Nama : Timothy Hosia Budianto

NRP : 5025211098

Resume pembahasan soal remedy kuiz 1 SPOJ MOVES

INPUT :

Soal ini pertama meminta input 2 nilai interger. Interger pertama bersimbol n akan menentukan jumlah panjang character yang ada pada string. Interger kedua bersimbol m menunjukan berapa queries yang akan dilakukan untuk memodifikasi urutan character string tersebut.

PENJELASAN :

Ini berarti soal ini berarti merupakan soal single testcase dengan multiple queries. Setelah memasukan 2 nilai integer tersebut, kemudian kita memasukan sebuah string dengan panjang n, lalu memasukan sebanyak m queries berisi 2 nilai interger lagi yaitu left dan right. 2 Nilai interger tersebut akan menentukan character mana yang akan dipindahkan ke awal string dengan ketentuan dari index left sampai index right dengan ketentuan index pertama dimulai dari nilai 1.

OUTPUT :

Program akan mengeluarkan string yang telah dimodifikasi, yaitu hasil dari string yang berulang kali dipindahkan

PENJELASAN :

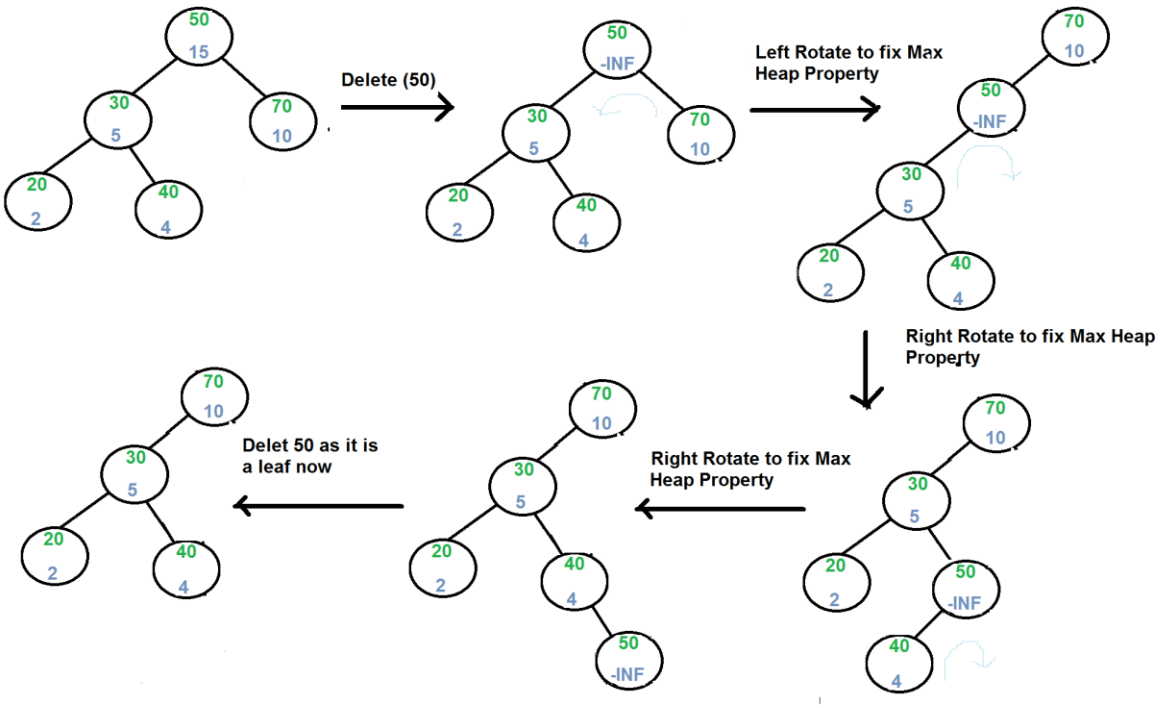
```
Input 1
10 3
abcdefghij
2 3
3 5
3 7

Output 1
ebcfgadhij

Explanation 1
Here is the string after each query.
abcdefghij
bcadefghij (moved 'bc' to the start since it's the substring from index 2 to 3)
adebcfghij (moved 'ade' to the start)
ebcfgadhij (moved 'ebcfg' to the start)
```

sumber dari spoj

Jika diselesaikan secara linear akan menghasilkan time complexity paling cepat yaitu m x n. Sehingga soal ini didapat bahwa soal ini tidak bisa diselesaikan secara liniear karna akan memerlukan waktku yang terlalu lama atau time limit exceeded. Oleh karena itu pada soal itu diperlukan pendekatan logaritmik.Dengan metode logaritmik dapat didapat waktu hingga m log n. Time complexity tersebut dapat dicapai dengan penggunaan struktur data yang efisien untuk dapat menyelesaikan persoalan dynamic query seperti ini. Struktur data yang tepat untuk soal ini adalah struktur data treap dan metode randomized algorithm khususnya di randomized binary search tree. Perlu diingat bahwa random disini merupakan pada range tertentu sesuai jumlah data untuk mendapatkan solusinya. Kedua metode tersebut dapat membuat setiap pencarian insertsi dan delesi memerlukan time complexity logaritmik.

Dikutip dari Wikipedia Treap pertama kali dijelaskan oleh Raimund Seidel dan Cecilia R. Aragon pada tahun 1989;[1][2] namanya adalah gabungan dari metode tree dan heap. Treap adalah suatu balanced binary tree yang menggunakan randomization untuk menjaga keseimbangan dari tree tersebut.

Treap menggunakan cara randomized untuk mendapatkan hasil yang sama tapi dengan waktu yg berada di rentang tertentu. Kelebihan dari treap ini adalah memiliki efisiensi waktu karna menggunakan konsep binary tree. Selain itu di soal ini pengerjaan jadi lebih cepat karna tidak menggunakan for loop tapi menggunakan rekursi yaitu memanggil fungsinya sendiri.

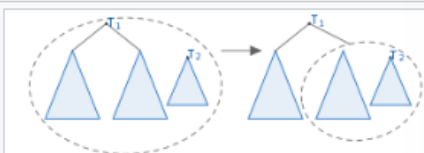


Sumber dari https://www.geeksforgeeks.org/treap-a-randomized-binary-search-tree/

Dalam treap memiliki berbagai macam operasi.Operasi pertama adalah join yaitu menggabungkan 2 subtree menjadi subtree menjadi 1 tree. Dengan pseudocode sebagai berikut:

The join algorithm is as follows:

```
function join(L, k, R)
    if prior(k, k(L)) and prior(k, k(R)) return Node(L, k,
R)
    if prior(k(L), k(R)) return Node(left(L), k(L),
join(right(L), k, R))
    return Node(join(L, k, left(R)), k(R), right(R))
```

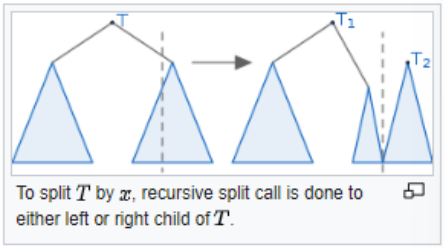


Join performed on treaps T_1 and T_2. Right child of T_1 after the join is defined as a join of its former right child and T_2.

Yang kedua adalah metode split. Split adlaah membagi tree menjadi 2 subtree yaitu left and right dengan menggunakan pendekatan rekursi.

The split algorithm is as follows:

```
function split(T, k)
    if (T = nil) return (nil, false, nil)
    (L, (m, c), R) = expose(T)
    if (k = m) return (L, true, R)
    if (k < m)
        (L', b, R') = split(L, k)
        return (L', b, join(R', m, R))
    if (k > m)
        (L', b, R') = split(R, k)
        return (join(L, m, L'), b, R'))
```



To split T by x, recursive split call is done to either left or right child of T.

The union of two treaps t_1 and t_2, representing sets A and B is a treap t that represents $A \cup B$. The following recursive algorithm computes the union:

```
function union(t1, t2):
    if t1 = nil:
        return t2
    if t2 = nil:
        return t1
    if priority(t1) < priority(t2):
        swap t1 and t2
    t<, t> ← split t2 on key(t1)
    return join(union(left(t1), t<), key(t1),
                union(right(t1), t>))
```

Here, *split* is presumed to return two trees: one holding the keys less than its input key, one holding the greater keys. (The algorithm is non-destructive, but an in-place destructive version exists as well.)

Sumber dari https://www.geeksforgeeks.org/treap-a-randomized-binary-search-tree/

Dalam pengaplikasiannya sendiri di source code adalah sebagai berikut

```
void merge(int &t, int l, int r){
    if(!l||!r)t=l|r;
    else if(l>r)merge(rn[l],rn[l],r),t=l;
    else merge(ln[r],l,ln[r]),t=r;
    if(t)sz[t]=sz[ln[t]]+sz[rn[t]]+1;
}

void split(int t, int &l, int &r, int sv){
    if(!t)l=r=0;
    else if(sz[ln[t]]<sv)
        split(rn[t],rn[t],r,sv-sz[ln[t]]-1),l=t;
    else split(ln[t],l,ln[t],sv),r=t;
    if(t)sz[t]=sz[ln[t]]+sz[rn[t]]+1;
}
```

Pada fungsi diatas menggunakan pointer agar menggunakan pass by refrence dan tidak perlu passing value satu per satu agar waktunya dipercepat.

Dari hasil submission menggunakan metode randomized ini memperoleh hasil memory dan time yang bervariasi. Berikut adalah hasil dan rata – ratanya selama 10 kali percobaan.

Submision	Time	memory (m)
1	0,78	5,4
2	1,02	5,4
3	0,93	5,4
4	0,8	5,4
5	0,74	5,4
6	0,88	5,4
7	0,8	5,4
8	0,81	5,4
9	0,76	5,4
10	0,79	5,5
Rata-rata	**0,831**	**5,41**

Terlihat dari table bahwa waktu yang digunakan berentang dari 0,74 sampai 1,02 dengan rata-rata 0,831. Selain itu memory yang digunakan pada umumnya 5,4 dan 5,5 di percobaan terakhir yang menyebabkan rata-ratanya menjadi 5,41