

图论

dijkstra堆优化

```
1  #include<iostream>
2  #include<cstdio>
3  #include<queue>
4  #include<cstring>
5  #include<algorithm>
6  using namespace std;
7  int n,m,s,g[10020],d[10020],vis[10020];
8  struct Edge{
9      int u,v,w,next;
10 }e[500020];
11 struct ds{
12     int d,u;
13     bool operator<(const ds &x)const{return d>x.d;}
14 };
15 void dij(int s){
16     memset(vis,0,sizeof(vis));
17     priority_queue<ds>q;
18     for(int i=1;i<=n;i++) d[i]=500000000;
19     d[s]=0;
20     q.push((ds){0,s});
21     while(!q.empty()){
22         ds x=q.top();q.pop();
23         int u=x.u;
24         if(vis[u]) continue;
25         vis[u]=1;
26         for(int i=g[u];i>0;i=e[i].next){
27             int v=e[i].v,w=e[i].w;
28             if(d[u]+w<d[v]){
29                 d[v]=d[u]+w;
30                 q.push((ds){d[v],v});
31             }
32         }
33     }
34 }
35
36
37 int main(){
38     scanf("%d%d%d",&n,&m,&s);
39     int x,y,z;
40     for(int i=1;i<=m;i++){
41         scanf("%d%d%d",&x,&y,&z);
42         e[i]=(Edge){x,y,z,g[x]};g[x]=i;
43     }
44     dij(s);
45     for(int i=1;i<=n;i++){
46         if(d[i]!=500000000) printf("%d ",d[i]);
47     }
```

```

48         printf("2147483647 ");
49     }
50 }
51 printf("\n");
52 return 0;
53 }

```

最小生成树

```

1  #include<iostream>
2  #include<cstdio>
3  #include<algorithm>
4  using namespace std;
5  int n,m;
6  int f[5020];
7  struct Edge{
8      int u,v,w;
9  }e[200020];
10 int cmp(Edge a,Edge b){
11     return a.w<b.w;
12 }
13 int find(int x){
14     return f[x]==0?x:f[x]=find(f[x]);
15 }
16 int main(){
17     scanf("%d%d",&n,&m);
18     int x,y,z;
19     for(int i=1;i<=m;i++){
20         scanf("%d%d%d",&x,&y,&z);
21         e[i]=(Edge){x,y,z};
22     }
23     sort(e+1,e+1+m,cmp);
24     int cnt=0,sum=0;
25     for(int i=1;i<=m;i++){
26         int x=find(e[i].u),y=find(e[i].v);
27         if(x!=y){
28             f[x]=y;
29             cnt++;
30             sum+=e[i].w;
31         }
32         if(cnt==n-1){
33             printf("%d\n",sum);
34             return 0;
35         }
36     }
37     printf("orz\n");
38     return 0;
39 }

```

SPFA

```

1  #include<iostream>
2  #include<cstdio>
3  using namespace std;
4  #include<cstring>
5  #include<algorithm>
6  #include<cmath>
7  #include<queue>
8  int n,m,s,t;
9  const int MAXN=10020;
10 const int MAXM=500020;
11 struct Edge{
12     int u,v,w,next;
13 }e[MAXN];
14 int g[MAXN];
15 int inq[MAXN],d[MAXN];
16 void spfa(int s){
17     queue<int>q;
18     q.push(s);
19     inq[s]=1;
20     memset(d,0x3f,sizeof(d));
21     d[s]=0;
22     while(!q.empty()){
23         int uu=q.front();inq[uu]=0;
24         q.pop();
25         for(int i=g[uu];i>0;i=e[i].next){
26             int vv=e[i].v,ww=e[i].w;
27             if(d[uu]+ww<d[vv]){
28                 d[vv]=d[uu]+ww;
29                 if(!inq[vv]){
30                     q.push(vv);
31                     inq[vv]=1;
32                 }
33             }
34         }
35     }
36     return;
37 }
38 int main(){
39     scanf("%d%d%d",&n,&m,&s);
40     for(int i=1;i<=m;i++){
41         int xx,yy,ww;
42         scanf("%d%d%d",&xx,&yy,&ww);
43         e[i]=(Edge){xx,yy,ww,g[xx]};g[xx]=i;
44     }
45     spfa(s);
46     for(int i=1;i<=n;i++){
47         if(d[i]>100000000) printf("2147483647 ");
48         else{
49             printf("%d ",d[i]);
50         }
51     }
52     return 0;
53 }

```

LCA最近公共祖先（倍增）

```
1 //C++代码实现:
2 /*输入
3 12 5
4 1
5 1
6 1
7 2
8 2
9 5
10 5
11 6
12 6
13 10
14 10
15 3 11
16 7 12
17 4 8
18 9 12
19 8 10
20 输出
21 1
22 2
23 1
24 6
25 2
26 */
27 #include<iostream>
28 #include<stdio.h>
29 #include<memory.h>
30 using namespace std;
31 #define max_size 1010
32 int d[max_size],p[max_size][10];
33 int head[max_size];
34 int cnt;
35 struct Edge{
36     int v;
37     int pre;
38 }eg[max_size];
39 //建树的函数
40 void add(int x,int y){
41     eg[cnt].v=y;//儿子
42     eg[cnt].pre=head[x];//head记录x连接的边在临界表中的编号.....
43     head[x]=cnt++;
44 }
45 //dfs()初始整颗树，算出d[1-n],p[1-n][j];
46 void dfs(int k){
47     if (head[k]==0) return;
48     int m,x,i,j;
49     for(i=head[k];i!=0;i=eg[i].pre){
50         x=eg[i].v;//找儿砸
51         p[x][0]=k;//记录x的爸爸(祖先)
52         m=k;
```

```

53     d[x]=d[k]+1;//记录深度
54     for(j=0;p[m][j]!=0;j++){
55         p[x][j+1]=p[m][j];//利用公式p[x][j]=p[p[x][j-1]][j-1],这里的m就是p[x][j-
56         m=p[m][j];//理解 2^3=2^2*2;
57     }
58     dfs(x);
59 }
60 }
61 int find_lca(int x,int y){
62     int m,k;
63     if(x==y) return x;
64     if(d[x]<d[y]){m=x;x=y;y=m;}//使x的深度>y
65     m=d[x]-d[y];
66     k=0;
67     while(m){/*将x的深度调到和y的深度一样*///倍增法 k记录2^k次方
68         if(m&1) x=p[x][k];
69         m>>=1;//m=m>>1;
70         k++;
71     }
72     if(x==y) return x;
73     k=0;/*向上调节,找最近公共祖先,算法的核心,相当于一个二分查找。*/
74     while(x!=y){
75         if (p[x][k]!=p[y][k]||p[x][k]==p[y][k]&& k==0){
76             /*如果p[x][k]还不相等,说明节点p[x][k]还在所求点的下面,
77             所以继续向上调节;如果相等了,并且就是他们父节点,则那个节点一定就是所求点
78             */
79             x=p[x][k];
80             y=p[y][k];
81             k++;
82         }
83         else k--;
84         /*如果p[x][k]=p[y][k],可以说明p[x][k]一定是x和y的共祖先,但不一定是最近
85         所以向下找看还有没有更近的公共祖先。*/
86     }
87     return x;
88 }
89 int main(){
90     int i,n,m,x,y;
91     while(cin>>n>>m){
92         memset(head,0,sizeof(head));
93         memset(p,0,sizeof(p));
94         memset(d,0,sizeof(d));
95         cnt=1;
96         for(i=2;i<=n;i++){
97             scanf("%d",&x);
98             add(x,i);
99         }
100         dfs(1);
101         for(i=0;i<m;i++){
102             scanf("%d%d",&x,&y);
103             printf("%d\n",find_lca(x,y));
104         }
105     }
106     return 0;
107 }

```

最大流

```
1  #include<iostream>
2  #include<cstring>
3  #include<cstdio>
4  using namespace std;
5  const int N=20,M=1010,inf=2147483647;
6  struct edge
7  {
8      int v,w,next;
9  };
10 edge e[M*2];
11 int h[N],n,m,x,y,z,top,T;
12 bool vis[N];
13 void add_edge(int xx,int yy,int zz)
14 {
15     top++;
16     e[top]=(edge){yy,zz,h[xx]};
17     h[xx]=top;
18     top++;
19     e[top]=(edge){xx,0,h[yy]};
20     h[yy]=top;
21     return ;
22 }
23 int dfs(int now,int f)
24 {
25     if(now==n) return f;
26     vis[now]=true;
27     for(int i=h[now];i>0;i=e[i].next)
28         if(!vis[e[i].v] && e[i].w>0)
29         {
30             int d=dfs(e[i].v,min(f,e[i].w));
31             if(d>0)
32             {
33                 e[i].w-=d;
34                 e[((i-1)^1)+1].w+=d;
35                 return d;
36             }
37         }
38     return 0;
39 }
40 int max_flow()
41 {
42     int flow,f;
43     flow=0;
44     for(;;)
45     {
46         memset(vis,false,sizeof(vis));
47         f=dfs(1,inf);
48         if(f==0) return flow;
49         flow+=f;
```

```

50     }
51     return 0;
52 }
53 int main()
54 {
55     scanf("%d",&T);
56     for(int k=1;k<=T;k++)
57     {
58         memset(h,0,sizeof(h));
59         top=0;
60         scanf("%d%d",&n,&m);
61         for(int i=1;i<=m;i++)
62         {
63             scanf("%d%d%d",&x,&y,&z);
64             add_edge(x,y,z);
65         }
66         printf("Case %d: %d\n",k,max_flow());
67     }
68     return 0;
69 }

```

dinic当前弧优化

```

1  #include<iostream>
2  #include<algorithm>
3  #include<cstring>
4  #include<cstdio>
5  #include<cstdlib>
6  #include<cmath>
7  #include<map>
8  #include<vector>
9  #include<queue>
10 #include<set>
11 #define inf 2147483647
12 using namespace std;
13 typedef pair<int,int> pii;
14 const int N=210,M=210;
15 struct edge
16 {
17     int v,w,next;
18 };
19 edge e[M*2];
20 int n,m,x,y,z,h[N],cur[N],dep[N],top,st,ed,num;//num为节点数
21 void add_edge(int xx,int yy,int zz)
22 {
23     top++;
24     e[top]=(edge){yy,zz,h[xx]};
25     h[xx]=top;
26     top++;
27     e[top]=(edge){xx,0,h[yy]};
28     h[yy]=top;
29     return ;
30 }

```

```

31 bool bfs()
32 {
33     int head;
34     queue<int> q;
35     memset(dep,0,sizeof(dep));
36     q.push(st);
37     dep[st]=1;
38     while(!q.empty())
39     {
40         head=q.front();
41         q.pop();
42         if(head==ed)
43             return true;
44         for(int i=h[head];i>0;i=e[i].next)
45             if(dep[e[i].v]==0 && e[i].w>0)
46             {
47                 q.push(e[i].v);
48                 dep[e[i].v]=dep[head]+1;
49             }
50     }
51     return false;
52 }
53 int dfs(int now,int f)
54 {
55     if(now==ed)
56         return f;
57     int ff=0,temp;
58     for(int &i=cur[now];i>0;i=e[i].next)??
59     {
60         if(e[i].w>0 && dep[e[i].v]==dep[now]+1)
61         {
62             temp=dfs(e[i].v,min(e[i].w,f-ff));
63             e[i].w-=temp;
64             e[((i-1)^1)+1].w+=temp;
65             ff+=temp;
66             if(f==ff) return f;
67         }
68     }
69     return ff;
70 }
71 int max_flow()
72 {
73     int flow=0,ff;
74     while(bfs()){
75         for(int i=1;i<=num;i++)
76             cur[i]=h[i];
77         ff=dfs(st,inf);
78         if(ff==0)
79             break;
80         flow+=ff;
81     }
82     return flow;
83 }
84 int main()
85 {

```



```

86     memset(h,0,sizeof(h));
87     top=0;
88     scanf("%d%d",&m,&n);
89     num=n;
90     for(int i=1;i<=m;i++)
91     {
92         scanf("%d%d%d",&x,&y,&z);
93         add_edge(x,y,z);
94     }
95     printf("%d\n",max_flow());
96     return 0;
97 }

```

匈牙利算法

```

1  #include<iostream>
2  #include<cstring>
3  #include<cstdio>
4  using namespace std;
5  int edge[1010*2][1010*2],pei[1010*2];
6  bool vis[1010*2];
7  int n,m,ans,x,y,z;
8  bool findz(int x)
9  {
10     int i;
11     for (i=n+1;i<=m+n;i++)
12         if(edge[x][i] && !vis[i])
13         {
14             vis[i]=true;
15             if(!pei[i] || findz(pei[i]))
16             {
17                 pei[i]=x;
18                 return true;
19             }
20         }
21     return false;
22 }
23 void Hungary()
24 {
25     ans=0;
26     memset(pei,0,sizeof(pei));
27     for (int i=1;i<=n;i++)
28     {
29         memset(vis,0,sizeof(vis));
30         if (findz(i))
31             ans++;
32     }
33     return ;
34 }
35 int main()
36 {
37     // freopen("testdata.in","r",stdin);
38     cin>>n>>m>>z;

```

```
39     memset(edge,0,sizeof(edge));
40     for (int i=1;i<=z;i++)
41     {
42         cin>>x>>y;
43         if(x>n||y>m)
44             continue;
45         edge[x][y+n]=1;
46
47         edge[y+n][x]=1;
48     }
49     Hungary();
50     cout<<ans<<endl;
51     return 0;
52 }
```