CRACK A HACK

BUILD A STRING

Course: ALGORITHMIC PROBLEM SOLVING

Course code: 17ECSE309

-
ABHAY KAGALKAR

USN: 01FE15BEC001

Problem Statement:

Greg wants to build a string S, of length N . Starting with an empty string, he can perform 2 operations:

1. Add a character to the end of S for A dollars.
2. Copy any substring of S, and then add it to the end of S for B dollars.

Calculate minimum amount of money Greg needs to build S.

Solution:
The problem is to be solved dynamically and knowledge of suffix array must be used.

- *dp*[*x*] is the answer for prefix size of *x*.
- *dp*[*x-1*]<= *dp*[*x*].

Variants:

```
1) we have got prefix(x) from prefix(x-1). that means dp[x]=dp[x-1]+A and  A>0
    so dp[x-1]<dp[x]

2) we have got prefix(x) NOT from prefix(x-1). that means dp[x]=dp[x-k]+B and  B>0 and k>0.
    so:
        if k=1 then:  dp[x]=dp[x-1]+B  so dp[x-1]<dp[x]
        if k>1 then:  dp[x]=dp[x-k]+B and dp[x-1]=min(dp[x-k]+B, other variants)
                      because substr[x-k+1,x-1] is in substr[x-k+1,x]
                      so dp[x-1]<=dp[x]
```

So that means if we wants to build *prefix(x)*, there is two variants.
1) build *prefix(x-1)* and add last symbolo.
2) build *prefix(x-k)* and copy substring size of k, where k is maximum.

So for each x we need to find maximum k where substr[*x-k+1,x*] is in substr[1,x-k]. because of *Lemma 1*, we can search *k* using binary search.

Now we need to check if substr[x-k+1,x] is in substr[1,x-k] or not.
we will use suffix array, in sorted suffixes we can easily find all substrings which are equal of given substring.

Python Code:

```
# Enter your code here. Read input from STDIN. Print output to
STDOUT
def sol():
    N, A, B = map(int, raw_input().strip().split())
    S = raw_input().strip()
    res = [0]*N
    res[0] = A
    maxl = 0
    for i in range(1,N):
        minv = res[i-1] + A
        cp, idx, newl = False, i, 0
        for k in range(maxl,-1,-1):
            if S[i-k:i+1] in S[0:i-k]:
                cp, idx, newl = True, i-k, k+1
                break
        if cp: minv = min(minv, res[idx-1]+B)
        maxl = newl
        res[i] = minv
    print res[-1]
T = int(raw_input().strip())
for x in range(T):
    sol()
```

References:

1) https://www.geeksforgeeks.org/suffix-array-set-1-introduction/

2) https://www.researchgate.net/publication/47841538_Fast_Lightweight_Suffix_Array_Construction_and_Checking

3) https://en.wikipedia.org/wiki/Maximum_subarray_problem

4) https://discuss.codechef.com/questions/21385/a-tutorial-on-suffix-arrays