NAMA : Fajar Satria
NRP : 05111940000083
Kelas : PAA B

MOZSATLA - Sharmeen and the Lost Array

Sharmeen loves array very much. Many days ago she wrote an array (of n elements and the index of this array is 1 based) in her notebook, but unfortunately she lost the notebook. She want to restore the array. The only clue she know that is if in any position(i) of the array (1<= i < n), the element in this position is greater than, equal or less than the next position(i+1) element. Can you help her to restore the array?.

SOLUSI

Inti dari persoalan ini adalah untuk mencari deret bilangan sepanjang **n** yang dibentuk dari **n-1** buah pernyataan. Pernyataan tersebut dapat berupa 0 untuk equal, 1 untuk less, dan 2 untuk greater. Cetak deret bilangan **lexicographically smallest one** dan tiap anggota deret bilangan **harus lebih besar dari 0.**

Setelah melakukan observasi, maka beberapa hal yang diperlukan adalah menyimpan pernyataan kedalam sebuah array **status[n-1]** dan menyimpan deret bilangan solusi kedalam **arr[n]**. Solusi tersebut didapatkan dengan mengiterasi **status** secara topdown dari **i=n-1** hingga **i=1** dengan terminalcase **arr[n]=1**

Sesuai dengan pernyataan **status**, maka pernyataan dapat dibagi menjadi 3 bagian

a. **status[i]=0** maka **arr[i]=arr[i+1]**
b. **status[i]=2** maka **arr[i]=arr[i+1]+1**
c. **status[i]=1** merupakan case spesial karena berkaitan erat dengan pembentukan **lexicographically smallest one** dan **greater than zero**. Untuk **greater than zero** maka **arr[i]=1**. Sedangkan untuk **lexicographically smallest one** maka perlu diiterasi ulang secara forward dari **j=i+1** hingga **j=n-1**. Dalam iterasi tersebut dilakukan pengecekan apakah memenuhi **status[j]=0** atau **status[j]=1** untuk **arr[j+1]<=arr[j]**. Jika memenuhi kondisi tersebut maka **arr[j+1]=arr[j+1]+1**. Sedangkan untuk **status[j]=2** tidak diperlukan penambahan **arr[j+1]** karena nanti hasilnya pasti tidak akan memenuhi **lexicographically smallest**.

Sehingga untuk input **{1, 2, 0, 1}** maka deret yang dihasilkan adalah **{1, 2, 1, 1, 2}** seperti tabel berikut

index ke	1	2	3	4	5	
status	1	2	0	1		
arr					1	terminalcase → arr[n]=1
arr				1	2	status[4]=1 → arr[4]=1; arr[4+1]=arr[4+1]+1
arr			1	1	2	status[3]=0 → arr[3]=arr[3+1]
arr		2	1	1	2	status[2]=2 → arr[2]=arr[2+1]+1
arr	1	2	1	1	2	status[1]=1 → arr[1]=1

PSEUDOCODE

```
SOLVE(A)
1.  n = A.length
2.  let B[0..n-1] be a new array
3.  B[n-1] = 1
4.  for i=n-2 to 0
5.          if status[i] = '0'
6.                  arr[i] = arr[i+1]
7.          else if status[i] = '2'
8.                  arr[i] = arr[i+1] + 1
9.          else
10.                 arr[i]=1
11.                 if arr[i+1] = 1
```

12. arr[i+1]++
13. for j=i+1 to n-2
14. if status[j]='0' or (status[j]='1' and arr[j+1]<=arr[j])
15. arr[j+1] = arr[j+1] + 1
16. else j=n
17. return B

SOURCECODE

```
#include <cstdio>
int i,j,n,t,arr[100000];
char c,status[100001];
int main(){
    while(1){
        c = getchar_unlocked();
        if(c>='0' && c<='9') t=10*t+c-'0';
        else break;
    }
    while(t--){
        n=0;
        while(1){
            c = getchar_unlocked();
            if(c>='0' && c<='9') n=10*n+c-'0';
            else break;
        }
        i=0;
        while(1){
            c = getchar_unlocked();
            if(c>='0' && c<='9') status[i]=c;
            else if(c==' ') i++;
            else break;
        }
        arr[n-1]=1;
        for(i=n-2;i>=0;i--){
            if(status[i]=='0')
                arr[i]=arr[i+1];
            else if(status[i]=='2')
                arr[i]=arr[i+1]+1;
            else{
                arr[i]=1;
                if(arr[i+1]==1){
                    arr[i+1]++;
                    for(j=i+1;j<n-1;j++){
                        if(status[j]=='0' || (status[j]=='1' && arr[j+1]<=arr[j]))
                            arr[j+1]++;
                        else j=n;
                    }
                }
            }
        }
        for(i=0;i<n-1;i++) printf("%d ",arr[i]);
        printf("%d\n",arr[n-1]);
    }
    return 0;
}
```

SUBMISSIONS

28031534	☐	2021-06-09 12:58:14	Sharmeen and the Lost Array	**accepted** edit ideone it	0.03	5.3M	CPP

Restaurants

In a certain city there is a restaurant network that has a total of **n** almost identical halls. The administration received **k** applications for holding corporate events on a pre-holiday day. Each application specifies the start and end time (from **00:00** to **23:59**). To carry out one event, one hall is needed entirety (which specifically does not matter). After the end of the event, you need at least half an hour to prepare for the next event in this room. It is required to satisfy as many applications as possible. If you can satisfy all, then the smallest number of halls should be used.

SOLUSI

Permasalahan ini hampir mirip dengan Activity Selection Problem pada umumnya. Activity Selection pada umunya hanya untuk 1 jadwal namun, pada permasalahan ini dapat dipecah hingga **n** jadwal. Intinya terdapat **k** applications (Activity) dan **n** halls. Dan permasalahannya adalah membuat jadwal untuk menampung application sebanyak mungkin dan penggunaan hall seminimal mungkin.

Pada soal ini untuk perhitungan penggunaan hall dapat disimpan pada **array** sepanjang n. Dan untuk application (Activity) dapat disimpan kedalam sebuah struktur data **struct**.

Seperti pada permasalahan Activity Selection pada umumnya, data activity yang terkumpul pada **struct** harus disort atau diurutkan berdasarkan suatu kondisi. Pada soal ini kondisi yang memungkinkan yaitu

1. Urut berdasarkan waktu mulai application tercepat
2. Jika waktu mulai application sama maka diurutkan berdasar waktu selesai tercepat

Setelah semua jadwal terurut, hal selanjutnya yang harus dilakukan adalah melakukan penyortiran untuk memenuhi 1 hall. Yaitu mengambil jadwal yang tidak saling bertabrakan, dengan ketentuan jadwal mulai suatu application harus lebih dari atau sama dengan jadwal berakhir suatu application + 30 menit istirahat.