ICPC Templates For Grooming

ChenJr

June 5, 2019

Contents

1 字符串

1.1 Manacher

```cpp
#include <bits/stdc++.h>
#define maxn 2000005
using namespace std;
int mp[maxn];
string str;
char c[maxn];
void Manacher(string s,int len){
    int l=0,R=0,C=0;;
    c[l++]='$', c[l++]='#';
    for(int i=0;i<len;i++){
        c[l++]=s[i], c[l++]='#';
    }
    for(int i=0;i<l;i++){
        mp[i]=R>i?min(mp[2*C-i],R-i):1;
        while(i+mp[i]<l&&i-mp[i]>0){
            if(c[i+mp[i]]==c[i-mp[i]]) mp[i]++;
            else break;
        }
        if(i+mp[i]>R){
            R=i+mp[i], C=i;
        }
    }
}
int main()
{
    int cnt=0;
    while(cin>>str){
        if(str=="END") break;
        int len=str.length();
        Manacher(str,len);
        int ans=0;
        for(int i=0;i<2*len+4;i++){
            ans=max(ans,mp[i]-1);
        }
        printf("Case %d: %d\n",++cnt,ans);
    }
    return 0;
}
```

2 图论

2.1 最短路

2.1.1 Dijkstra

```cpp
const int INF=0x3f3f3f3f;
int head[maxn],cnt;
struct edge{
    int to,next;
    long long cost;
}q[maxn];
void add_edge(int from,int to,int cost){
    q[cnt].to=to;
    q[cnt].cost=cost;
    q[cnt].next=head[from];
    head[from]=cnt++;
}
typedef pair<int,int>P;
long long d[maxn];
int a[maxn],b[maxn],c[maxn];
void dijikstra(int s){
    memset(d,INF,sizeof(d));
    priority_queue<P,vector<P>,greater<P> >que;
    d[s]=0;
    que.push(P(0,1));
    while(!que.empty()){
        P p=que.top();
        que.pop();
        int x=p.second;
        //if(d[x]<p.first) continue;
        for(int i=head[x];i!=-1;i=q[i].next){
            edge id=q[i];
            if(d[id.to]>d[x]+id.cost){
                d[id.to]=d[x]+id.cost;
                que.push((P(d[id.to],id.to)));
            }
        }
    }
}
```

2.2 网络流

2.2.1 Dinic

网络流之 Dinic，此模板带有当前弧优化。

```cpp
#include <bits/stdc++.h>
using namespace std;
const int maxn=50005;
const int maxm=500005;
const int inf=0x3f3f3f3f;
struct Node{
    int to,val,next;
}q[maxm<<1];
int head[maxn],cnt=0,dep[maxn],cur[maxn],vis[maxn];
int sp,ep,maxflow;
void init(){
    memset(head,-1,sizeof(head));
    cnt=2,maxflow=0;
}
void addedge(int from,int to,int val){
    q[cnt].to=to;
    q[cnt].val=val;
    q[cnt].next=head[from];
    head[from]=cnt++;
}
void add_edge(int from,int to,int val){
    addedge(from,to,val);
    addedge(to,from,0);
}
bool bfs(int n){
    for(int i=0;i<=n;i++){
        cur[i]=head[i],dep[i]=0x3f3f3f3f;
        vis[i]=0;
    }
    dep[sp]=0;
    queue<int>que;
    que.push(sp);
    while(!que.empty()){
        int x=que.front();
        que.pop();
        vis[x]=0;
        for(int i=head[x];i!=-1;i=q[i].next){
            int to=q[i].to;
            if(dep[to]>dep[x]+1&&q[i].val){
                dep[to]=dep[x]+1;
                if(!vis[to]){
                    que.push(to);
                    vis[to]=1;
                }
            }
        }
    }
    if(dep[ep]!=inf) return true;
    else return false;
}
int dfs(int x,int flow){
    int rlow=0;
    if(x==ep){
        maxflow+=flow;
        return flow;
    }
    int used=0;
    for(int i=cur[x];i!=-1;i=q[i].next){
        cur[x]=i;
        int to=q[i].to;
        if(q[i].val&&dep[to]==dep[x]+1){
            if(rlow=dfs(to,min(flow-used,q[i].val))){
                used+=rlow;
                q[i].val-=rlow;
                q[i^1].val+=rlow;
                if(used==flow) break;
            }
        }
    }
    return used;
}
int dinic(int n){
    while(bfs(n)){
        dfs(sp,inf);
    }
    return maxflow;
}
int main()
{
    int n,m;
    scanf("%d%d%d%d",&n,&m,&sp,&ep);
    register int i;
    int u,v,val;
    init();
    for(i=1;i<=m;i++){
        scanf("%d%d%d",&u,&v,&val);
        add_edge(u,v,val);
    }
    printf("%d",dinic(n));
    return 0;
}
```

2.3 强连通分量缩点

```
#include <bits/stdc++.h>
#define maxn 200005
using namespace std;
struct edge{
    int next,to;
}q[maxn];
int head[maxn],dfn[maxn],low[maxn],cnt,tot;
int vis[maxn],belong[maxn],index,belong_num[maxn],num_index;
int indegree[maxn],outdegree[maxn];
void add_edge(int from,int to){
    q[cnt].next=head[from];
    q[cnt].to=to;
    head[from]=cnt++;
}
void init(){//初始化
    memset(vis,0,sizeof(vis));
    memset(dfn,0,sizeof(dfn));
    memset(head,-1,sizeof(head));
    memset(low,0,sizeof(low));
    memset(belong_num,0,sizeof(belong_num));//在某个连通块中有多少个结点
    memset(indegree,0,sizeof(indegree));//新图的入度
    memset(outdegree,0,sizeof(outdegree));
    index=num_index=cnt=tot=0;
}
stack<int>st;
void tarjin(int x){//Tarjin的主体
```

```
    dfn[x]=low[x]=++tot;
    vis[x]=1;
    st.push(x);
    for(int i=head[x];i!=-1;i=q[i].next){
        edge e=q[i];
        if(!dfn[e.to]){
            tarjin(e.to);
            low[x]=min(low[e.to],low[x]);
        }
        else if(vis[e.to]==1){
            low[x]=min(low[x],dfn[e.to]);
        }
    }
    if(dfn[x]==low[x]){
        int v;
        index=index+1;
        do{
            v=st.top();
            st.pop();
            belong[v]=index;
            belong_num[index]++;
            vis[v]=0;
        }while(v!=x);
    }
}
void solve(int n,int m,int root){
    for(int i=1;i<=n;i++){//对图进行Tarjin
        if(!dfn[i]){
            tarjin(i);
        }
    }
    //如果连通分量只有一个，则直接输出0
    if(index==1){
        puts("0");
        return ;
    }
    indegree[belong[root]]=1;//确保初始点root所在的连通分量入度不为0

    for(int i=1;i<=n;i++){//重构图的过程
        for(int j=head[i];j!=-1;j=q[j].next){
            edge e=q[j];
            if(belong[i]==belong[e.to]) continue;
            indegree[belong[e.to]]++;
            outdegree[belong[i]]++;
        }
    }
    int cnt=0;//统计入度为0的点
    for(int i=1;i<=index;i++){
        if(indegree[i]==0){
            cnt++;
        }
    }
    cout<<cnt<<endl;
```

```
}
```