ICPC 代码模板-V2.0 图论

SDU(QD)　sdu1801

2019 年 11 月 14 日

目录

1 图论

1.1 2-SAT

对于 a_i，用第 $2i-1$ 和 $2i$ 表示取 0 和 1，一条边 $x->y$ 表示选了 x 必选 y i' 表示 i 的反面：

1. i,j 不能同时选：$i \to j', j \to i'$，一般为 $a_i xor a_j = 1$
2. i,j 必须同时选：$i \to j, j \to i$，一般为 $a_i xor a_j = 0$
3. i,j 只能选其一：$i \to j', j \to i', i' \to j, j' \to i$ 一般为 $a_i or a_j = 1$
4. 必须选 i：$i' \to i$ 一般为 $a_i = 1$ 或 $a_i and a_j = 1$

1.2 生成树计数

G 的度数矩阵 $D[G]$,G 的邻接矩阵 $A[G]$ G 的 $Kirchhoff$ 矩阵 $C[G] = D[G]-A[G]$ $Matrix-Tree$ 定理 $C[G]$ 的任何一个 $n-1$ 阶主子式的行列式的绝对值，为所有不同生成树的个数求行列式用高斯消元

```
1
2  LL a[MAXN][MAXN];
3  const LL mod=1e9;
4  void add(int u,int v){//加边 双向
5      a[u][u]++,a[v][v]++;
6      a[u][v]--,a[v][u]--;
7  }
8  int Gauss(int n){//高斯消元
9      int ans=1;
10     for(int i=1;i<=n;i++){
11         for(int k=i+1;k<=n;k++){
12             while(a[k][i]){
13                 int d=a[i][i]/a[k][i];
14                 for(int j=i;j<=n;j++)a[i][j]=(a[i][j]-1ll*d*a[k][j]%mod+mod)%mod;
15                 swap(a[i],a[k]),ans=-ans;
16             }
17         }
18         ans=1ll*ans*a[i][i]%mod,ans=(ans%mod+mod)%mod;
19     }
20     return ans;
21 }
22 int n,m,id[MAXN][MAXN];
23 char s[MAXN][MAXN];
24 int main(){
25     scanf("%d%d",&n,&m);
26     for(int i=1;i<=n;i++)scanf("%s",s[i]+1);
27     int idx=0;
28     for(int i=1;i<=n;i++)
29         for(int j=1;j<=m;j++)
30             if(s[i][j]=='.')id[i][j]=++idx;
31     for(int i=1;i<=n;i++)
32         for(int j=1;j<=m;j++)
33             if(id[i][j]){
34                 if(id[i-1][j])add(id[i][j],id[i-1][j]);
35                 if(id[i][j-1])add(id[i][j],id[i][j-1]);
36             }
37     cout<<Gauss(idx-1);
38     return 0;
39 }
```

1.3 Floyd 求最小环

```
#include<algorithm>
#include<iostream>
#include<cstdio>
#define INF 0x3f3f
#define MAXN 511
using namespace std;
int dis[MAXN][MAXN], mp[MAXN][MAXN], n, m;
int mincost_route(){
    int mincost = INF;
    for(int k = 1; k <= n; k ++){
        for(int i = 1; i < k; i ++)
            for(int j = i+1; j < k; j ++)
                mincost = min(mincost, dis[i][j] + mp[i][k] + mp[k][j]);
        for(int i = 1; i <= n; i ++)
            for(int j = 1; j <= n; j ++)
                dis[i][j] = min(dis[i][j], dis[i][k] + dis[k][j]);
    }
    return mincost;
}
int main(){
    scanf("%d%d", &n, &m);
    for(int i = 1; i <= n; i ++)
        for(int j = 1; j <= n; j ++)
            mp[i][j] = dis[i][j] = INF;
    for(int u, v, w, i = 1; i <= m; i ++){
        scanf("%d%d%d",&u, &v, &w);
        mp[u][v] = mp[v][u] = dis[u][v] = dis[v][u] = min(w, mp[u][v]);
    }
    cout << mincost_route() << endl;
}
```

1.4 BFS 判负环

```

const int maxn=200010;
struct gra {
    bool vis[maxn];
    int head[maxn], to[maxn << 1], nxt[maxn << 1], cnt;
    int dis[maxn],c[maxn];
    int f[maxn << 1],N;
    void clear(int n) {N=n,fill(head, head+5+n, 0),cnt = 0;}
    void add(int a, int b, int c) {nxt[++cnt] = head[a], to[head[a] = cnt] = b, f[cnt]
        = c;}
    bool spfa(int i){
        queue<int>q;
        q.push(i),c[i]++;
        dis[i]=0;
        while(!q.empty()){
            i=q.front(),q.pop();
            vis[i]=0;
            for(int j=head[i];j;j=nxt[j]){
                if(dis[to[j]]>dis[i]+f[j]){
```

```
19                      dis[to[j]]=dis[i]+f[j];
20                      if(!vis[to[j]]){
21                          c[to[j]]++;
22                          if(c[to[j]]>N)return 1;
23                          vis[to[j]]=1,q.push(to[j]);
24                      }
25                  }
26              }
27          }
28          return 0;
29      }
30      bool ne_c(int n,int i){
31          fill(dis,dis+5+n,1000000000);
32          fill(vis,vis+5+n,0);
33          fill(c,c+5+n,0);
34          return spfa(i);
35      }
36  }G;
37  int n,m1,m2;
38  int edge1[maxn][4],edge2[maxn][4];
39  bool check(int p){
40      G.clear(n+1);
41      for(int i=1;i<=m1;i++)
42          G.add(edge1[i][2],edge1[i][1]-1,-edge1[i][3]);
43      for(int i=1;i<=m2;i++)
44          G.add(edge2[i][1]-1,edge2[i][2],p-edge2[i][3]);
45      for(int i=1;i<=n;i++)G.add(i,i-1,0),G.add(i-1,i,1);
46      G.add(0,n,p),G.add(n,0,-p);
47      return G.ne_c(n,0);
48  }
49  int main(){
50      int T;cin>>T;
51      while(T--){
52          scanf("%d%d%d",&n,&m1,&m2);
53          for(int i=1;i<=m1;i++)
54              scanf("%d%d%d",&edge1[i][1],&edge1[i][2],&edge1[i][3]);
55          for(int i=1;i<=m2;i++)
56              scanf("%d%d%d",&edge2[i][1],&edge2[i][2],&edge2[i][3]);
57          int l=0,r=n,mid;
58          while(l<r){
59              mid=(l+r)>>1;
60              if(check(mid)) l=mid+1;
61              else r=mid;
62          }
63          printf("%d\n",l);
64      }
65      return 0;
66  }
```

1.5 DFS 判负环

```
1
2
3  const int maxn = 200010;
```

```
4
5  struct gra {
6      bool ins[maxn];
7      int head[maxn], to[maxn << 1], nxt[maxn << 1], cnt, flag;
8      int dis[maxn], f[maxn << 1];
9      void clear(int n) { fill(head, head + 5 + n, 0), flag = cnt = 0; }
10     void add(int a, int b, int c) { nxt[++cnt] = head[a], to[head[a] = cnt] = b, f[cnt
           ] = c; }
11     void spfa(int h) {
12         ins[h] = 1;
13         for (int i = head[h]; i; i = nxt[i])
14             if (dis[to[i]] > dis[h] + f[i]) {
15                 if (ins[to[i]] || flag) {
16                     flag = 1;
17                     break;
18                 }
19                 dis[to[i]] = dis[h] + f[i];
20                 spfa(to[i]);
21             }
22         ins[h] = 0;
23     }
24
25     bool ne_c(int n) {
26         fill(dis, dis + 5 + n, 0);
27         fill(ins, ins + 5 + n, 0);
28         flag = 0;
29         for (int i = 0; i <= n && !flag; i++)spfa(i);
30         return flag;
31     }
32 } G;
33
34 int n, m1, m2;
35 int edge1[maxn][4], edge2[maxn][4];
36
37 bool check(int p) {
38     G.clear(n + 1);
39     for (int i = 1; i <= m1; i++)
40         G.add(edge1[i][2], edge1[i][1] - 1, -edge1[i][3]);
41     for (int i = 1; i <= m2; i++)
42         G.add(edge2[i][1] - 1, edge2[i][2], p - edge2[i][3]);
43     for (int i = 1; i <= n; i++)G.add(i, i - 1, 0), G.add(i - 1, i, 1);
44     G.add(0, n, p), G.add(n, 0, -p);
45     return G.ne_c(n);
46 }
47
48 int main() {
49     int T;
50     cin >> T;
51     while (T--) {
52         scanf("%d%d%d", &n, &m1, &m2);
53         for (int i = 1; i <= m1; i++)
54             scanf("%d%d%d", &edge1[i][1], &edge1[i][2], &edge1[i][3]);
55         for (int i = 1; i <= m2; i++)
56             scanf("%d%d%d", &edge2[i][1], &edge2[i][2], &edge2[i][3]);
```

```
57          int l = 0, r = n, mid;
58          while (l < r) {
59              mid = (l + r) >> 1;
60              if (check(mid)) l = mid + 1;
61              else r = mid;
62          }
63          printf("%d\n", l);
64      }
65      return 0;
66  }
```

1.6 匈牙利算法

```
1   //匈牙利求最小边覆盖（点数 − 最大匹配）
2   //最小点覆盖 = 最大匹配
3   #include<iostream>
4   #include<cstdio>
5   #include<cstring>
6
7   using namespace std;
8   const int N = 100010;
9   int matching[N], te, check[N], ans, head[N], n, m;
10  struct edge {
11      int v, next;
12  } e[200010];
13
14  void add(int u, int v) {
15      e[++te].v = v;
16      e[te].next = head[u];
17      head[u] = te;
18  }
19
20  int dfs(int u) {
21      for (int i = head[u]; i; i = e[i].next) {
22          int v = e[i].v;
23          if (!check[v]) {
24              check[v] = 1;
25              if (matching[v] == -1 || dfs(matching[v])) {
26                  matching[u] = v;
27                  matching[v] = u;
28                  return true;
29              }
30          }
31      }
32      return false;
33  }
34
35  int hungarian() {
36      memset(matching, -1, sizeof(matching));
37      for (int i = 1; i <= n; i++) {
38          if (matching[i] == -1) {
39              memset(check, 0, sizeof(check));
40              if (dfs(i))++ans;
41          }
```

```
    }
}

int main() {
    ans = 0;
    memset(head, 0, sizeof(head));
    cin >> n >> m;
    for (int i = 1; i <= m; ++i) {
        int u, v;
        scanf("%d%d", &u, &v);
        add(u, v + n);
    }
    hungarian();
    cout << n - ans;
}
```

(1) 二分图的最大匹配

匈牙利算法

(2) 二分图的最小点覆盖

二分图的最小点覆盖 = 二分图的最大匹配

求最小点覆盖：从右边所有没有匹配过的点出发，按照增广路的“交替出现”的要求 DFS。最终右边没有访问过的点和左边访问过的点组成最小点覆盖。

(3) 二分图的最少边覆盖

二分图的最少边覆盖 = 点数-二分图的最大匹配

证明：

先贪心选一组最大匹配的边放进集合，对于剩下的没有匹配的点，随便选一条与之关联的边放进集合，那么得到的集合就是最小边覆盖。所以有：最小边覆盖 = 最大匹配 + 点数-2* 最大匹配 = 点数-最大匹配

(4) 二分图的最大独立集

二分图的最大独立集 = 点数-二分图的最大匹配

证明：

我们可以这样想，先把所有的点放进集合，然后删去最少的点和与之相关联的边，使得全部边都被删完，这就是最小点覆盖。所以有：最大独立集 = 点数-最小点覆盖

(5) 有向无环图的最少不相交路径覆盖

我们把原图中的点 V 拆成两个点 Vx 和 Vy，对于原图中的边 A>B，我们在新图中连 Ax>By。那么最少不相交路径覆盖 = 原图的点数-新图的最大匹配

证明：

一开始每个点都独立为一条路径，在二分图中连边就是将路径合并，每连一条边路径数就减一。因为路径不能相交，所以不能有公共点，这恰好就是匹配的定义。所以有：最少不相交路径覆盖 = 原图的点数-新图的最大匹配

(6) 有向无环图的最少可相交路径覆盖

先用 floyd 求出原图的传递闭包，如果 a 到 b 有路，那么就加边 a->b。然后就转化成了最少不相交路径覆盖问题。

(7) 有向无环图中最少不相交路径覆盖和最大独立集的相互转化

用偏序集，一般可以抽象为有向无环图。

Dilworth 定理：有向无环图的最大独立集 = 有向无环图最少不相交路径覆盖

1.7 费用流-EK(Dij-单路增广)

```
typedef int T;

struct gra{
    int head[maxn], to[maxn<<1], nxt[maxn<<1], cnt;
    T f[maxn<<1], v[maxn << 1];
    void clear(int n) {fill(head, head+1+n, 0), cnt = 1;}
    void add(int a, int b, T c, T d){
        nxt[++ cnt] = head[a], head[a] = cnt, to[cnt] = b, f[cnt] = c, v[cnt] = d;
```

```
9           nxt[++cnt] = head[b], head[b] = cnt, to[cnt] = a, f[cnt] = 0, v[cnt] = -d;
10      }
11  };
12
13  struct cost_flow : public gra{
14      int pr[maxn], q[maxn], s, t, mx;
15      bool vis[maxn];
16      T dis[maxn], mnf[maxn];
17      void init(int ss, int tt, int mxx){s = ss, t = tt, mx = mxx, clear(mx);}
18      bool spfa(){
19          for(int i = 0; i <= mx; i ++) dis[i] = inf, vis[i] = 0;
20          int l = 1, r = 0;
21          mnf[s] = inf, vis[s] = 1, dis[s] = 0, q[++r] = s;
22          while(l <= r){
23              int x = q[l ++];
24              for(int i = head[x]; i; i = nxt[i]){
25                  int u = to[i];
26                  if(f[i] > 0 && dis[u] > dis[x] + v[i]){
27                      dis[u] = dis[x] + v[i];
28                      pr[u] = i, mnf[u] = min(mnf[x], f[i]);
29                      if(vis[u] == 0) vis[u] = 1, q[++r] = u;
30                  }
31              }
32              vis[x] = 0;
33          }
34          return dis[t] != inf;
35      }
36      pair<T, T> ek(){
37          T flow = 0, cost = 0;
38          while(spfa()){
39              cost += mnf[t]*dis[t], flow += mnf[t];
40              for(int x = t; x != s; x = to[pr[x] ^ 1]){
41                  f[pr[x]] -= mnf[t], f[pr[x]^1] += mnf[t];
42              }
43          }
44          return make_pair(flow, cost);
45      }
46  };
```

1.8 最大流-Dinic

```
1   typedef int T;
2
3   struct gra{
4       int head[maxn], to[maxn<<1], nxt[maxn<<1], cnt;
5       T f[maxn<<1];
6       void clear(int n) {fill(head, head+1+n, 0), cnt = 1;}
7       void add(int a, int b, T c){
8           nxt[++cnt] = head[a], head[a] = cnt, to[cnt] = b, f[cnt] = c;
9           nxt[++cnt] = head[b], head[b] = cnt, to[cnt] = a, f[cnt] = 0;
10      }
11  };
12
13  struct flow : public gra{
```

```
    int dep[maxn], q[maxn], cur[maxn], s, t, mx;
    void init(int ss, int tt, int mxx){s = ss, t = tt, mx = mxx, clear(mx);}
    bool bfs(){
        for(int i = 0; i <= mx; i ++) dep[i] = -1;
        int l = 1, r = 0;
        q[++ r] = s, dep[s] = 1;
        while(l <= r){
            int x = q[l ++];
            for(int i = head[x]; i; i = nxt[i]){
                int u = to[i];
                if(dep[u] == -1 && f[i] > 0)
                    q[++ r] = u, dep[u] = dep[x] + 1;
            }
        }
        return dep[t] != -1;
    }
    T dfs(int x, T mr){
        if(x == t || mr == 0) return mr;
        T c = 0, res = 0;
        for(int &i = cur[x]; i; i = nxt[i]){
            int u = to[i];
            if(dep[u] == dep[x] + 1 && (c = dfs(u, min(mr, f[i]))))
                f[i] -= c, f[i^1] += c, mr -= c, res += c;
            if(mr == 0) break;
        }
        return res;
    }
    T dinic(){
        T res = 0;
        while(bfs()){
            for(int i = 0; i <= mx; i ++) cur[i] = head[i];
            res += dfs(s, inf);
        }
        return res;
    }
};
```

1.9 最大流-Dinic

```
typedef int T;

struct gra{
    int head[maxn], to[maxn<<1], nxt[maxn<<1], cnt;
    T f[maxn<<1];
    void clear(int n) {fill(head, head+1+n, 0), cnt = 1;}
    void add(int a, int b, T c){
        nxt[++ cnt] = head[a], head[a] = cnt, to[cnt] = b, f[cnt] = c;
        nxt[++ cnt] = head[b], head[b] = cnt, to[cnt] = a, f[cnt] = 0;
    }
};

struct flow : public gra{
    int dep[maxn], q[maxn], cur[maxn], s, t, mx;
    void init(int ss, int tt, int mxx){s = ss, t = tt, mx = mxx, clear(mx);}
```

```
16      bool bfs(){
17          for(int i = 0; i <= mx; i ++) dep[i] = -1;
18          int l = 1, r = 0;
19          q[++ r] = s, dep[s] = 1;
20          while(l <= r){
21              int x = q[l ++];
22              for(int i = head[x]; i; i = nxt[i]){
23                  int u = to[i];
24                  if(dep[u] == -1 && f[i] > 0)
25                      q[++ r] = u, dep[u] = dep[x] + 1;
26              }
27          }
28          return dep[t] != -1;
29      }
30      T dfs(int x, T mr){
31          if(x == t || mr == 0) return mr;
32          T c = 0, res = 0;
33          for(int &i = cur[x]; i; i = nxt[i]){
34              int u = to[i];
35              if(dep[u] == dep[x] + 1 && (c = dfs(u, min(mr, f[i]))))
36                  f[i] -= c, f[i^1] += c, mr -= c, res += c;
37              if(mr == 0) break;
38          }
39          return res;
40      }
41      T dinic(){
42          T res = 0;
43          while(bfs()){
44              for(int i = 0; i <= mx; i ++) cur[i] = head[i];
45              res += dfs(s, inf);
46          }
47          return res;
48      }
49  };
```

1.10 最大流-ISAP

```
1   typedef int T;
2
3   struct gra{
4       int head[maxn], to[maxn<<1], nxt[maxn<<1], cnt;
5       T f[maxn<<1];
6       void clear(int n) {fill(head, head+1+n, 0), cnt = 1;}
7       void add(int a, int b, T c){
8           nxt[++ cnt] = head[a], head[a] = cnt, to[cnt] = b, f[cnt] = c;
9           nxt[++ cnt] = head[b], head[b] = cnt, to[cnt] = a, f[cnt] = 0;
10      }
11  };
12
13  struct flow : public gra{
14      int dep[maxn], vm[maxn], q[maxn], cur[maxn], s, t, mx;
15      void init(int ss, int tt, int mxx){s = ss, t = tt, mx = mxx, clear(mx);}
16      void bfs(){
17          for(int i = 0; i <= mx; i ++) dep[i] = -1, vm[i] = 0;
```

```
18          int l = 1, r = 0;
19          q[++ r] = t, dep[t] = 1, vm[1] = 1;
20          while(l <= r){
21              int x = q[l ++];
22              for(int i = head[x]; i; i = nxt[i]){
23                  int u = to[i];
24                  if(dep[u] == -1)
25                      dep[u] = dep[x] + 1, vm[dep[u]] ++, q[++ r] = u;
26              }
27          }
28      }
29      T dfs(int x, T mr){
30          if(x == t || mr == 0) return mr;
31          T c = 0, res = 0;
32          for(int &i = cur[x]; i; i = nxt[i]){
33              int u = to[i];
34              if(f[i] && dep[u] + 1 == dep[x] && (c = dfs(u, min(f[i], mr))))
35                  f[i] -= c, f[i^1] += c, mr -= c, res += c;
36              if(mr == 0) return res;
37          }
38          if(-- vm[dep[x]] == 0) dep[s] = mx + 1;
39          dep[x] ++, vm[dep[x]] ++;
40          return res;
41      }
42      T isap(){
43          T res = 0; bfs();
44          while(dep[s] <= mx) {
45              for(int i = 0; i <= mx; i ++) cur[i] = head[i];
46              res += dfs(s, inf);
47          }
48          return res;
49      }
50  };
```

1.11 最大流-ISAP(非封装版)

```
1   //注意：te初值为1， mx为总点数， head记得清空
2   #include<cstdio>
3   #include<iostream>
4   #include<cstring>
5   #include<algorithm>
6   #include<string>
7
8   using namespace std;
9
10  const int N = 510, M = 100010;
11  const int inf = 0x3f3f3f3f;
12  int te;
13  int head[N];
14  struct edge{
15      int u, v, f, nxt;
16  }e[M];
17  inline void add(int u, int v, int cap){
18      e[++te] = (edge){u, v, cap, head[u]};
```

```
19      head[u] = te;
20  }
21  int dep[N], vm[N], q[N], cur[N], s, t, mx;
22  void bfs(){
23      for (int i = 0; i <= mx; ++i) dep[i] = -1, vm[i] = 0;
24      int l = 1, r = 0;
25      q[++r] = t, dep[t] = 1, vm[1] = 1;
26      while(l <= r){
27          int x = q[l++];
28          for (int i = head[x]; i; i = e[i].nxt){
29              int v = e[i].v;
30              if (dep[v] == -1)
31                  dep[v] = dep[x] + 1, vm[dep[v]]++, q[++r] = v;
32          }
33      }
34  }
35  inline int dfs(int x, int mr){
36      if (x == t || mr == 0) return mr;
37      int c = 0, res = 0;
38      for (int &i = cur[x]; i; i = e[i].nxt){
39          int v = e[i].v;
40          if (e[i].f && dep[v] + 1 == dep[x] && (c = dfs(v, min(e[i].f, mr))))
41              e[i].f -= c, e[i ^ 1].f += c, mr -= c, res += c;
42          if (mr == 0) return res;
43      }
44      if (-- vm[dep[x]] == 0) dep[s] = mx + 1;
45      dep[x]++, vm[dep[x]]++;
46      return res;
47  }
48  int res;
49  void isap(){
50      bfs();
51      while(dep[s] <= mx){
52          for (int i = 0; i <= mx; ++i) cur[i] = head[i];
53          res += dfs(s, inf);
54          // cout<<11111<<' '; cout<<res<<endl;
55      }
56  }
57  int n;
```

1.12 tarjan

1.12.1 求割点割边

```
1  // 数函数调用init与work
2  struct Tarjan : public gra{
3      int dfn[maxn], low[maxn], st[maxn], scc[maxn], sz[maxn];
4      int tp, tim, num, n;
5      gra e;
6      void init(int N){clear(n = N), tim = 0, num = 0, tp = 0, mem(dfn, 0), mem(scc, 0)
           ;}
7      int dfs(int x){
8          dfn[x] = low[x] = ++ tim, st[++tp] = x;
9          int stx = tp;
```

```
        for(int i = head[x]; i; i = nxt[i]){
            int u = to[i];
            if(!dfn[u]) dfs(u), low[x] = min(low[x], low[u]);
            else if(!scc[u]) low[x] = min(low[x], dfn[u]);
        }
        if(dfn[x] == low[x]){
            sz[++ num] = tp - stx + 1;
            while(tp >= stx) scc[st[tp --]] = num;
        }
    }
    void build(){
        unordered_set <int> has[n+1];
        for(int x = 1; x <= n; x ++){
            for(int i = head[x]; i; i = nxt[i]){
                int u = to[i], sx = scc[x], su = scc[u];
                if(has[sx].find(su) == has[sx].end() && sx != su)
                    e.add(sx, su), has[sx].insert(su);
            }
        }
    }
    void work(){
        for(int i = 1; i <= n; i ++) if(!dfn[i]) dfs(i);
        build();
        // 这里写操作
    }
};
```

1.12.2 边双联通分量

```
const int maxn = 1e4 + 10;
int n, m;
struct data {
    int to, next;
} e[maxn * 10];
int head[maxn], cnt = 0;
int cur = 0, top = 0;
int low[maxn], dfn[maxn], vis[maxn], ans[maxn * 10];

void ins(int u, int v) {
    e[++cnt].to = v;
    e[cnt].next = head[u];
    head[u] = cnt;
}

void tarjan(int u, int fa) {
    vis[u] = 1;
    dfn[u] = low[u] = ++cur;
    for (int i = head[u]; i; i = e[i].next) {
        int x = e[i].to;
        if (vis[x] == 1 && x != fa) {
            if (low[u] > dfn[x])low[u] = dfn[x];
            ans[++top] = u;
            ans[++top] = x;
        }
```

```
26          if (vis[x] == 0) {
27              tarjan(x, u);
28              if (low[x] < low[u])low[u] = low[x];
29              if (low[x] > dfn[u]) {
30                  ans[++top] = u;
31                  ans[++top] = x;
32                  ans[++top] = x;
33                  ans[++top] = u;
34              } else {
35                  ans[++top] = u;
36                  ans[++top] = x;
37              }
38          }
39      }
40      vis[u] = 2;
41  }
42  //tarjan(i ,-1);
```

1.12.3 点双联通分量

```
1   const int maxn = 5010;
2   int n, m, q, head[maxn];
3   struct data {
4       int to, next;
5   } e[maxn * 4];
6   int cnt = 0, top = 0, tot = 0;
7   int dfn[maxn], low[maxn], s[maxn], vis[maxn];
8   vector<int> color[maxn];
9   int block[maxn];
10  int block_color = 0;
11
12  void ins(int u, int v) {
13      e[++cnt].to = v;
14      e[cnt].next = head[u];
15      head[u] = cnt;
16  }
17
18  void bic(int v, int fa, int dep) {
19      vis[v] = 1;
20      s[++top] = v;
21      block[v] = block_color;
22      dfn[v] = low[v] = dep;
23      for (int i = head[v]; i; i = e[i].next) {
24          int x = e[i].to;
25          if (vis[x] == 0) {
26              bic(x, v, dep + 1);
27              if (low[x] >= dfn[v]) {
28                  tot++;
29                  do {
30                      color[s[top]].push_back(tot);
31                      top--;
32                  } while (s[top + 1] != x);
33                  if (low[x] == dfn[v])color[v].push_back(tot);
34              }
```

```
35              if (low[x] <= low[v])low[v] = low[x];
36          } else if (x != fa && vis[x] == 1)low[v] = min(low[v], dfn[x]);
37      }
38      vis[v] = 2;
39  }
40
41  //bic(x, -1, 0);
```

1.12.4 极大强连通分量

```
1   const int maxn = 1e4 + 10;
2
3   int n, m;
4   int cur = 0;
5   vector<int> v[maxn];
6   bool vis[maxn];
7   int low[maxn], dfn[maxn];
8   int s[maxn];
9   int top = 0, tot = 0;
10
11  void tarjan(int x) {
12      s[++top] = x;
13      vis[x] = 1;
14      dfn[x] = low[x] = ++cur;
15      for (int i = 0; i < v[x].size(); ++i) {
16          int u = v[x][i];
17          if (dfn[u] == 0)tarjan(u);
18          if (vis[u] && low[x] > low[u])low[x] = low[u];
19      }
20      if (low[x] == dfn[x]) {
21          tot = 0;
22          while (s[top + 1] != x) {
23              top--;
24              tot++;
25          }
26      }
27  }
```