Standard Code Library

Part5 - Graph Theory

Jiangxi Normal University

HeartFireY, eroengine, yezzz

November 7, 2022

Standard Code Library

HeartFireY, eroengine, yezzz

Jiangxi Normal University

November 7, 2022

Contents

图论

1.LCA

倍增

```cpp
int n;
vector<int>to[N];
int rt;
int depth[N];
int fa[N][21];//fa[u][i] 表示 u 开始向上 2^i 的父节点
void dfs_lca(int u, int father) {
    depth[u] = depth[father] + 1;
    fa[u][0] = father;
    for (int i = 1; (1 << i) <= depth[u]; i++) {
        fa[u][i] = fa[fa[u][i - 1]][i - 1];
    }
    for (auto ed : to[u]) {
        int v = ed;
        if (v != father) {
            dfs_lca(v, u);
        }
    }
}
int lca(int u,int v) {
    if (depth[u] < depth[v])swap(u, v);
    for (int k = 20; k >= 0; k--) {
        if (depth[fa[u][k]] >= depth[v]) {
            u = fa[u][k];
        }
    }
    if (u == v) {
        return u;
    }
    for (int k = 20; k >= 0; k--) {
        if (fa[u][k] != fa[v][k]) {
            u = fa[u][k];
            v = fa[v][k];
        }
    }
    return fa[u][0];
}
void solve() {
    cin >> n;
    for (int i = 1; i <= n; i++) {
        int u, v;
        cin >> u >> v;
        if (v == -1) {
            rt = u;
        }
        to[u].pb(v);
        to[v].pb(u);
    }
    dfs_lca(rt, 0);
    int q;
    cin >> q;
    while (q--) {
        int u, v;
        cin >> u >> v;
        cout << lca(u, v) << "\n";
    }
}
```

HLD 树剖

```cpp
#include <bits/stdc++.h>

#define maxm 200010
namespace LCA{
    struct edge{ int to, len, next; } E[maxm];
```

```
6     int cnt, last[maxm], fa[maxm], top[maxm], deep[maxm], siz[maxm], son[maxm], val[maxm];
7     void addedge(int a, int b, int len = 0){
8         E[++cnt] = (edge){b, len, last[a]}, last[a] = cnt;
9     }
10    void dfs1(int x){
11        deep[x] = deep[fa[x]] + 1;
12        siz[x] = 1;
13        for (int i = last[x]; i; i = E[i].next){
14            int to = E[i].to;
15            if (fa[x] != to && !fa[to]){
16                val[to] = E[i].len;
17                fa[to] = x;
18                dfs1(to);
19                siz[x] += siz[to];
20                if (siz[son[x]] < siz[to]) son[x] = to;
21            }
22        }
23    }
24    void dfs2(int x){
25        if (x == son[fa[x]]) top[x] = top[fa[x]];
26        else top[x] = x;
27        for (int i = last[x]; i; i = E[i].next)
28            if (fa[E[i].to] == x) dfs2(E[i].to);
29    }
30    void init(int root) { dfs1(root), dfs2(root); }
31    int query(int x, int y){
32        for (; top[x] != top[y]; deep[top[x]] > deep[top[y]] ? x = fa[top[x]] : y = fa[top[y]]);
33        return deep[x] < deep[y] ? x : y;
34    }
35 }
36 int n, m, x, y, v;
37 int main(){
38     scanf("%d%d", &n, &m);
39     for (int i = 1; i < n; i++){
40         scanf("%d%d", &x, &y);
41         LCA::addedge(x, y, v);
42         LCA::addedge(y, x, v);
43     }
44     LCA::init(1);
45     for (int i = 1; i <= m; i++){
46         scanf("%d%d", &x, &y);
47         printf("%d\n", LCA::query(x, y));
48     }
49     return 0;
50 }
```

2.Kruskal 重构树

```
1  const int N = 1e5 + 10;
2  int n = 0, m = 0;
3
4  namespace Graph
5  {
6      struct edge { int to, nxt, val; } edges[N << 1];
7      int cnt, head[N << 1], val[N << 1];
8      void add(int u, int v, int val = 0){
9          edges[++cnt] = (edge){v, head[u], val};
10         head[u] = cnt;
11     }
12 } // namespace Graph
13
14 namespace KR{
15     using Graph::add;
16     struct edge{
17         int u, v, w;
18         const bool operator< (const edge &x) const { return w < x.w; }
19     } edges[N] ;
20     int fa[N];
21     void init(int n){ for(int i = 1; i <= n; i++) fa[i] = i; }
22     int find(int x){ return fa[x] == x ? x : (fa[x] = find(fa[x])); }
23
```

```
24
25     void kruskal(){
26         int tot = 0, cnt = n;
27         sort(edges + 1, edges + 1 + m);
28         for(int i = 1; i <= m; i++){
29             int fau = find(edges[i].u), fav = find(edges[i].v);
30             if(fau != fav){
31                 cnt++, fa[fau] = fa[fav] = cnt;
32                 add(fau, cnt), add(cnt, fau);
33                 add(fav, cnt), add(cnt, fav);
34                 Graph::val[cnt] = edges[i].w;
35                 tot++;
36             }
37             if(tot == n - 1) break;
38         }
39
40     }
41 } // namespace KR
```

3.KM 二分图最大权匹配

```
1  /* 好难懂啊，还是抄板子吧 */
2  typedef long long ll;
3  typedef unsigned long long ull;//卡精度
4  typedef pair<int,int> PII;
5
6  const int N = 507, M = 5e3 +7, maxn = 1007;
7  const int mod = 1e9+7;
8  const ll INF = 1e15+7;
9
10 double w[N][N];//边权
11 double la[N], lb[N];//左、右部点的顶标
12 bool va[N], vb[N];//访问标记，是否在交错树中
13 int match[N];//右部点匹配的左部点（一个只能匹配一个嘛）
14 int n;
15 double delta, upd[N];
16 int p[N];
17 double c[N];
18
19 void bfs(int x)
20 {
21     int a, y = 0, y1 = 0;
22
23     for(int i = 1; i <= n; ++ i)
24         p[i] = 0, c[i] = INF;
25
26     match[y] = x;
27
28     do{
29         a = match[y], delta = INF, vb[y] = true;
30         for(int b = 1; b <= n; ++ b){
31             if(!vb[b]){
32                 if(c[b] > la[a] + lb[b] - w[a][b])
33                     c[b] = la[a] + lb[b] - w[a][b], p[b] = y;
34                 if(c[b] < delta)//？ 还是取最小的
35                     delta = c[b], y1 = b;
36             }
37         }
38         for(int b = 0; b <= n; ++ b)
39             if(vb[b])
40                 la[match[b]] -= delta, lb[b] += delta;
41             else c[b] -= delta;
42         y = y1;
43     }while(match[y]);
44     while(y) match[y] = match[p[y]], y = p[y];
45 }
46
47 double KM()
48 {
49     for(int i = 1; i <= n; ++ i)
50         match[i] = lb[i] = la[i] = 0;
```

```
51        for(int i = 1; i <= n; ++ i){
52            for(int j = 1; j <= n; ++ j)
53                vb[j] = false;
54            bfs(i);
55        }
56        double res = 0;
57        for(int y = 1; y <= n; ++ y)
58            res += w[match[y]][y];
59        return res;
60    }
```

4. 二分图匹配

```
1    bool dfs(int u, int tag)
2    {
3        if (vis[u] == tag) return false; // 即说明已经被访问了，就不需要清空了
4        // 其实只是将 bool 改成用数字标记，快的也很有限
5        vis[u] = tag;
6        for (auto v : e[u])
7            if(!pre[v] || dfs(pre[v], tag))
8            {
9                pre[v] = u;
10                return true;
11            }
12        return false;
13    }
```

5. 拓扑排序

```
1    const int N = 1e5 + 10;
2    std::vector<int> g[N];
3    int in_cnt[N];
4
5    bool TopoSort(int n){
6        std::vector<int> ans;
7        std::queue<int> q;
8        for(int i = 1; i <= n; i++) if(!in_cnt[i]) q.emplace(i);
9        while(q.size()){
10            int now = q.front(); q.pop();
11            ans.emplace_back(now);
12            for(auto nxt : g[now]) if(!--in_cnt[nxt]) q.emplace(nxt);
13        }
14        if(ans.size() == n){
15            for(auto ansi : ans) cout << ansi << ' ';
16            return true;
17        }
18        else return false;
19    }
20
21    inline void solve(){
22        int n, m; std::cin >> n >> m;
23        for(int i = 1; i <= m; i++){
24            int u, v; std::cin >> u >> v;
25            g[u].emplace_back(v);
26            in_cnt[v]++;
27        }
28        bool status = TopoSort(n);
29    }
```

6.EK+SPFA

```
1    #include <bits/stdc++.h>
2    using namespace std;
3
4    const int N = 5100, M = 20010, INF = 1e8;
5    int n, m, s, t;
6    template <int N,int M>struct MCMF{
7        //EK+spfa
8        const int INF=1e8;
```

```
9        int S,T;
10       MCMF(int s,int t):S(s),T(t){}
11       int h[N], e[M], f[M], w[M], ne[M], idx;
12       int q[N], d[N], pre[N], incf[N];
13       bool st[N];
14
15       void add(int a, int b, int c, int d)
16       {
17           e[idx] = b, f[idx] = c, w[idx] = d, ne[idx] = h[a], h[a] = idx ++ ;
18           e[idx] = a, f[idx] = 0, w[idx] = -d, ne[idx] = h[b], h[b] = idx ++ ;
19       }
20
21       bool spfa()
22       {
23           int hh = 0, tt = 1;
24           memset(d, 0x3f, sizeof d);
25           memset(incf, 0, sizeof incf);
26           q[0] = S, d[S] = 0, incf[S] = INF;
27           while (hh != tt)
28           {
29               int t = q[hh ++ ];
30               if (hh == N) hh = 0;
31               st[t] = false;
32
33               for (int i = h[t]; ~i; i = ne[i])
34               {
35                   int ver = e[i];
36                   if (f[i] && d[ver] > d[t] + w[i])
37                   {
38                       d[ver] = d[t] + w[i];
39                       pre[ver] = i;
40                       incf[ver] = min(f[i], incf[t]);
41                       if (!st[ver])
42                       {
43                           q[tt ++ ] = ver;
44                           if (tt == N) tt = 0;
45                           st[ver] = true;
46                       }
47                   }
48               }
49           }
50
51           return incf[T] > 0;
52       }
53
54       void EK(int& flow, int& cost)
55       {
56           flow = cost = 0;
57           while (spfa())
58           {
59               int t = incf[T];
60               flow += t, cost += t * d[T];
61               for (int i = T; i != S; i = e[pre[i] ^ 1])
62               {
63                   f[pre[i]] -= t;
64                   f[pre[i] ^ 1] += t;
65               }
66           }
67       }
68   };
69   signed main()
70   {
71       int x;
72       cin>>n;
73       s=0,t=n+n+1;
74       MCMF<N,M>mcmf1(s,t);
75       MCMF<N,M>mcmf2(s,t);
76
77       memset(mcmf1.h, -1, sizeof mcmf1.h);
78       for(int i=1;i<=n;i++){
79           for(int j=1;j<=n;j++){
```

```
80          cin>>x;
81          mcmf1.add(i,j+n,1,x);
82          mcmf2.add(i,j+n,1,-x);
83      }
84      mcmf1.add(0,i,1,0);
85      mcmf1.add(i+n,n+n+1,1,0);
86      mcmf2.add(0,i,1,0);
87      mcmf2.add(i+n,n+n+1,1,0);
88  }
89
90  int flow, cost,flow2,cost2;
91  mcmf1.EK(flow, cost);
92  //mcmf2.EK(flow2, cost2);
93  printf("%d %d\n",cost,-cost2);
94
95  return 0;
96 }
```

7.Dinic

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  template <int N> struct Dinic {
4      const int INF = 1e9;
5      struct E {
6          int to, cap, rev;
7      };
8      vector<E> G[N];
9      int lev[N], cur[N];
10     inline void add(int x, int y, int c) {
11         G[x].push_back({ y, c, (int)G[y].size() });
12         G[y].push_back({ x, 0, (int)G[x].size() - 1 });
13     }
14     void bfs(int s) {
15         queue<int> q;
16         memset(lev, -1, sizeof lev);
17
18         for (lev[s] = 0, q.push(s); q.size();) {
19             int x = q.front();
20             q.pop();
21
22             for (auto &e : G[x])
23                 if (e.cap > 0 && lev[e.to] < 0)
24                     lev[e.to] = lev[x] + 1, q.push(e.to);
25         }
26     }
27     int dfs(int x, int t, int f) {
28         if (x == t)
29             return f;
30
31         for (int &i = cur[x], sz = G[x].size(), d; i < sz; i++) {
32             auto &e = G[x][i];
33
34             if (e.cap > 0 && lev[x] < lev[e.to]) {
35                 if ((d = dfs(e.to, t, min(f, e.cap))) > 0) {
36                     e.cap -= d, G[e.to][e.rev].cap += d;
37                     return d;
38                 }
39             }
40         }
41
42         return 0;
43     }
44     int64_t maxflow(int s, int t) {
45         for (int64_t flow = 0, f;;) {
46             bfs(s);
47
48             if (lev[t] < 0)
49                 return flow;
50
51             memset(cur, 0, sizeof cur);
```

```
52              while ((f = dfs(s, t, INF)) > 0)
53                  flow += f;
54          }
55      }
56  };
57  Dinic<1005> din;
58  signed main() {
59      cin.tie(0)->sync_with_stdio(0);
60      int n, m, s, t;
61      cin >> n ;
62      for(int i=1;i<=n;i++){
63          for(int j=1;j<=n;j++){
64              int x;
65              cin>>x;
66              din.add(0,i,x);
67              din.add(j+100,201,x);
68              din.add(i,j+100,1e9);
69          }
70      }
71      return cout << din.maxflow(0, 201), 0;
72  }
```

8.MCMF（dij+ek）

```
1   #include <bits/stdc++.h>
2   using namespace std;
3   const int mod = 1000000007,N = 100005,inf = 1e9;
4   template <int N> struct MCMF{
5       struct E{
6           int to,cap,val,inv;
7       };
8       vector <E> g[N];
9       int dis[N],now[N],h[N],pre[N],preu[N];
10      void add(int u,int v,int f,int w){
11          g[u].push_back({v,f,w,(int)g[v].size()});
12          g[v].push_back({u,0,-w,(int)g[u].size()-1});
13      }
14      void dijkstra(int st){
15          priority_queue<pair<int,int>,vector<pair<int,int>>,greater<pair<int,int>>>q;
16          memset(dis,0x3f,sizeof dis);
17          memset(pre,-1,sizeof pre);
18          memset(preu,-1,sizeof preu);
19          dis[st]=0;q.push({0,st});
20          while(q.size()){
21              auto [d,u]=q.top();q.pop();
22              if(dis[u]<d)continue;
23              int x=0;
24              for(auto [v,f,w,inv]:g[u]){
25                  if(f&&dis[v]>dis[u]+w+h[u]-h[v]){
26                      dis[v]=dis[u]+h[u]-h[v]+w;
27                      pre[v]=x;
28                      preu[v]=u;
29                      q.push({dis[v],v});
30                  }
31                  x++;
32              }
33          }
34      }
35      pair<int,int> min_cost_max_flow(int st,int ed){
36          memset(h,0,sizeof h);
37          for(int flow=0,cost=0,res=inf;;res=inf){
38
39              dijkstra(st);
40              if(dis[ed]>inf)return {flow,cost};
41              for(int i=0;i<N;i++){
42                  h[i]+=dis[i];
43              }
44              for(int i=ed;i!=st;i=preu[i]){
45                  res=min(res,g[preu[i]][pre[i]].cap);
46              }
```

```
47          flow+=res;
48          cost+=res*h[ed];
49          for(int i=ed;i!=st;i=preu[i]){
50              g[i][g[preu[i]][pre[i]].inv].cap+=res;
51              g[preu[i]][pre[i]].cap-=res;
52          }
53      }
54   }
55 };
56 MCMF<505>mcmf;
57 int n,m,s,t;
58 signed main(){
59     cin>>n>>m>>s>>t;
60     for(int i=1;i<=m;i++){
61         int u,v,w,c;cin>>u>>v>>w>>c;
62         mcmf.add(u,v,w,c);
63     }
64     auto [f,c]=mcmf.min_cost_max_flow(s,t);
65     cout<<f<<" "<<c<<endl;
66 }
```

9.Scc_Tarjan(有向图强连通分量)

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  int n,m;
4  int newid[200005];
5  vector<int>p[200005];
6  vector<vector<int>>scc;
7  int dfn[200005],low[200005],ins[200005],idx,cnt;
8  vector<int>st,f;
9  void scc_tarjan(int u){
10     low[u]=dfn[u]=++idx;
11     st.push_back(u);
12     ins[u]=1;
13     for(auto v:p[u]){
14         if(!dfn[v])scc_tarjan(v);
15         if(ins[v])low[u]=min(low[u],low[v]);
16     }
17     if(low[u]==dfn[u]){
18         ++cnt;
19         f.clear();
20         while(1){
21             int v=st.back();st.pop_back();
22             f.push_back(v);
23             ins[v]=0;
24             newid[v]=cnt;
25             if(u==v)break;
26         }
27         scc.push_back(f);
28     }
29 }
30
31 signed main(){
32     cin>>n>>m;
33     for(int i=1;i<=m;i++){
34         int u,v;cin>>u>>v;
35         p[u].push_back(v);
36     }
37     for(int i=1;i<=n;i++){
38         if(dfn[i])continue;
39         scc_tarjan(i);
40     }
41 }
```

10.Edcc_Tarjan（边双连通分量）

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  int n,m;
```

```cpp
int newid[500005],cnt;
vector<int>g[200005];
vector<vector<int>>Edcc;
//下部分为不同题目所需变量
vector<int>ans;

//下为原图 + 跑 tarjan 所需变量
vector<pair<int,int>>p[500005];
int dfn[500005],low[500005],idx;
vector<int>st,f;
void Edcc_tarjan(int u,int id){
    low[u]=dfn[u]=++idx;
    st.push_back(u);
    for(auto [v,x]:p[u]){
        if(!dfn[v])Edcc_tarjan(v,x);
        if(id!=x)low[u]=min(low[u],low[v]);
    }
    if(id!=0&&low[u]==dfn[u]){
        f.push_back(id);
    }
    if(low[u]==dfn[u]){
        ++cnt;
        f.clear();
        while(1){
            int v=st.back();st.pop_back();
            f.push_back(v);
            newid[v]=cnt;
            if(u==v)break;
        }
        Edcc.push_back(f);
    }
}
signed main(){
    cin>>n>>m;
    for(int i=1;i<=m;i++){
        int u,v;cin>>u>>v;
        p[u].push_back({v,i});
        p[v].push_back({u,i});
    }
    for(int i=1;i<=n;i++){
        if(dfn[i])continue;
        Edcc_tarjan(i,0);
    }
    cout<<Edcc.size()<<endl;
    for(auto &x:Edcc){
        cout<<x.size()<<" ";
        for(auto &y:x){
            cout<<y<<" ";
        }
        cout<<endl;
    }
}
```

11.Vdcc_Tarjan

```cpp
#include <bits/stdc++.h>
using namespace std;
int n,m;
int newid[500005],cnt,cut[500005];
vector<int>g[200005];
vector<vector<int>>Vdcc;
//下部分为不同题目所需变量
vector<int>ans;
//下为原图 + 跑 tarjan 所需变量
vector<int>p[500005];
int dfn[500005],low[500005],idx;
vector<int>st,f;

void Vdcc_tarjan(int u,int fa){
    low[u]=dfn[u]=++idx;
    st.push_back(u);
```

```cpp
        int tot=0;
        for(auto v:p[u]){
            if(v==fa)continue;
            if(!dfn[v]){
                Vdcc_tarjan(v,u);tot++;
                low[u]=min(low[v],low[u]);
                if(low[v]>=dfn[u]){
                    f.clear();
                    f.push_back(u);
                    while(1){
                        int x=st.back();st.pop_back();
                        f.push_back(x);
                        if(x==v)break;
                    }
                    Vdcc.push_back(f);
                }
            }
            low[u]=min(low[u],dfn[v]);
        }
        if(fa==0&&tot==0){
            f.clear();f.push_back(u);Vdcc.push_back(f);
        }
}
signed main(){
    cin>>n>>m;
    for(int i=1;i<=m;i++){
        int u,v;cin>>u>>v;
        if(u==v)continue;
        p[u].push_back(v);
        p[v].push_back(u);
    }
    for(int i=1;i<=n;i++){
        if(dfn[i])continue;
        Vdcc_tarjan(i,0);
    }
    cout<<Vdcc.size()<<endl;
    for(auto &x:Vdcc){
        cout<<x.size()<<" ";
        for(auto &y:x){
            cout<<y<<" ";
        }
        cout<<endl;
    }
}
```