



Standard Code Library

Part5 - Graph Theory

Jiangxi Normal University
HeartFireY, eroengine, yezzz

September 10, 2022

Standard Code Library

FOREIGNERS

Jiangxi Normal University HeartFireY

September 1202

Contents

图论	2
1.LCA	2
倍增	2
HLD 树剖	2
2.Kruskal 重构树	3
3.KM 二分图最大权匹配	4
4. 二分图匹配	4
5. 拓扑排序	5
6.EK+SPFA	5
7.Dinic	7
8.MCMF (dij+ek)	8
9.Scc_Tarjan(有向图强连通分量)	9
10.Edcc_Tarjan (边双连通分量)	9
11.Vdcc_Tarjan	10

图论

1.LCA

倍增

```
1  int n;
2  vector<int>to[N];
3  int rt;
4  int depth[N];
5  int fa[N][21]; //fa[u][i] 表示 u 开始向上 2^i 的父节点
6  void dfs_lca(int u, int father) {
7      depth[u] = depth[father] + 1;
8      fa[u][0] = father;
9      for (int i = 1; (1 << i) <= depth[u]; i++) {
10         fa[u][i] = fa[fa[u][i-1]][i-1];
11     }
12     for (auto ed : to[u]) {
13         int v = ed;
14         if (v != father) {
15             dfs_lca(v, u);
16         }
17     }
18 }
19 int lca(int u, int v) {
20     if (depth[u] < depth[v]) swap(u, v);
21     for (int k = 20; k >= 0; k--) {
22         if (depth[fa[u][k]] >= depth[v]) {
23             u = fa[u][k];
24         }
25     }
26     if (u == v) {
27         return u;
28     }
29     for (int k = 20; k >= 0; k--) {
30         if (fa[u][k] != fa[v][k]) {
31             u = fa[u][k];
32             v = fa[v][k];
33         }
34     }
35     return fa[u][0];
36 }
37 void solve() {
38     cin >> n;
39     for (int i = 1; i <= n; i++) {
40         int u, v;
41         cin >> u >> v;
42         if (v == -1) {
43             rt = u;
44         }
45         to[u].pb(v);
46         to[v].pb(u);
47     }
48     dfs_lca(rt, 0);
49     int q;
50     cin >> q;
51     while (q--) {
52         int u, v;
53         cin >> u >> v;
54         cout << lca(u, v) << "\n";
55     }
56 }
57
```

HLD 树剖

```
1  #include <bits/stdc++.h>
2
3  #define maxm 200010
4  namespace LCA{
5      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::addege(x, y, v);
42          LCA::addege(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 namespace RMatch{
2     #define LL long long
3     const int M = 400 + 5, INF = 2E9;
4     int n = 0; /// Attention-Outside !//
5     int w[M][M], kx[M], ky[M], linky[M], vy[M], slk[M], pre[M];
6
7     int KM(){
8         for(int i = 1; i <= n; i++)
9             for(int j = 1; j <= n; j++) kx[i] = max(kx[i], w[i][j]);
10        for(int i = 1; i <= n; i++) {
11            fill(vy, vy + n + 1, 0);
12            fill(slk, slk + n + 1, INF);
13            fill(pre, pre + n + 1, 0);
14            int k = 0, p = -1;
15            for(linky[k = 0] = i; linky[k]; k = p){
16                vy[k] = 1;
17                int d = INF, x = linky[k];
18                for(int j = 1; j <= n; j++){
19                    if (!vy[j]) {
20                        int t = kx[x] + ky[j] - w[x][j];
21                        if (t < slk[j]) { slk[j] = t; pre[j] = k; }
22                        if (slk[j] < d) { d = slk[j]; p = j; }
23                    }
24                }
25                for(int j = 0; j <= n; j++){
26                    if (vy[j]) { kx[linky[j]] -= d; ky[j] += d; }
27                    else slk[j] -= d;
28                }
29                for(; k; k = pre[k]) linky[k] = linky[pre[k]];
30            }
31            int ans = 0;
32            for(int i = 1; i <= n; i++) ans += kx[i] + ky[i];
33            return ans;
34        }
35    }
36    inline void add(int x, int y, int val){ w[x][y] = val; }
37 }

```

4. 二分图匹配

```

1 /*
2  ! + 最小覆盖数 = 最大匹配数
3  ! + 最大独立集 = 顶点数 - 二分图匹配数
4  ! + DAG 最小路径覆盖数 = 结点数 - 拆点后二分图最大匹配数
5  */
6
7 const int N = 1e3 + 10;
8 struct MaxMatch {
9     int n;
10     vector<int> g[N];

```

```

11     int vis[N], left[N], clk;
12
13     void init(int n) {
14         this->n = n;
15         for(int i = 0; i <= n; i++) g[i].clear();
16         memset(left, -1, sizeof left);
17         memset(vis, -1, sizeof vis);
18     }
19
20     bool dfs(int u) {
21         for (int v: g[u])
22             if (vis[v] != clk) {
23                 vis[v] = clk;
24                 if (left[v] == -1 || dfs(left[v])) {
25                     left[v] = u;
26                     return true;
27                 }
28             }
29         return false;
30     }
31
32     int match() {
33         int ret = 0;
34         for (clk = 0; clk <= n; ++clk)
35             if (dfs(clk)) ++ret;
36         return ret;
37     }
38 };

```

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
53         while ((f = dfs(s, t, INF)) > 0)
54             flow += f;
55     }
56 }
57 };
58 Dinic<1005> din;
59 signed main() {
60     cin.tie(0)->sync_with_stdio(0);
61     int n, m, s, t;
62     cin >> n ;
63     for(int i=1;i<=n;i++){
64         for(int j=1;j<=n;j++){
65             int x;
66             cin>>x;
67             din.add(0,i,x);
68             din.add(j+100,201,x);
69             din.add(i,j+100,1e9);
70         }
71     }
72     return cout << din.maxflow(0, 201), 0;
73 }

```

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;
4  int newid[500005],cnt;
5  vector<int>g[200005];
6  vector<vector<int>>Edcc;
7  //下部分为不同题目所需变量
8  vector<int>ans;
9
10 //下为原图 + 跑 tarjan 所需变量
11 vector<pair<int,int>>p[500005];
12 int dfn[500005],low[500005],idx;
13 vector<int>st,f;
14 void Edcc_tarjan(int u,int id){
15     low[u]=dfn[u]++;idx;
16     st.push_back(u);
17     for(auto [v,x]:p[u]){
18         if(!dfn[v])Edcc_tarjan(v,x);
19         if(id!=x)low[u]=min(low[u],low[v]);
20     }
21     if(id!=0&&low[u]==dfn[u]){
22         f.push_back(id);
23     }
24     if(low[u]==dfn[u]){
25         ++cnt;
26         f.clear();
27         while(1){
28             int v=st.back();st.pop_back();
29             f.push_back(v);
30             newid[v]=cnt;
31             if(u==v)break;
32         }
33         Edcc.push_back(f);
34     }
35 }
36 signed main(){
37     cin>>n>>m;
38     for(int i=1;i<=m;i++){
39         int u,v;cin>>u>>v;
40         p[u].push_back({v,i});
41         p[v].push_back({u,i});
42     }
43     for(int i=1;i<=n;i++){
44         if(dfn[i])continue;
45         Edcc_tarjan(i,0);
46     }
47     cout<<Edcc.size()<<endl;
48     for(auto &x:Edcc){
49         cout<<x.size()<<" ";
50         for(auto &y:x){
51             cout<<y<<" ";
52         }
53         cout<<endl;
54     }
55 }

```

11.Vdcc_Tarjan

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  int n,m;
4  int newid[500005],cnt,cut[500005];
5  vector<int>g[200005];
6  vector<vector<int>>Vdcc;
7  //下部分为不同题目所需变量
8  vector<int>ans;
9  //下为原图 + 跑 tarjan 所需变量
10 vector<int>p[500005];
11 int dfn[500005],low[500005],idx;
12 vector<int>st,f;
13
14 void Vdcc_tarjan(int u,int fa){
15     low[u]=dfn[u]++;idx;

```

```

16     st.push_back(u);
17     int tot=0;
18     for(auto v:p[u]){
19         if(v==fa)continue;
20         if(!dfn[v]){
21             Vdcc_tarjan(v,u);tot++;
22             low[u]=min(low[v],low[u]);
23             if(low[v]>=dfn[u]){
24                 f.clear();
25                 f.push_back(u);
26                 while(1){
27                     int x=st.back();st.pop_back();
28                     f.push_back(x);
29                     if(x==v)break;
30                 }
31                 Vdcc.push_back(f);
32             }
33         }
34         low[u]=min(low[u],dfn[v]);
35     }
36     if(fa==0&&tot==0){
37         f.clear();f.push_back(u);Vdcc.push_back(f);
38     }
39 }
40 signed main(){
41     cin>>n>>m;
42     for(int i=1;i<=m;i++){
43         int u,v;cin>>u>>v;
44         if(u==v)continue;
45         p[u].push_back(v);
46         p[v].push_back(u);
47     }
48     for(int i=1;i<=n;i++){
49         if(dfn[i])continue;
50         Vdcc_tarjan(i,0);
51     }
52     cout<<Vdcc.size()<<endl;
53     for(auto &x:Vdcc){
54         cout<<x.size()<<" ";
55         for(auto &y:x){
56             cout<<y<<" ";
57         }
58         cout<<endl;
59     }
60 }

```