图论与搜索

1. **基础算法**
    a. 离散化
2. **搜索**
    a. 双向bfs
3. **图论**
    a. spfa
    b. spfa判负环
    c. 二分图的最大匹配
    d. 最近公共祖先在线做法(LCA/ST表)(O(nlogn+m))
    e. 最近公共祖先离线做法(Tarjan)(O(n+m))
    f. Tarjan算法求有向图的最大连通分量O(n+m)
4. **网络流**
    a. 最大流之Dinic算法

基础算法

离散化

```
#include<bits/stdc++.h>
using namespace std;
map<int, int, greater<int>>nu;
int n, m, x, c, r, l;
int main() {
    cin >> n >> m;
    nu[-0x3f3f3f3f] = 0;
    for (int i = 1;i <= n;i++) {
        cin >> x >> c;
        nu[x] += c;
    }   //加C

    for (map<int, int>::reverse_iterator h = ++nu.rbegin(), q = nu.rbegin();h != nu.rend();h++, q++) {
        h->second += q->second;
    }   //求前缀和

    for (int i = 1;i <= m;i++) {
        cin >> l >> r;
        cout << nu.lower_bound(r)->second - nu.upper_bound(l)->second << endl;
    }   //询问区间和

    return 0;
}
```

搜索

双向bfs

```
#include<bits/stdc++.h>
#include<unordered_map>
using namespace std;
const int MAXN = 6;
string a[MAXN];
string b[MAXN];

int n = 0;
int extend(queue<string>& q, unordered_map<string, int>& da, unordered_map<string, int>& db,string a[],string b[]) {
    for (int k = 0, sk = q.size();k < sk;k++) {
        string t = q.front();
        q.pop();
        for (int i = 0;i < t.size();i++) {
            for (int j = 0;j < n;j++) {
                if (t.substr(i, a[j].size()) == a[j]) {
                    string state = t.substr(0, i) + b[j] + t.substr(i + a[j].size(), t.size());
                    if (da.find(state) == da.end()) {
                        da[state] = da[t] + 1;
                        q.push(state);
                        if (db.find(state) != db.end()) {
                            return da[state] + db[state];
                        }
                    }
                }
            }
        }
    }
    return 11;
}


int bfs(string x, string y) {
    queue<string>qa;
    queue<string>qb;
    unordered_map<string, int>da;
    unordered_map<string, int>db;
    qa.push(x);
    da[x] = 0;
    qb.push(y);
    db[y] = 0;
    int step = 1;
    while ((!qa.empty()) && (!qb.empty()))
    {
        int temp;
        if (qa.size() < qb.size())temp = extend(qa, da, db,a,b);
        else temp = extend(qb, db, da,b,a);
        if (temp <= 10)return temp;
        step++;
        if (step > 10) {//判读是否在10步以内
            break;
        }
    }
    return 11;
}

int main() {
    string x, y;
    cin >> x >> y;
    while (cin >> a[n] >> b[n])n++;
    int temp = bfs(x, y);
    if (temp <= 10) {
        cout << temp << endl;
    }
    else {
        cout << "NO ANSWER!" << endl;
    }
    return 0;
}
```

图论

spfa

```
#include<bits/stdc++.h>
using namespace std;
const int N=100010;
const int inf=0x3f3f3f3f;
int n,m;
int h[N];
int idx=1;
struct ee{
  int  v;
  int  w;
  int  nt;
};
ee e[N];
bool vis[N];
int dis[N];
void add(int u,int v,int w){
    e[idx].v=v;
    e[idx].w=w;
    e[idx].nt=h[u];
    h[u]=idx++;
}
void spfa(){
    memset(dis,0x3f,sizeof dis);
    dis[1]=0;
    queue<int>q;
    q.push(1);
    vis[1]=1;
    while(!q.empty()){
        int ram=q.front();
        q.pop();
        vis[ram]=0;
        for(int i=h[ram];i;i=e[i].nt){
            if(dis[ram]+e[i].w<dis[e[i].v]){
                dis[e[i].v]=dis[ram]+e[i].w;
                if(!vis[e[i].v]){
                    vis[e[i].v]=1;
                    q.push(e[i].v);
                }
            }
        }
    }
}
int a,b,c;
int main(){
    scanf("%d%d",&n,&m);
    for(int i=1;i<=m;i++){
        scanf("%d%d%d",&a,&b,&c);
        add(a,b,c);
    }
    spfa();
    if(dis[n]!=inf)
    printf("%d",dis[n]);
    else
    puts("impossible");
    return 0;
}
```

spfa判负环

```
#include<bits/stdc++.h>
using namespace std;
```

```
const int MAXN=1e5+5;
const int inf=0x3f3f3f3f;
typedef pair<int,int> pii;
struct ee{
  int v;
  int w;
  int nt=0;
};
ee e[MAXN];
int h[MAXN];
bool vis[MAXN];
int dis[MAXN];
int idx=1;
int n,m;
void add(int u,int v,int w){
    e[idx].v=v;
    e[idx].w=w;
    e[idx].nt=h[u];
    h[u]=idx++;
}

bool spfa(){
    queue<pii>q;
    memset(dis,0x3f,sizeof dis);
    dis[1]=0;
    for(int i=1;i<=n;i++){
        q.push(pii(i,0));
        vis[i]=1;
    }
    while(!q.empty()){
        pii ram=q.front();
        if(ram.second>=n-1){
            return 1;
        }
        q.pop();
        vis[ram.first]=0;
        //cout<<"？？？？？？？？？？？？？"<<endl;
        for(int i=h[ram.first];i;i=e[i].nt){
                  //  cout<<i<<endl;
            if(dis[e[i].v]>dis[ram.first]+e[i].w){

                dis[e[i].v]=dis[ram.first]+e[i].w;
                if(!vis[e[i].v]){
                    q.push(pii(e[i].v,ram.second+1));

                    vis[e[i].v]=1;
                }
            }
        }
    }
    return 0;
}
int a,b,c;
int main(){
    scanf("%d%d",&n,&m);
    for(int i=1;i<=m;i++){
        scanf("%d%d%d",&a,&b,&c);
        if(a==b&&c<0){
            puts("Yes");
            return 0;
        }else{
        add(a,b,c);
        }
    }
    if(spfa()){
        puts("Yes");
    }else{
        puts("No");
    }
    return 0;
}
```

二分图的最大匹配

```
#include<bits/stdc++.h>
#include<unordered_set>
using namespace std;
const int MAXN=1e5+5;
int ts;
unordered_set<int>nu[MAXN];
int match[MAXN];
bool vis[MAXN];
int n1,n2,m;
int sum=0;
int a,b;

bool dfs(int num){
    for(auto e:nu[num]){
        if(!vis[e]){
        vis[e]=1;
        if((!match[e])||dfs(match[e])){
            match[e]=num;
            return 1;
        }
        }
    }
    return 0;
}

int main(){
    scanf("%d%d%d",&n1,&n2,&m);
    for(int i=0;i<m;i++){
        scanf("%d%d",&a,&b);
        nu[a].insert(b);
    }
    for(int i=1;i<=n1;i++){
        memset(vis,0,sizeof vis);
        if(dfs(i))sum++;
    }
    cout<<sum<<endl;
    return 0;
}
```

最近公共祖先在线做法(LCA/ST表)(O(nlogn+m))

```
#include <bits/stdc++.h>
using namespace std;
const int N = 40010, M = N * 3;

struct Node
{
    int data;
    int nt;
};

int idx;
Node cun[M];
int h[N];
int depth[N], fa[N][16];

void add(int a, int b)
{
    cun[idx].data = b;
    cun[idx].nt = h[a];
    h[a] = idx;
    idx++;
}

void bfs(int root)
{
    memset(depth, 0x3f, sizeof depth);
    depth[0] = 0;
    depth[root] = 1;
    queue<int> q;
```

```
    q.push(root);
    while (!q.empty())
    {
        int t = q.front();
        q.pop();
        for (int i = h[t]; i; i = cun[i].nt)
        {
            if (depth[cun[i].data] > depth[t] + 1)
            {
                depth[cun[i].data] = depth[t] + 1;
                q.push(cun[i].data);
                fa[cun[i].data][0] = t;
                for (int k = 1; k <= 15; k++)
                {
                    fa[cun[i].data][k] = fa[fa[cun[i].data][k - 1]][k - 1];
                }
            }
        }
    }
}

int lca(int a, int b)
{
    if (depth[a] < depth[b])
        swap(a, b);
    for (int k = 15; k >= 0; k--)
    {
        if (depth[fa[a][k]] >= depth[b])
        {
            a = fa[a][k];
        }
    }
    if (a == b)
        return a;
    for (int k = 15; k >= 0; k--)
    {
        if (fa[a][k] != fa[b][k])
        {
            a = fa[a][k];
            b = fa[b][k];
        }
    }
    return fa[a][0];
}

int main()
{
    int n, m;
    scanf("%d", &n);
    int root = 0;

    for (int i = 0; i < n; i++)
    {
        int a, b;
        scanf("%d%d", &a, &b);
        if (b == -1)
            root = a;
        else
            add(a, b), add(b, a);
    }

    bfs(root);

    scanf("%d", &m);
    while (m--)
    {
        int a, b;
        scanf("%d%d", &a, &b);
        int p = lca(a, b);
        if (p == a)
            puts("1");
        else if (p == b)
            puts("2");
        else
            puts("0");
```

```
    }

    return 0;
}
```

最近公共祖先离线做法(Tarjan)(O(n+m))

```
#include <bits/stdc++.h>
using namespace std;
const int MAXN = 1e4 + 5;
typedef pair<int, int> pii;
struct Node
{
    int data;
    int w;
    int nt;
};

Node cun[MAXN * 3];
vector<pii> query[MAXN];
int dist[MAXN];
int res[MAXN * 2];
int vis[MAXN];
int ancs[MAXN];
int h[MAXN];
int idx = 1;

void init()
{
    for (int i = 0; i < MAXN; i++)
    {
        ancs[i] = i;
    }
}

void add(int u, int v, int w)
{
    cun[idx] = {v, w, h[u]};
    h[u] = idx++;
}

void dfs(int u, int fa)
{
    for (int i = h[u]; i; i = cun[i].nt)
    {
        if (cun[i].data == fa)
            continue;
        dist[cun[i].data] = dist[u] + cun[i].w;
        dfs(cun[i].data, u);
    }
}

int find_ancs(int u)
{
    return ancs[u] == u ? u : ancs[u] = find_ancs(ancs[u]);
}

void tarjan(int u)
{
    vis[u] = 1;
    for (int i = h[u]; i; i = cun[i].nt)
    {
        if (!vis[cun[i].data])
        {
            tarjan(cun[i].data);
            ancs[cun[i].data] = u;
        }
    }

    for (pii e : query[u])
    {
        if (vis[e.first] == 2)
```

```
        {
            int acs = find_ancs(e.first);
            res[e.second] = dist[u] + dist[e.first] - dist[acs] - dist[acs];
        }
    }

    vis[u] = 2;
}

int main()
{
    init();
    int n, m;
    scanf("%d%d", &n, &m);
    int a, b, c;

    for (int i = 0; i < n - 1; i++)
    {
        scanf("%d%d%d", &a, &b, &c);
        add(a, b, c);
        add(b, a, c);
    }

    for (int i = 0; i < m; i++)
    {
        scanf("%d%d", &a, &b);
        if (a != b)
        {
            query[a].push_back({b, i});
            query[b].push_back({a, i});
        }
    }

    dfs(1, -1);

    tarjan(1);

    for (int i = 0; i < m; i++)
    {
        printf("%d\n", res[i]);
    }

    return 0;
}
```

Tarjan算法求有向图的最大连通分量O(n+m)

```
#include <cstdio>
#include <cstring>
#include <iostream>
#include <algorithm>

using namespace std;

const int N = 10010, M = 50010;

int n, m;
int h[N], e[M], ne[M], idx;
int dfn[N], low[N], timestamp;
int stk[N], top;
bool in_stk[N];
int id[N], scc_cnt, Size[N];
int dout[N];

void add(int a, int b)
{
    e[idx] = b, ne[idx] = h[a], h[a] = idx ++ ;
}

void tarjan(int u)
{
    dfn[u] = low[u] = ++ timestamp;
```

```
    stk[ ++ top] = u, in_stk[u] = true;
    for (int i = h[u]; i != -1; i = ne[i])
    {
        int j = e[i];
        if (!dfn[j])
        {
            tarjan(j);
            low[u] = min(low[u], low[j]);
        }
        else if (in_stk[j]) low[u] = min(low[u], dfn[j]);
    }

    if (dfn[u] == low[u])
    {
        ++ scc_cnt;
        int y;
        do {
            y = stk[top -- ];
            in_stk[y] = false;
            id[y] = scc_cnt;
            Size[scc_cnt] ++ ;
        } while (y != u);
    }
}

int main()
{
    scanf("%d%d", &n, &m);
    memset(h, -1, sizeof h);
    while (m -- )
    {
        int a, b;
        scanf("%d%d", &a, &b);
        add(a, b);
    }

    for (int i = 1; i <= n; i ++ )
        if (!dfn[i])
            tarjan(i);

    for (int i = 1; i <= n; i ++ )
        for (int j = h[i]; ~j; j = ne[j])
        {
            int k = e[j];
            int a = id[i], b = id[k];
            if (a != b) dout[a] ++ ;
        }
//重新判边，这步很重要

    int zeros = 0, sum = 0;
    for (int i = 1; i <= scc_cnt; i ++ )
        if (!dout[i])
        {
            zeros ++ ;
            sum += Size[i];
            if (zeros > 1)
            {
                sum = 0;
                break;
            }
        }

    printf("%d\n", sum);

    return 0;
}
```

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e4 + 5;
const int M = 5e4 + 5;
struct Node
{
```

```
    int data;
    int nt;
};
int n, m;
int h[N], idx = 1;
Node cun[M];
int dfn[N], low[N], timestamp;
stack<int> stk;
bool in_stk[N];
int id[N], scc_cnt, Size[N];
int dout[N];

void add(int a, int b)
{
    cun[idx] = {b, h[a]};
    h[a] = idx++;
}

void tarjan(int u)
{
    dfn[u] = low[u] = ++timestamp;
    in_stk[u] = 1;
    stk.push(u);
    for (int i = h[u]; i; i = cun[i].nt)
    {
        if (!dfn[cun[i].data])
        {
            tarjan(cun[i].data);
            low[u] = min(low[u], low[cun[i].data]);
        }
        else if (in_stk[cun[i].data])
        {
            low[u] = min(low[u], dfn[cun[i].data]);
        }
    }

    if (dfn[u] == low[u])
    {
        ++scc_cnt;
        int y;
        do
        {
            y = stk.top();
            stk.pop();
            in_stk[y] = 0;
            id[y] = scc_cnt;
            Size[scc_cnt]++;
        } while (y != u);
    }
}

int main()
{
    scanf("%d%d", &n, &m);
    int a, b, c;
    for (int i = 1; i <= m; i++)
    {
        scanf("%d%d", &a, &b);
        add(a, b);
    }
    for (int i = 1; i <= n; i++)
    {
        if (!dfn[i])
        {
            tarjan(i);
        }
    }

    for (int i = 1; i <= n; i++)
    {
        a = id[i];
        for (int j = h[i]; j; j = cun[j].nt)
        {
            b = id[cun[j].data];
            if(a!=b){
```

```
                dout[a]++;
            }
        }
    }
    int zero=0,sum=0;
    for(int i=1;i<=scc_cnt;i++){
        if(!dout[i]){
            zero++;
            sum+=Size[i];
            if(zero>1){
                sum=0;
                break;
            }
        }
    }

    printf("%d",sum);

    return 0;
}
```

```
#include <bits/stdc++.h>
using namespace std;
const int N = 2e6 + 5;
const int M = 2e6 + 5;
struct Node
{
    int data;
    int nt;
};
int n, m;
int h[N], idx = 1;
Node cun[M];
int dfn[N], low[N], timestamp;
stack<int> stk;
bool in_stk[N];
int id[N], scc_cnt, Size[N];
int dout[N];
int din[N];
Node cun2[M];
int h2[N], idx2 = 1;
int mx[N];
typedef pair<int, int> pii;
set<pii> mvis;
void add(int a, int b)
{
    cun[idx] = {b, h[a]};
    h[a] = idx++;
}
void add2(int a, int b)
{
    cun2[idx2] = {b, h2[a]};
    h2[a] = idx2++;
}

void tarjan(int u)
{
    dfn[u] = low[u] = ++timestamp;
    in_stk[u] = 1;
    stk.push(u);
    for (int i = h[u]; i; i = cun[i].nt)
    {
        if (!dfn[cun[i].data])
        {
            tarjan(cun[i].data);
            low[u] = min(low[u], low[cun[i].data]);
        }
        else if (in_stk[cun[i].data])
        {
            low[u] = min(low[u], dfn[cun[i].data]);
        }
    }
```

```
    if (dfn[u] == low[u])
    {
        ++scc_cnt;
        int y;
        do
        {
            y = stk.top();
            stk.pop();
            in_stk[y] = 0;
            id[y] = scc_cnt;
            Size[scc_cnt]++;
        } while (y != u);
    }
}

int main()
{
    scanf("%d%d", &n, &m);
    int a, b, c;
    for (int i = 1; i <= m; i++)
    {
        scanf("%d%d", &a, &b);
        add(a, b);
    }
    for (int i = 1; i <= n; i++)
    {
        if (!dfn[i])
        {
            tarjan(i);
        }
    }

    for (int i = 1; i <= n; i++)
    {
        a = id[i];
        for (int j = h[i]; j; j = cun[j].nt)
        {
            b = id[cun[j].data];
            if (a != b)
            {
                // dout[a]++;

                if (mvis.find(pii(a, b)) == mvis.end())
                {
                    mvis.insert(pii(a, b));
                    add2(a, b);
                    din[b]++;
                }
            }
        }
    }

  //  for (int i = 1; i <= n; i++)
  //  {
   //     cout << i << " " << id[i] << endl;
  //  }

    queue<int> q;
    int zero = 0, sum = 0;
    for (int i = 1; i <= scc_cnt; i++)
    {
        // cout<< Size[i]<<endl;
        if (!din[i])
        {
          //  cout << i << endl;
            mx[i] = Size[i];
            q.push(i);
        }
    }

    int mxi = 0;
    while (!q.empty())
    {
        int tp = q.front();
      //  cout<<tp<<" "<<mx[tp]<<endl;
```

```
        q.pop();
        mxi = max(mxi, mx[tp]);
        for (int i = h2[tp]; i; i = cun2[i].nt)
        {
            din[cun2[i].data]--;
            mx[cun2[i].data] = max(mx[cun2[i].data], mx[tp] + Size[cun2[i].data]);
            if (!din[cun2[i].data])
            {
                q.push(cun2[i].data);
            }
        }
    }

    printf("%d", mxi);

    return 0;
}
```

网络流

最大流之Dinic算法

```
inline void add(int a,int b,int c){
    e[idx]=b,f[idx]=c,ne[idx]=h[a],h[a]=idx++;
    e[idx]=a,f[idx]=0,ne[idx]=h[b],h[b]=idx++;
}

int dfs(int id,int lim){
    if(id==T) return lim;
    int flow=0;
    for(int i=arc[id];~i&&flow<lim;i=ne[i]){//优化之一
        int ver=e[i];
        arc[id]=i;//优化之二
        if(d[ver]==d[id]+1&&f[i]){
            int t=dfs(ver,min(f[i],lim-flow));
            if(!t) d[ver]=-1;//优化之三
            f[i]-=t,f[i^1]+=t,flow+=t;
        }
    }
    return flow;
}

bool bfs(){
    memset(d,-1,sizeof d);
    q[0]=S,arc[S]=h[S],d[S]=0;
    int hh=0,tt=1;
    while(hh<tt){
        int ver=q[hh++];
        for(int i=h[ver];~i;i=ne[i]){
            int t=e[i];
            if(d[t]==-1&&f[i]){
                d[t]=d[ver]+1;
                arc[t]=h[t];
                if(t==T) return 1;
                q[tt++]=t;
            }
        }
    }
    return 0;
}

int dinic(){
    int F=0,flow=0;
    while(bfs()) while(flow=dfs(S,INF)) F+=flow;
    return F;
}
```