Team notebook

Javalieron

July 23, 2018

Contents

1 DS

1.1 lazySegtree

```
#include <bits/stdc++.h>
using namespace std;
#define INF 1e9
const int MAXN = 1e5+5;
int a[MAXN],t[4*MAXN],lazy[4*MAXN];
//funcion para construir el segtree
void build(int v,int tl,int tr){
	memset(lazy,0,sizeof(lazy));
	if(tl==tr)
		t[v]=a[tl];
	else{
		int tm = (tl+tr)/2;
		build(v*2,tl,tm);
		build(v*2+1,tm+1,tr);
		t[v]=max(t[v*2],t[v*2+1]);
	}
}
//funcion para propagar el valor a los hijos del nodo.
void push(int v){
	t[v*2]=lazy[v];
	lazy[v*2]=lazy[v];
	t[v*2+1]=lazy[v];
	lazy[v*2+1]=lazy[v];
	lazy[v]=0;
}
//funcion para hacer update de un rango con un valor dado.
void update(int v,int tl, int tr, int l, int r,int val){
	if(l>r)
		return;
	if(l==tl&&tr==r){
		t[v]=val;
		lazy[v]=val;
	}
	else{
		//solo es necesario propagar el valor si existe un valor
		    lazy guardado.
		if(lazy[v]!=0)
			push(v);
		int tm=(tl+tr)/2;
		update(v*2,tl,tm,l,min(tm,r),val);
		update(v*2+1,tm+1,tr,max(l,tm+1),r,val);
		t[v]=max(t[v*2],t[v*2+1]);
	}
}
//funcion para realizar range query.
int query(int v, int tl, int tr, int l, int r){
	if(l>r)
```

```
                return -INF;
        if(tl==l&&tr==r){
                return t[v];

        }
        if(lazy[v]!=0)
                push(v);
        int tm = (tl+tr)/2;
        return
            max(query(v*2,tl,tm,l,min(r,tm)),query(v*2+1,tm+1,tr,max(tm+1,l),r));
}
int n,m,i,j;
int main(){
        while(scanf("%d%d",&n,&m)==2&&n+m){
                for(int i = 0; i<n; i++){
                        a[i]=i;
                }
                build(1,0,n-1);
                printf("%d\n",query(1,0,n-1,0,n-1));
                update(1,0,n-1,n/2,n-1,0);
                printf("%d\n",query(1,0,n-1,0,n-1));
        }
        return 0;
}
```

1.2 simpleSegtree

```
#include <bits/stdc++.h>
using namespace std;
const int MAXN = 1e5+5;
int a[MAXN],t[4*MAXN];
void build(int v, int tl, int tr){
        if(tl==tr)
                t[v]=a[tl];
        else{
                int tm = (tl+tr)/2;
                build(v*2,tl,tm);
                build(v*2+1,tm+1,tr);
                //depende la operacin a realizar.
                t[v]=t[v*2]+t[v*2+1];
        }
}
int get(int v, int tl, int tr, int l, int r){
        if(l>r)
                //retornar valor neutro de la opracin.
                return 0;
        if(tl==l&&tr==r)
                return t[v];
        int tm = (tl+tr)/2;
        return get(v*2,tl,tm,l,min(r,tm))+get(v*2,tm+1,tl,max(tm+1,l),r);

}
void update(int v,int tl, int tr, int pos, int new_val){
        if(tl==tr)
                t[v]=new_val;
        else{
                int tm = (tl+tr)/2;
                if(pos<=tm)
                        update(v*2,tl,tm,pos,new_val);
                else
                        update(v*2+1,tm+1,tr,pos,new_val);
                t[v]=t[v*2]+t[v*2+1];
        }
}
int main(){
        int n=100,val=0,pos=5,i=2,j=3;
        //leer arreglo
        //construir el segtree
        build(1,0,n-1);
        int res = get(1,0,n-1,i,j);
        update(1,0,n-1,pos,val);
        res = get(1,0,n-1,0,10);
        printf("%d\n",res);


}
```

2 Grafos

2.1 kosaraju

```
#include <bits/stdc++.h>
using namespace std;
//choose MAXN according to the problem.
const int MAXN = 100005;
```

```
vector<int> g[MAXN],gr[MAXN];
bool vis[MAXN];
stack<int> tp;
int n,m;
int scc = 0;
void dfs(int x){
	vis[x]=1;
	for(vector<int>::iterator it = g[x].begin(); it!=g[x].end(); ++it){
		int y = *it;
		if(!vis[y])
			dfs(y);
	}
	tp.push(x);
}
void dfs2(int x){
	vis[x]=1;
	for(vector<int>::iterator it = gr[x].begin(); it!=gr[x].end();
	    ++it){
		int y = *it;
		if(!vis[y])
			dfs2(y);
	}
}
int main(){
	//read graph.
	//kosaraju
	memset(vis,0,sizeof(vis));
	for(int i = 0; i<n; i++)
		if(!vis[i])
			dfs(i);
	memset(vis,0,sizeof(vis));
	while(!tp.empty()){
		int x = tp.top();
		tp.pop();
		if(!vis[x]){
			scc++;
			dfs2(x);
			//do extra things like graph condensation.
		}
	}
	return 0;
}
```

3 Strings

3.1 pi-function

```
#include <bits/stdc++.h>
using namespace std;
//funcion para calcular el prefix function de un string
//prefix("abcabcabc")={0,0,0,1,2,3,1,2,3}
vector<int> pi_function(string s){
	int n = (int)s.length();
	vector<int> pi(n);
	for(int i =1; i<n; i++){
		int j = pi[i-1];
		while(j>0&&s[i]!=s[j])
			j=pi[j-1];
		if(s[i]==s[j])
			j++;
		pi[i]=j;
	}
	return pi;
}
int main(){
	string p;
	string t;
	cin>>t>>p;
	int len = (int)p.length();
	// hacer kmp es igual a halla la prefix function de p+"#"+s y
	    calcular y un match es equivalente a pi[i]=len(p)
	string kmp = p+"#"+t;
	vector<int> pi = pi_function(kmp);
	for(int i = 0; i<(int)pi.size();i++)
		printf("%d ",pi[i]);
	int ans = 0;
	for(int i = len+1; i<(int)pi.size(); i++)
		if(pi[i]==len)
			ans++;
	printf("%d\n",ans);
	return 0;
}
```

3.2 z-function

```cpp
#include <bits/stdc++.h>
using namespace std;
//funcion para calcular la z function de un string
vector<int> z_function(string s){
	int n = (int) s.length();
	vector<int> z(n);
	for(int i=1,l=0,r=0; i<n; i++){
		if(i<=r)
			z[i]=min(r-i+1,z[i-l]);
		while(i+z[i]<n && s[z[i]]==s[i+z[i]])
			++z[i];
		if(i+z[i]-1>r)
			l=i,r=i+z[i]-1;
	}
	return z;
}
int main(){
	string s("abcabcabc");
	vector<int> z = z_function(s);
	//z == {0,0,0,6,0,0,3,0,0}
	for(int i = 0; i<(int)z.size(); i++)
		printf("%d ",z[i]);
	return 0;
}
```