String Workshop

Input file:	*standard input*
Output file:	*standard output*
Time limit:	11 seconds
Memory limit:	1024 mebibytes

Cleopatra loves strings. When she is given a string s of length n (consisting of characters $s[1], \ldots, s[n]$), she immediately starts sorting it. During the sorting process, she performs a sequence of swaps of the form $s[i] \leftrightarrow s[i+1]$, where $i \in \{1, 2, \ldots, n-1\}$. In the end, she returns the sorted string and demands a number of gold coins equal to the sum of i for all the swaps made (if a swap was made multiple times with the same i, then that i will be counted in the sum as many times as it was swapped). Cleopatra is not wasteful: among all the ways to sort the string, she chooses the one where the number of coins she receives for sorting is minimal.

Catherine de' Medici also loves strings. She has a string of length n, and she wants to play with it. Specifically, she is interested in the following operations:

1. "`1` i c" — replace $s[i]$ with c;
2. "`2` i j" — take a copy of the substring s from the i-th to the j-th character inclusive ($i \le j$) and sort it, while the characters of the string s remain unchanged.
3. "`3` i j" — extract the substring s from the i-th to the j-th character inclusive ($i \le j$), sort it, and insert it back into the string s at the same position.

Catherine de' Medici can handle the first type of queries herself, but the second and third types of queries are too complex, and she will go to Cleopatra for them. When assessing the cost of this procedure, Cleopatra will sum not the indices from the string s (from i to j), but the indices relative to the beginning of the substring (from 1 to $j - i + 1$).

Catherine de' Medici is also not wasteful and carefully monitors her budget. Therefore, for each of her requests to Cleopatra, calculate how many gold coins Catherine de' Medici will have to spend.

Input

The first line of input contains an integer t — the number of test cases ($1 \le t \le 3 \cdot 10^5$). For each test case:

The first line contains two integers, n and q — the length of the string and the number of queries ($1 \le n, q \le 3 \cdot 10^5$). The next line contains a string of n uppercase English letters. The following q lines describe the queries. Each query description has one of the three types:

1. "`1` i c" — replace $s[i]$ with c ($1 \le i \le n$; $c \in \{\texttt{A}, \texttt{B}, \ldots, \texttt{Z}\}$);
2. "`2` i j" — find the cost of sorting the substring $s[i..j]$, while the string s remains unchanged ($1 \le i, j \le n$);
3. "`3` i j" — find the cost of sorting the substring $s[i..j]$, while the substring $s[i..j]$ in the string s is replaced with the sorted one ($1 \le i, j \le n$).

In each test case, there will be at least one query of the second or third type.

The sum of n over all test cases is at most $3 \cdot 10^5$. The sum of q over all test cases is at most $3 \cdot 10^5$.

Output

For each test case, output in a separate line the answers to all queries of the second and third type, that is, the number of coins that Cleopatra will receive for each of the sorts.

Examples

standard input	*standard output*
1 5 20 DBCAA 2 1 5 2 2 4 1 3 Z 2 1 3 3 1 5 2 1 5 1 5 A 3 2 4 2 1 5 1 1 C 2 1 2 3 4 5 2 3 5 1 2 B 2 1 5 3 1 1 2 1 1 1 4 D 2 1 5 3 1 5	18 3 1 16 0 0 7 1 1 1 9 0 0 3 3
3 3 5 CBA 2 1 3 3 1 3 2 1 3 1 2 C 2 1 2 5 5 ABCDE 2 2 5 1 5 A 2 1 5 3 1 5 2 1 5 6 5 YYYYYY 2 1 6 1 3 A 2 1 6 3 2 5 2 1 6	4 4 0 0 0 9 9 0 0 3 1 1

Note

Consider the first example. In it, the initial string is "DBCAA". Let's go through all 20 queries in order:

1. Query "2 1 5". Current string: "DBCAA". Substring $t[1..5]$ = "DBCAA", answer: 18. The string does not change.
2. Query "2 2 4". String: "DBCAA". Substring $t[2..4]$ = "BCA", answer: 3. The string does not change.
3. Query "1 3 Z". Update: $t[3] \leftarrow$ "Z". Was "DBCAA", now "DBZAA".
4. Query "2 1 3". String: "DBZAA". Substring $t[1..3]$ = "DBZ", answer: 1. The string does not change.
5. Query "3 1 5". String: "DBZAA". Substring $t[1..5]$ = "DBZAA", answer: 16. After mutation, sorting the substring: "DBZAA" $\rightarrow$ "AABDZ". The string became "AABDZ".
6. Query "2 1 5". String: "AABDZ". Substring "AABDZ", answer: 0. The string does not change.
7. Query "1 5 A". Was "AABDZ", now "AABDA".
8. Query "3 2 4". String: "AABDA". Substring $t[2..4]$ = "ABD", answer: 0. After sorting "ABD" does not change, the string remains "AABDA".
9. Query "2 1 5". String: "AABDA". Substring "AABDA", answer: 7. The string does not change.
10. Query "1 1 C". Was "AABDA", now "CABDA".
11. Query "2 1 2". String: "CABDA". Substring $t[1..2]$ = "CA", answer: 1. The string does not change.
12. Query "3 4 5". String: "CABDA". Substring $t[4..5]$ = "DA", answer: 1. After mutation: "DA" $\rightarrow$ "AD". The string became "CABAD".
13. Query "2 3 5". String: "CABAD". Substring $t[3..5]$ = "BAD", answer: 1. The string does not change.
14. Query "1 2 B". Was "CABAD", now "CBBAD".
15. Query "2 1 5". String: "CBBAD". Substring "CBBAD", answer: 9. The string does not change.
16. Query "3 1 1". String: "CBBAD". Substring $t[1..1]$ = "C", answer: 0. Sorting a length of 1 does not change anything, the string remains "CBBAD".
17. Query "2 1 1". String: "CBBAD". Substring $t[1..1]$ = "C", answer: 0. The string does not change.
18. Query "1 4 D". Was "CBBAD", now "CBBDD".
19. Query "2 1 5". String: "CBBDD". Substring $t[1..5]$ = "CBBDD", answer: 3. The string does not change.
20. Query "3 1 5". String: "CBBDD". Substring $t[1..5]$ = "CBBDD", answer: 3. After mutation: "CBBDD" $\rightarrow$ "BBCDD". The final string is "BBCDD".