ACM ICPC Template

GuessEver

Last Modify at January 29, 2015

Contents

1 Graph

1.1 Shortest path

1.1.1 Dijkstra

```
void dijkstra()
{
    memset(dist, 0, sizeof(dist));
    while(!Q.empty())
    {
        int x = Q.top().second; Q.pop();
        if(done[x]) continue;
        done[x] = 1;
        for(Link p = head[x]; p; p = p->next)
            if(dist[p->y] > dist[x] + p->z)
            {
                dist[p->y] = dist[x] + p->z;
                Q.push(make_pair(dist[p->y], p->y));
            }
    }
}
```

1.1.2 Spfa

```
void spfa()
{
    memset(dist, 0x3f, sizeof(dist));
    Q.push(S);//S为源点
    while(!Q.empty())
    {
        int x = Q.front();
        Q.pop(); inQ[x] = 0;
        for(Link p = head[x]; p; p = p->next)
            if(dist[p->y] > dist[x] + p->z)
            {
                dist[p->y] = dist[x] + p->z;
                if(!inQ[p->y])
                {
                    Q.push(p->y);
                    inQ[p->y] = 1;
                }
            }
    }
}
```

1.1.3 Floyd

```
void floyd()
{
    for(int k = 1; k <= n; k++) // 这里可以看作是一个加边的过程
```

```
        for(int i = 1; i <= n; i++)
            for(int j = 1; j <= n; j++)
                map[i][j] = min(map[i][j], map[i][k] + map[k][j]);
}

// 最小环
void MinCircle()
{
    cap[] = map[];
    int circle = 0x3f3f3f3f;
    for(int k = 1; k <= n; k++)
    {
        for(int i = 1; i < k; i++)
            for(int j = i+1; j < k; j++)
                circle = min(circle, map[i][j] + cap[j][k]+cap[k][i]);
        for(int i = 1; i <= n; i++)
            for(int j = 1; j <= n; j++)
                map[i][j] = min(map[i][j], map[i][k] + map[k][j]);
    }
    return circle == 0x3f3f3f3f ? -1 : circle;
}

// floyd判圈法 ( 大白书 p44 )
void Circle()
{
    int ans = k;
    int k1 = k, k2 = k;
    do{
        k1 = next(k1);
        k2 = next(k2); ans = max(ans, k2);
        k2 = next(k2); ans = max(ans, k2);
    }while(k1 != k2);
    return ans;
}
```

1.2 Minimum Spanning Tree

1.2.1 Prime

```
void prime()
{
    memset(dist, 0, sizeof(dist));
    int res = 0;
    while(!Q.empty())
    {
        int x = Q.top().second;
        if(done[x]) {Q.pop(); continue;}
        res += Q.top().first;
        Q.pop();
        for(Link p = head[x]; p; p = p->next)
            if(dist[p->y] > p->z)
            {
```

```
                dist[p->y] = p->z;
                Q.push(make_pair(dist[p->y], p->y));
            }
    }
}
```

1.2.2 Kruskal

```
void prime()
{
    sort(edge, edge+Cnt, cmp);
    int res = 0;
    for(int i = 0; i < Cnt; i++)
    {
        if(getroot(edge[i].x) == getroot(edge[i].y)) continue;
        merge(edge[i].x, edge[i].y);
        res += edge[i].z;
    }
}
```

1.3 Tarjan - Strong Union

```
void dfs(int x)
{
    now[x] = low[x] = ++dfstime;
    hash[x] = 1;
    st.push(x); inst[x] = 1;
    for(int i = 1; i <= n; i++)
        if(map[x][i])
        {
            if(!hash[i])
            {
                dfs(i);
                low[x] = min(low[x], low[i]);
            }
            else if(inst[i]) low[x] = min(low[x], now[i]);
        }
    if(low[x] == now[x])
    {
        while(!st.empty())
        {
            int u = st.top();
            st.pop(); inst[u] = 0;
            belong[u] = number;
            if(u == x) break;
        }
        numer++;
    }
}
void tarjan()
{
```

```
30      for(int i = 1; i <= n; i++)
31          if(!hash[i]) dfs(i);
32      if(!st.empty()) // 这是一个未知bug   栈中还会剩下一个强连通分量
33      {
34          while!st.empty())
35          {
36              int u = st.top();
37              st.pop();
38              belong[u] = number;
39          }
40          number++;
41      }
42  }
```

1.4 LCA

1.4.1 Tarjan

```
1   // poj 1330 (changed something)
2   // LCA tarjan
3   #include <cstdio>
4   #include <cstring>
5
6   const int N = 10000 + 10;
7
8   int n;
9   struct Link{int y, idx; Link *next;}*head[N], *ask[N];
10  int tx, ty;
11  bool in[N], vis[N];
12  int f[N];
13  int ans[N]; // Query Answer
14
15  void inLink(int x, int y)
16  {
17      Link *p = new Link;
18      p -> y = y;
19      p -> next = head[x];
20      head[x] = p;
21  }
22  void inAsk(int x, int y, int idx)
23  {
24      Link *p = new Link;
25      p -> y = y;
26      p -> idx = idx;
27      p -> next = ask[x];
28      ask[x] = p;
29  }
30
31  int getroot(int x)
32  {
33      return f[x] == x ? x : f[x] = getroot(f[x]);
34  }
```

```
35
36 void LCA(int x)
37 {
38     vis[x] = 1;
39     f[x] = x;
40     for(Link *p = ask[x]; p; p = p -> next)
41         if(vis[p->y]) ans[p->idx] = getroot(p->y);
42     for(Link *p = head[x]; p; p = p -> next)
43         if(!vis[p->y])
44         {
45             LCA(p->y);
46             f[p->y] = x;
47         }
48 }
49
50 int main()
51 {
52     int T; scanf("%d", &T);
53     while(T--)
54     {
55         memset(head, 0, sizeof(head));
56         memset(ask, 0, sizeof(ask));
57         memset(in, 0, sizeof(in));
58         memset(vis, 0, sizeof(vis));
59         scanf("%d", &n);
60         for(int i = 1; i <= n; i++) f[i] = i;
61         for(int i = 1; i < n; i++)
62         {
63             int x, y;
64             scanf("%d%d", &x, &y);
65             inLink(x, y);
66             in[y] = 1;
67         }
68         int q = 1;// the number of query
69         for(int i = 1; i <= q; i++)
70         {
71             int x, y; scanf("%d%d", &x, &y);
72             inAsk(x, y, i); inAsk(y, x, i);
73         }
74         int root = -1;
75         for(int i = 1; i <= n; i++)
76             if(!in[i]) {root = i; break;}
77         LCA(root);
78         for(int i = 1; i <= q; i++)
79             printf("%d\n", ans[i]);
80     }
81     return 0;
82 }
```

1.4.2 Doubling Algorithm

还不会...

1.5 Bipartite Graph

1.5.1 Maximal Matching - The Hungarian algorithm

```
int ttt = 0; // 全局时间戳变量

bool search(int x)
{
    for(int i = 1; i <= m; i++)
        if(map[x][i] && vis[i] != ttt)
        {
            vis[i] = ttt;
            if(pre[i] == -1 || search(pre[i]))
            {
                pre[i] = x;
                return 1;
            }
        }
    return 0;
}

int match()
{
    int res = 0;
    for(int i = 1; i <= n; i++)
    {
        ++ttt; // 这里不用memset节省时间
        res += search(i);
    }
    return res;
}
```

1.5.2 Optimal Matching - KM

不会... 用费用流解决

1.6 Network flow

1.6.1 Maximum Flow - isap

```
//   h[x] :   点 x 在第 h[x] 层
//   v[k] :   第 k 层有 v[k] 个点
int sap(int x, int flow)
{
    if(x == n) return flow;
    int res = 0;
    for(int i = S; i <= T; i++)
        if(g[x][i] && h[x] == h[i] + 1)
        {
            int t = sap(i, min(g[x][i], flow - res));
            res += t; g[x][i] -= t; g[i][x] += t;
            if(res == flow) return res;
```

```
            if(h[S] >= T) return res;
        }
    //if(h[S] >= T) return res;
    if((--v[h[x]]) == 0) h[S] = T;
    ++v[++h[x]];
    return res;
}
int main()
{
    v[0] = T;
    int maxflow = 0;
    while(h[S] < T) maxflow += sap(1, inf);
    reutrn 0;
}
```

1.6.2 Minimum Cost Maximum Flow - spfa

```
struct EG{int from,to,flow,cost,next;}edge[M];

void add_edge(int a,int b,int c,int d)
{
    edge[L]=(EG){a,b,c,+d,head[a]};
    head[a]=L++;
    edge[L]=(EG){b,a,0,-d,head[b]};
    head[b]=L++;
}

bool spfa()
{
    memset(inQ, 0, sizeof(inQ));
    memset(dist, 0x3f, sizeof(dist));
    dist[S] = 0;
    q.push(S);
    while(!q.empty())
    {
        int x = q.front();
        q.pop();
        inQ[x] = 0;
        for(int i = head[x]; i != -1; i = edge[i].next)
            if(edge[i].flow && dist[edge[i].to] > dist[x] + edge[i].
               cost)
            {
                pre[edge[i].to] = i;
                dist[edge[i].to] = dist[x] + edge[i].cost;
                if(!inQ[edge[i].to])
                {
                    inQ[edge[i].to] = 1;
                    q.push(edge[i].to);
                }
            }
    }
    return dist[T] != inf;
```

```
}
void MFMC()
{
    memset(head, -1, sizeof(head));
    建图调用 add_edge();

    int mincost = 0, maxflow = 0;
    while(spfa())
    {
        int res = inf;
        for(int i = T; i != S; i = edge[pre[i]].from)
        {
            res = min(res, edge[pre[i]].flow);
        }
        for(int i = T; i != S; i = edge[pre[i]].from)
        {
            edge[pre[i]].flow -= res;
            edge[pre[i] ^ 1].flow += res;
        }
        maxflow += res;
        mincost += res * dist[T];
    }
}
```

2 Tools

2.1 C char*

```
头文件cstring
strlen(s);//获取长度  O(N)
strcpy(a+2,b+1)//从b+1开始全部赋值给a+2开始的字符串
strncpy(a+2,b+1,2)//从b+1开始赋值2个给a+2开始的字符串
strcmp(a,b)//比较a和b的大小，相等返回0，a>b返回正整数
strcat(a,b)//相当于string类的 a += b;
strstr(a,b)-a;//返回b在a中第一次出现的位置，不存在返回NULL(即0)，由于-a,所以最后应该是-a
```

2.2 C++ std::string

```
//====初始化====
头文件string并加上std::
string s(str);//相当于string s=str;
string s(cstr);//把char数组类型的字符串cstr作为s的初值
s.clear();//清空，相当于 s="";

//====长度====
s.length();//获取s的长度，O(1)
s.size();//一样

//====插入删除====
```

```
s.insert(2, "a"); //在s的位置2插入string类字符串"a"
s.erase(2, 3); //从s的位置2开始删除3个字符

//====查找====
s.find("abc");//查找字符串"abc"在s中第一次出现的位置（据说是KMP实现的）
//s="aabcc"; printf("%d %d\n",(int)s.find("abc"),(int)s.find("aabb"));
//上一行程序应输出 1 -1 （若没找到必须强行转换为int才为 -1 ）
```

2.3 Batch test

2.3.1 @Linux

```
mkdata=mk
filea=a
fileb=b

g++ $mkdata.cpp -o $mkdata
g++ $filea.cpp -o $filea
g++ $fileb.cpp -o $fileb
cas=0
while true; do
    ./$mkdata > $filea.in
    ./$filea < $filea.in > $filea.out
    ./$fileb < $filea.in > $fileb.out
    if ! diff $filea.out $fileb.out
    then
        echo "␣Wrong␣Answer"
        break
    fi
    echo $((cas=cas+1)) "␣Accepted"
done
```

2.3.2 @Windows

```
:loop
    mk > A.in
    A < A.in > A.out
    p < A.in > p.out
    fc A.out p.out
    if errorlevel 1 goto end
    goto loop
:end
    pause
```