercase Latin letters.

It is guaranteed that the sum of n values over all test cases does not exceed $2 \cdot 10^5$. Note that there is no limit on the sum of k values over all test cases.

Output

For each test case, output on a separate line "YES" if Drahyrt can change *spell* s to t and "NO" otherwise.

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

input
7 6 3 talant atltna 7 3 abacaba aaaabbc 12 3 abracadabraa avadakedavra 5 3 accio cicao 5 3 lumos molus 4 3 uwjt twju 4 3 kvpx vxpk

output
YES YES NO YES NO YES NO

The first example is explained in the condition.

In the second example we can proceed as follows:

- Swap the letters at positions 2 and 5 (distance 3), then we get the spell "aaacbba".
- Swap the letters at positions 4 and 7 (distance 3), then you get the spell "aaaabbc".

In the third example, we can show that it is impossible to get the string t from the string s by swapping the letters at a distance of 3 or 4.

In the fourth example, for example, the following sequence of transformations is appropriate:

- "accio" → "aocic" → "cocia" → "iocca" → "aocci" → "aicco" → "cicao"

In the fifth example, you can show that it is impossible to get the string s from the string t.

In the sixth example, it is enough to swap the two outermost letters.

E2. Unforgivable Curse (hard version)

1 second, 256 megabytes

This is a complex version of the problem. This version has no additional restrictions on the number k.

The chief wizard of the Wizengamot once caught the evil wizard Drahyrt, but the evil wizard has returned and wants revenge on the chief wizard. So he stole *spell* s from his student Harry.

The *spell* — is a n-length string of lowercase Latin letters.

Drahyrt wants to replace *spell* with an unforgivable curse — string t.

Dragirt, using ancient magic, can swap letters at a distance k or $k+1$ in *spell* as many times as he wants. In other words, Drahyrt can change letters in positions i and j in *spell* s if $|i-j|=k$ or $|i-j|=k+1$.

For example, if $k=3, s=$ "talant" and $t=$ "atltna", Drahyrt can act as follows:

- swap the letters at positions 1 and 4 to get *spell* "aaltnt".
- swap the letters at positions 2 and 6 to get *spell* "atltna".

You are given *spells* s and t. Can Drahyrt change *spell* s to t?

Input

The first line of input gives a single integer T ($1 \le T \le 10^4$) — the number of test cases in the test.

Descriptions of the test cases are follow.

The first line contains two integers n, k ($1 \le n \le 2 \cdot 10^5$, $1 \le k \le 2 \cdot 10^5$) — the length *spells* and the number k such that Drahyrt can change letters in a spell at a distance k or $k+1$.

The second line gives *spell* s — a string of length n consisting of lowercase Latin letters.

The third line gives *spell* t — a string of length n consisting of lowercase Latin letters.

It is guaranteed that the sum of n values over all test cases does not exceed $2 \cdot 10^5$. Note that there is no limit on the sum of k values over all test cases.

Output

For each test case, output on a separate line "YES" if Drahyrt can change *spell* s to t and "NO" otherwise.

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

input
7 6 3 talant atltna 7 1 abacaba aaaabbc 12 6 abracadabraa avadakedavra 5 3 accio cicao 5 4 lumos molus 4 3 uwjt twju 4 3 kvpx vxpk

output
YES YES NO YES NO YES NO

The first case is explained in the condition.

In the second case, we can swap adjacent letters, so we can sort the string using bubble sorting, for example.

In the third case, we can show that from the string s we cannot get the string t by swapping letters at a distance of 6 or 7.

In the fourth case, for example, the following sequence of transformations is appropriate:

- "accio" → "aocic" → "cocia" → "iocca" → "aocci" → "aicco" → "cicao"

In the fifth case, we can show that it is impossible to get the string s from the string t.

In the sixth example, it is enough to swap the two outermost letters.

F. Dasha and Nightmares

4 seconds, 512 megabytes

Dasha, an excellent student, is studying at the best mathematical lyceum in the country. Recently, a mysterious stranger brought n words consisting of small latin letters $s_1, s_2, \ldots, s_n$ to the lyceum. Since that day, Dasha has been tormented by *nightmares*.

Consider some pair of integers $\langle i, j \rangle$ ($1 \le i \le j \le n$). *A nightmare* is a string for which it is true:

- It is obtained by concatenation $s_i s_j$;
- Its length is **odd**;
- The number of different letters in it is **exactly** 25;
- The number of occurrences of each letter that is in the word is **odd**.

For example, if s_i = "abcdefg" and s_j = "ijklmnopqrstuvwxyz", the pair $\langle i, j \rangle$ creates a *nightmare*.

Dasha will stop having *nightmares* if she counts their number. There are too many *nightmares*, so Dasha needs your help. Count the number of different *nightmares*.

Nightmares are called different if the corresponding pairs $\langle i, j \rangle$ are different. The pairs $\langle i_1, j_1 \rangle$ and $\langle i_2, j_2 \rangle$ are called different if $i_1 \ne i_2$ **or** $j_1 \ne j_2$.

Input

The first line contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the number of words.

The following n lines contain the words $s_1, s_2, \ldots, s_n$, consisting of small latin letters.

It is guaranteed that the total length of words does not exceed $5 \cdot 10^6$.

Output

Print a single integer — the number of different *nightmares*.

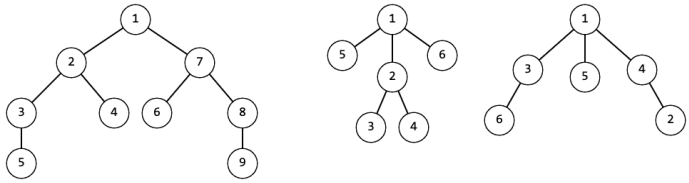
input
10 ftl abcdefghijklmnopqrstuvwxy abcdeffghijkllmnopqrsttuvwxy ffftl aabbccddeeffgghhiijjkkllmmnnooppqqrrssttuuvvwwxxyy thedevid bcdefghhiiiijklmnopqrsuwxyz gorillasilverback abcdefg ijklmnopqrstuvwxyz
output
5

In the first test, *nightmares* are created by pairs $\langle 1, 3 \rangle$, $\langle 2, 5 \rangle$, $\langle 3, 4 \rangle$, $\langle 6, 7 \rangle$, $\langle 9, 10 \rangle$.

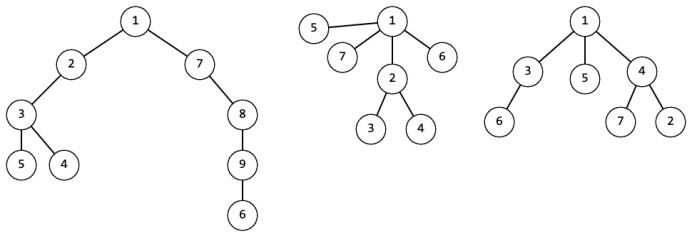
G. Symmetree

2 seconds, 256 megabytes

Kid was gifted a tree of n vertices with the root in the vertex 1. Since he really like *symmetrical* objects, Kid wants to find out if this tree is *symmetrical*.



For example, the trees in the picture above are *symmetrical*.



And the trees in this picture are not *symmetrical*.

Formally, a tree is *symmetrical* if there exists an order of children such that:

- The subtree of the leftmost child of the root is a mirror image of the subtree of the rightmost child;
- the subtree of the second-left child of the root is a mirror image of the subtree of the second-right child of the root;
- ...
- if the number of children of the root is odd, then the subtree of the middle child should be *symmetrical*.

Input

The first line of input data contains single integer t ($1 \le t \le 10^4$) — the number of test cases in the test.

The first line of each case contains an integer n ($1 \le n \le 2 \cdot 10^5$) — the number of vertices in the tree.

The following $n - 1$ lines contain two integers each u and v ($1 \le u, v \le n$, $u \ne v$) — indices of vertices connected by an edge.

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

Output

Output t strings, each of which is the answer to the corresponding test case. As an answer, output "YES" if this tree is *symmetrical*, 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).

input

```
6
6
1 5
1 6
1 2
2 3
2 4
7
1 5
1 3
3 6
1 4
4 7
4 2
9
1 2
2 4
2 3
3 5
1 7
7 6
7 8
8 9
10
2 9
9 10
2 3
6 7
4 3
1 2
3 8
2 5
6 5
10
3 2
8 10
9 7
4 2
8 2
2 1
4 5
6 5
5 7
1
```

output

```
YES
NO
YES
NO
NO
YES
```