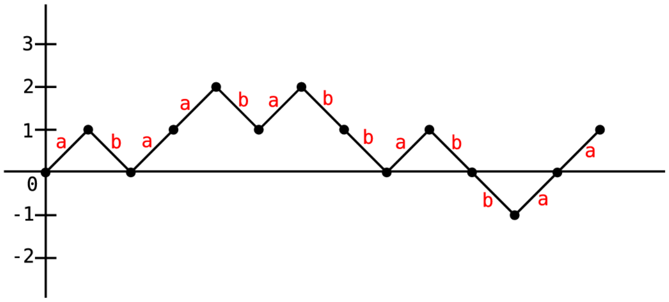
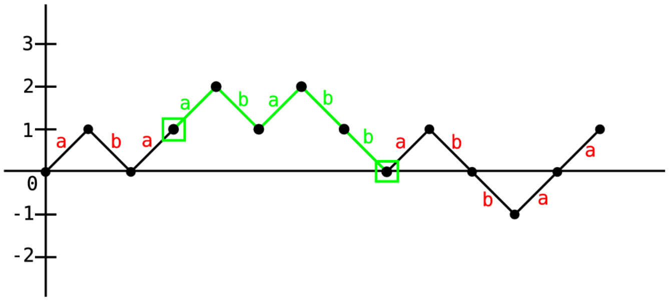
Solution for Linear Garden

For simplicity, we will consider balanced string consisting of lowercase *a*s and *b*s in this solution. We will start by analyzing what makes a string balanced.



The graph above shows a graphical representation of the string *abaababbabbaa*.
Formally, let A(S) and B(S) be the counts of letters *a* and *b* in the string S. Let Diff(S) = A(S) – B(S). The graph shows the values Diff(T) for each prefix T of the string S = *abaababbabbaa*.

Using the values shown in the graph, we can easily compute the difference between the number of *a*s and *b*s in any substring of S. More precisely, let S be the concatenation of strings T, U, and V. Then Diff(U) = Diff(TU) – Diff(T). In other words, we just need to look at the graph and read the values at the beginning and at the end of the substring in question.



In the graph above, the highlighted substring U = *ababb* starts at "height" 1, ends at "height" 0, and thus Diff(U) = 0–1 = -1.

Using this graphical representation, we can give a new definition of balanced strings. A balanced string is a string such that its graph lies entirely inside a horizontal strip of height 2.

We will now prove this. Suppose that for the string S the graph contains two vertices whose heights differ by x>=3. Then these two vertices determine a substring U where |Diff(U)|=x, and thus S is not balanced. On the other hand, if for any two vertices of the graph the height difference is at most 2, then for any substring U we have |Diff(U)|<=2, meaning that S is balanced.

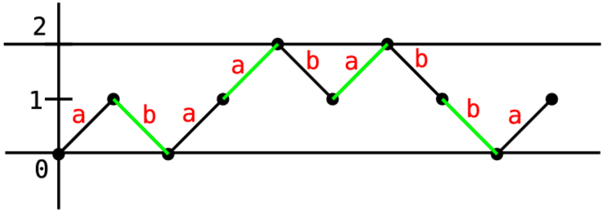
Using this knowledge, we can now try to solve the given task. To find the number assigned to the given string, we have to find an efficient way to count all balanced strings that are less or equal to the given string.

One possible way is to use dynamic programming. When generating a balanced string left to right, all we need to know at any given moment are three numbers: the lowest value A and the highest value B in the graph so far, and the current value C. All of these numbers come from the range -2 to 2. Thus we can define subproblems as follows: Let Count(K,A,B,C) be the number of ways in which we can fill in the last K letters if we are in a state described by A, B, and C. In this way we get 5^3*N subproblems (in fact, less, not all triples A,B,C are valid), and each of them can be solved in O(1).

Using the values Count(K,A,B,C) we can count all balanced strings less than the input string in linear time. For example, if the input string is S=*abaabab*, we have to count all the strings of the form *aa?????*, *abaaa??*, and *abaabaa*.

There is an even simpler solution, based on the fact that using one more observation we can count those sets of balanced strings directly.

First, consider a slightly easier problem: What is the count of all balanced strings of length N?
We will first count all balanced strings whose graphs lie in the strip between 0 and 2. Take a look at the following graph:



It is easy to realize that odd steps are always uniquely determined, and in even steps we always get to make a choice whether to add an *a* or a *b*. Thus the total count of such balanced strings is $2^{\text{floor}(N/2)}$. For strip -2 to 0 the count is the same from symmetry. For the strip -1 to 1, the odd steps are free, thus the count is $2^{\text{ceil}(N/2)}$. Of course, we counted two strings twice: the string *ababa...* and the string *babab...* Therefore the answer is $2^{\text{floor}(N/2)}+2^{\text{floor}(N/2)}+2^{\text{ceil}(N/2)}-2$.

Using a similar reasoning we can compute the number of ways how to finish any string to keep it balanced. Let the string so far be U, and let there be K letters missing. There are three possible cases:

- If the graph of U takes more than 2 rows, the answer is 0.
- If it takes 2 rows, the answer is either $2^{\text{ceil}(K/2)}$ or $2^{\text{floor}(K/2)}$, depending on whether we are currently in the middle of the strip or not. For example, there are 2^3=8 balanced strings of the form *aab?????*.
- If it takes 1 row, we have to count two types of strings. For example, if we are counting balanced strings of the form *aba?????*, we have to count all such strings in the strip 0 to 2, and all such strings in the strip -1 to 1. And obviously, in doing so we will count the string *ababa...* twice.
  Thus the answer in this case is always $2^{\text{floor}(K/2)}+2^{\text{ceil}(K/2)}-1$. For example, there are 4+8-1=11 balanced strings of the form *aba?????*.

In this way we get a very simple solution with time complexity O(N).