[Ural 1099 Work Scheduling]【带花树】【Edmonds's matching algorithm】【一般图最大匹配】【例题】

发表于二月 **22, 2011**由**edward_mj**

【地址】http://acm.timus.ru/problem.aspx?space=1&num=1099

给定一个无向图，求最大匹配。

带花树……其实这个算法很容易理解，但是实现起来非常奇葩（至少对我而言）。

除了wiki和amber的程序我找到的资料看着都不大靠谱 

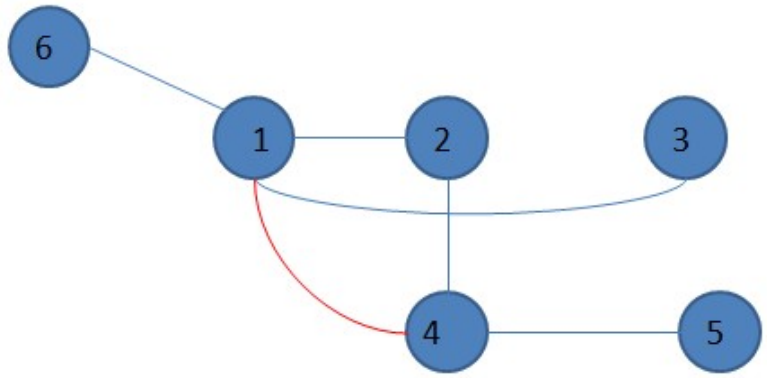
比如昨晚找到一篇鄙视带花树的论文，然后介绍了一种O(E)的一般图最大匹配……我以为找到了神论文，然后ACM_DIY众神纷纷表示这个是错的……于是神论文成为了”神论文“……

又比如围观nocow上带花树标程，一看……这显然是裸的匈牙利算法……货不对板啊

当然……如果二分图的匈牙利算法还不会请先围观求二分图最大匹配的匈牙利算法。

实际上任意图求最大匹配也是找增广路，但是由于奇环的出现，找增广路变得困难。

首先明确一点，增广路上是不能有重复出现的点的。

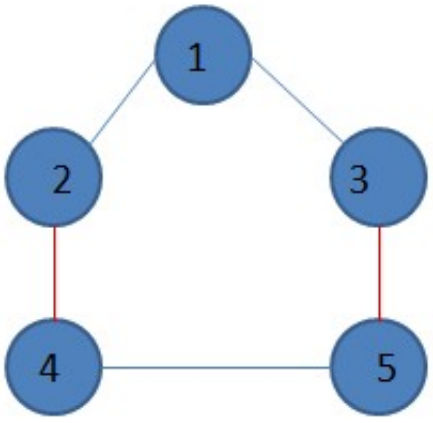


二分图中，匹配边可以看作是有向的，比如定义总是从X集指向Y集。假若定义了起点必须在X集中，那么增广路中出现该匹配边时，必然是按照这个方向的。所以一个点在增广路中的奇偶性是确定的。

而这个图中，从增广路3->1->4->5和2->4->1->6可以看出，对于有奇环的任意图，1和4这两个点在增广路中所在位置的奇偶性不再一定。于是我们考虑处理这些奇环。

定义奇环：包含2k+1个点和k条匹配边的一个环。（如果不是这样，我们找增广路不会走上去）

对于这个奇环，k条匹配覆盖了2k个点，那么显然有一个点未被覆盖。我们拿出这个点来讨论。



比如图中的1号点就是这个这个特殊的点。除了这个点以外，其它的点都被覆盖了，所以只能向外连非匹配边，而1号点可以向外连匹配边
或非匹配边。

如果1号点没有被外面的点匹配，那么无论从其它的哪个点走进来，都能以1为终点找到增广路。(要么顺时针跑到1，要么逆时针)

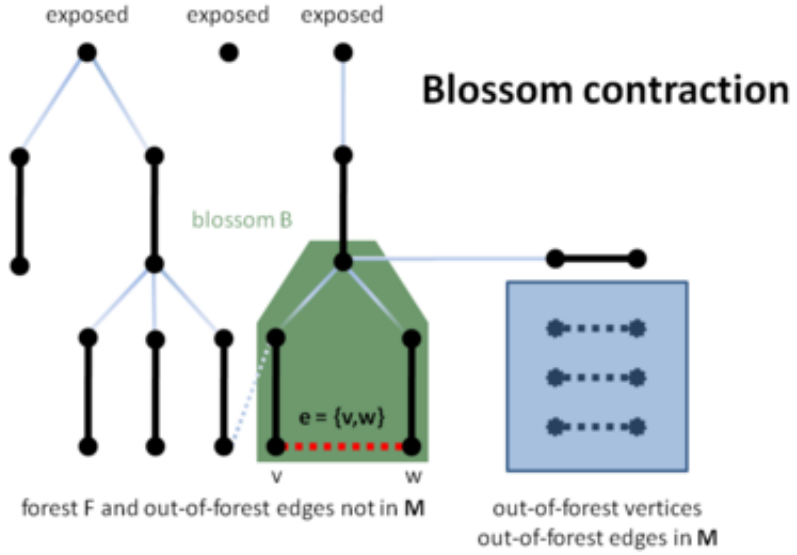
同理如果1号点被外面的点匹配了，那么无论从其它的哪个点走进来，都能把这个圈看成一个点，然后从1的那条匹配边穿出去。(要么顺时针，要么逆时针)

于是这个奇环就可以看成一个点，其主要特性由1号点体现(诸如和谁匹配了之流)。

这个合成点就叫做花。这个算法的思想就是不断地把奇环合成点，直至找到增广路（合成了某朵花以后就把整朵花当成一个点）。

考虑用BFS搜索增广路。

围观wiki这个图



由于BFS的性质，我们找到奇环只能是和同层的点，或者下下一层的点。

然后奇环的关键点必然是这棵BFS树里深度最浅的点。然后考虑合成以后，花如何展开对应的路径，使得我们能够增广。

花套花这个东西想起来都纠结>_<。

amber的程序里面并没有把点真的合成，只是弄了一个表示集合的标号：Base，然后邻接矩阵就不用变来变去了。

对于花中连向父亲的是匹配边的点，他的增广路显然是直接顺着父亲走，而如果连向父亲的边是非匹配边的点，那么显然是往后走然后跑过红色的横插边，然后再向上跑回关键点。

注意到如果连向父子的边是匹配边的点原先是不需要Father这个域来描述的，直接用表示匹配的那个域就可以了。但是现在在花中，他的Father这个域就要起作用了，用来向后指向，然后绕过红色横插边然后再跑回关键点。

 实在是太精妙了。

【代码】

```
#include

#include

#include

#include

#include

using namespace std;

const int N=250;

int n;

int head;

int tail;

int Start;

int Finish;

int link[N];     //表示哪个点匹配了哪个点

int Father[N];   //这个就是增广路的Father……但是用起来太精髓了

int Base[N];     //该点属于哪朵花

int Q[N];

bool mark[N];

bool map[N][N];
```

```
bool lnBlossom[N];

bool in_Queue[N];



void CreateGraph(){

    int x,y;

    scanf("%d",&n);

    while (scanf("%d%d",&x,&y)!=EOF)

      map[x][y]=map[y][x]=1;

}



void BlossomContract(int x,int y){

    fill(mark,mark+n+1,false);

    fill(lnBlossom,lnBlossom+n+1,false);

    #define pre Father[link[i]]

    int lca,i;

    for (i=x;i;i=pre) {i=Base[i]; mark[i]=true; }

    for (i=y;i;i=pre) {i=Base[i]; if (mark[i]) {lca=i; break;} }  //寻找lca之旅……一定要注意i=Base[i]

    for (i=x;Base[i]!=lca;i=pre){

        if (Base[pre]!=lca) Father[pre]=link[i]; //对于BFS树中的父边是匹配边的点，Father向后跳

        lnBlossom[Base[i]]=true;

        lnBlossom[Base[link[i]]]=true;

    }
```

```
    for (i=y;Base[i]!=lca;i=pre){
        if (Base[pre]!=lca) Father[pre]=link[i]; //同理
        InBlossom[Base[i]]=true;
        InBlossom[Base[link[i]]]=true;
    }
    #undef pre
    if (Base[x]!=lca) Father[x]=y;     //注意不能从lca这个奇环的关键点跳回来
    if (Base[y]!=lca) Father[y]=x;
    for (i=1;i<=n;i++)
      if (InBlossom[Base[i]]){
          Base[i]=lca;
          if (!in_Queue[i]){
              Q[++tail]=i;
              in_Queue[i]=true;     //要注意如果本来连向BFS树中父结点的边是非匹配边的点，可能是没有入队的
          }
      }
}

void Change(){
    int x,y,z;
    z=Finish;
    while (z){
```

```
            y=Father[z];
            x=link[y];
            link[y]=z;
            link[z]=y;
            z=x;
        }
    }

    void FindAugmentPath(){
        fill(Father,Father+n+1,0);
        fill(in_Queue,in_Queue+n+1,false);
        for (int i=1;i<=n;i++) Base[i]=i;
        head=0; tail=1;
        Q[1]=Start;
        in_Queue[Start]=1;
        while (head!=tail){
            int x=Q[++head];
            for (int y=1;y<=n;y++)
              if (map[x][y] && Base[x]!=Base[y] && link[x]!=y)  //无意义的
边
                if ( Start==y || link[y] && Father[link[y]] )   //精髓地用Father
表示该点是否
                    BlossomContract(x,y);
                else if (!Father[y]){
```

```
                Father[y]=x;
                if (link[y]){
                    Q[++tail]=link[y];
                    in_Queue[link[y]]=true;
                }
                else{
                    Finish=y;
                    Change();
                    return;
                }
            }
        }
    }

    void Edmonds(){
        memset(link,0,sizeof(link));
        for (Start=1;Start<=n;Start++)
          if (link[Start]==0)
            FindAugmentPath();
    }

    void output(){
        fill(mark,mark+n+1,false);
```

```
    int cnt=0;
    for (int i=1;i<=n;i++)
      if (link[i]) cnt++;
    printf("%dn",cnt);
    for (int i=1;i<=n;i++)
      if (!mark[i] && link[i]){
          mark[i]=true;
          mark[link[i]]=true;
          printf("%d %dn",i,link[i]);
      }
}

int main(){
//    freopen("input.txt","r",stdin);
    CreateGraph();
    Edmonds();
    output();
    return 0;
}
```