Find the nth character

Find the N-th character

Medium Accuracy: 10.13% Submissions: 14K+ Points: 4

Internship Alert!
New month-> Fresh Chance to top the leaderboard and get SDE Internship!

Given a binary string **s**. Perform **r** iterations on string s, where in each iteration **0** becomes **01** and **1** becomes **10**. Find the **nth** character (considering **0 based** indexing) of the string after performing these r iterations (see examples for better understanding).

Example 1:

Input:
s = "1100"
r = 2
n = 3
Output:
1
Explanation:
After 1st iteration **s** becomes "10100101".
After 2nd iteration **s** becomes "1001100101100110".
Now, we can clearly see that the character at 3rd index is 1, and so the output.

Example 2:

Input:
s = "1010"
r = 1
n = 2
Output:
0
Explanation :
After 1st iteration **s** becomes "10011001".
Now, we can clearly see that the character at 2nd index is 0, and so the output.

→ Brute force :-

s = "1010" r = 1 n = 2

answer = "1010"
while(r--) {

process = "";

answer = "1010" process = "10"

answer = "1010" process = "1001"

answer = "1010" process = "100110"

answer = "1010" process = "10011001"

answer = process;

}

return answer[n];

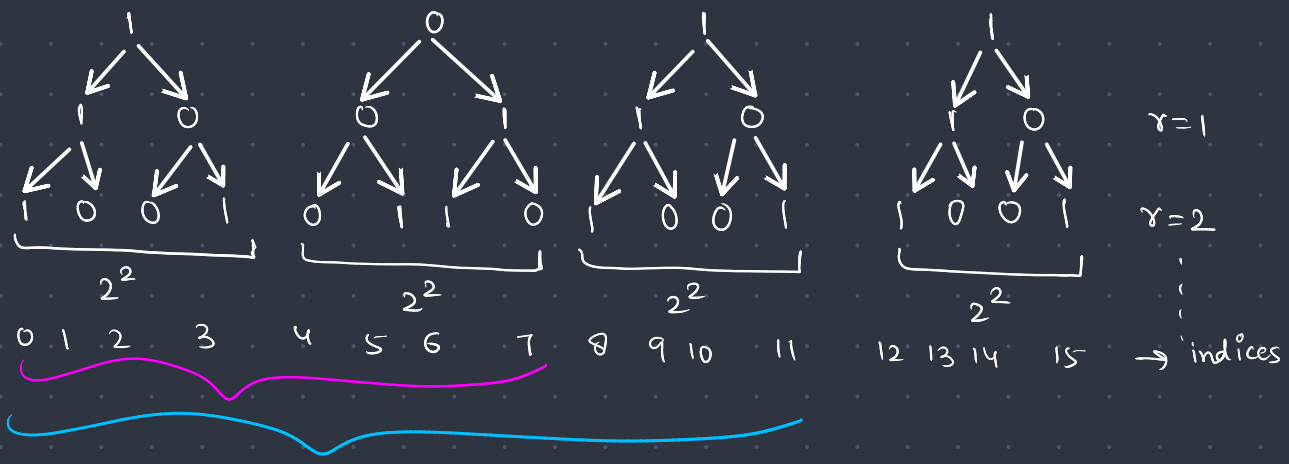
```
class Solution{
  public:
    char nthCharacter(string s, int r, int n) {
        string answer = s;

        while(r--) {
            string process = "";
            for(int i = 0; i < answer.size(); i++) {
                if(answer[i] == '1') process += "10";
                else process += "01";
            }
            answer = process;
        }
        return answer[n];
    }
};
```

Time Complexity : $O(r * 2^r)$

Space Complexity : $O(2^r)$

→ Optimized Approach :-

↳ Why to traverse the whole array.



if n = 7 ⇒ We only need to run the loop to 1st index of the original array.

if n = 10 ⇒ We only need to run the loop to 2nd index of the original array.

```
class Solution{
  public:
    char nthCharacter(string s, int r, int n) {
        string answer = s;
        int indexContainer = (1<<r) - 1;
        int index = 0;
        while(n > indexContainer) {
            index++;
            indexContainer += (1<<r);
        }
        while(r--) {
            string process = "";
            for(int i = 0; i <= index; i++) {
                if(answer[i] == '1') process += "10";
                else process += "01";
            }
            answer = process;
            index = answer.size() - 1;
        }
        return answer[n];
    }
};
```

Time Complexity : $O(r * 2^r)$

Space Complexity : $O(2^r)$