二分图匹配之最佳匹配——KM算法

今天也大致学了下KM算法，用于求二分图匹配的最佳匹配。

何为最佳？我们能用匈牙利算法对二分图进行最大匹配，但匹配的方式不唯一，如果我们假设每条边有权值，那么一定会存在一个最大权值的匹配情况，但对于KM算法的话这个情况有点特殊，这个匹配情况是要在完全匹配（就是各个点都能一一对应另一个点）情况下的前提。

自然，KM算法跟匈牙利算法有相似之处。

其算法步骤如下：

1.用邻接矩阵（或其他方法也行啦）来储存图，注意：**如果只是想求最大权值匹配而不要求是完全匹配的话，请把各个不相连的边的权值设置为0。**

2.运用贪心算法初始化标杆。

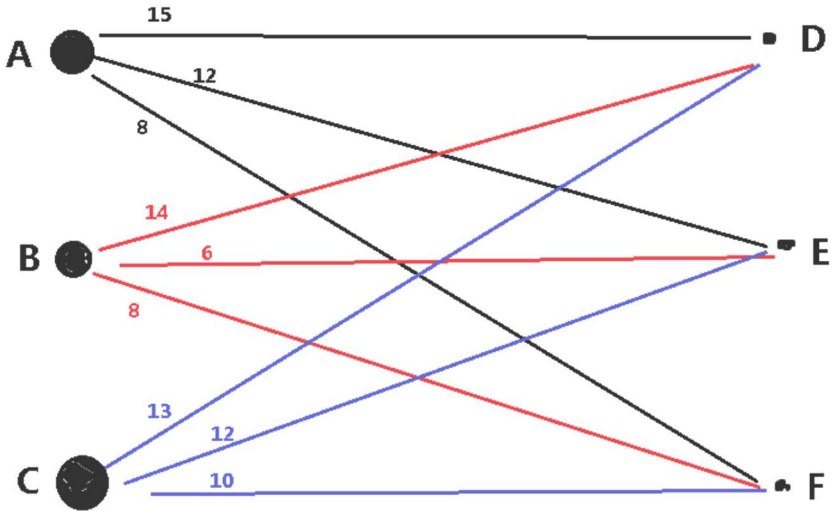
3.运用匈牙利算法找到完备匹配。

4.如果找不到，则通过修改标杆，增加一些边。

5.重复3，4的步骤，直到完全匹配时可结束。

一言不合地冒出了个标杆？？标杆是什么？？？

在解释这个问题之前，我们先来假设一个很简单的情况，用我们人类伟大的智能思维去思考思考。



如上的一个二分图，我们要求它的最大权值匹配（最佳匹配）

我们可以思索思索

二分图最佳匹配还是二分图匹配，所以跟和匈牙利算法思路差不多

二分图是特殊的网络流，最佳匹配相当于求最大（小）费用最大流，所以FF方法也能实现

所以我们可以把这匈牙利算法和FF方法结合起来

FF方法里面，我们每次是找最长（短）路进行通流

所以二分图匹配里面我们也找最大边进行连边!

但是遇到某个点被匹配了两次怎么办?

那就用匈牙利算法进行更改匹配!

这就是KM算法的思路了：尽量找最大的边进行连边，如果不能则换一条较大的。

所以，根据KM算法的思路，我们一开始要对边权值最大的进行连线，那问题就来了，我们如何让计算机知道该点对应的权值最大的边是哪一条？或许我们可以通过某种方式

记录边的另一端点，但是呢，后面还要涉及改边，又要记录边权值总和，而这个记录端点方法似乎有点麻烦，于是KM采用了一种十分巧妙的办法（也是KM算法思想的精髓）：

添加标杆（顶标）

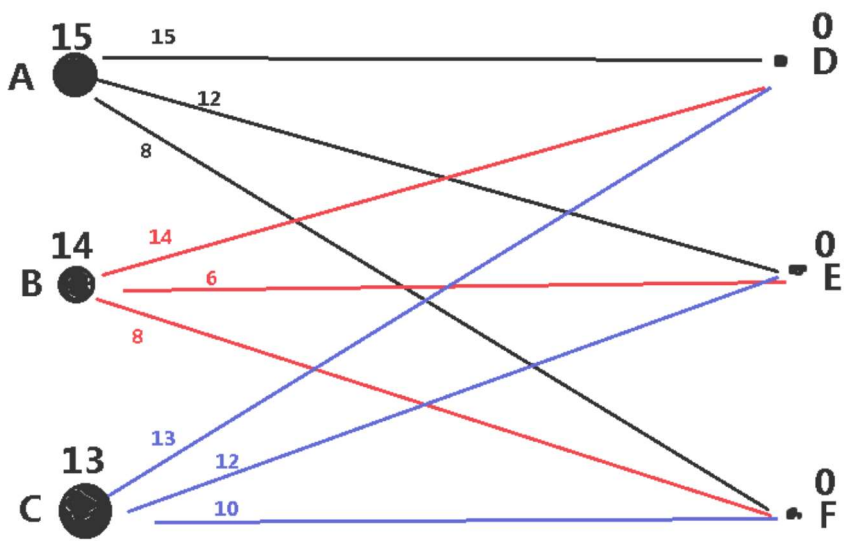
是怎样子呢？我们对左边每个点Xi和右边每个点Yi添加标杆Cx和Cy。

其中我们要满足Cx+Cy>=w[x][y]（w[x][y]即为点Xi、Yi之间的边权值）

对于一开始的初始化，我们对于每个点分别进行如下操作

Cx=max(w[x][y]);

Cy=0;



然后，我们可以进行连边，即采用匈牙利算法，只是在判断两点之间是否有连线的条件下，因为我们要将最大边进行连线，所以原来判断是否有边的条件w[x][y]==0换成了

Cx+Cy==w[x][y]

此时，有一个新的名词——**相等子图**。

因为我们通过了巧妙的处理让计算机自动连接边权最大的边，换句话说，其他边计算机就不会连了，也就“不存在”这个图中，但我们可以随时加上这些“不存在”图中的边。此时这个图可以认为是原图的子图，并且是等效。

这样，计算机在枚举右边的点的时候，满足以上条件，就能够知道这条边是我们要连的最大的边，就能进行连边了。

于是乎我们连了AD。

接下来就尴尬了，计算机接下来要连B点的BD，但是D点已经和A点连了，怎么办呢？？？

随笔档案

2019年8月(4)
2019年7月(1)
2017年9月(2)
2017年8月(44)
2017年7月(2)
2017年1月(3)
2016年12月(2)

友情链接

Krain~
蒟蒻のBLOG

最新评论

1. Re:二分图匹配之最佳匹配——KM算法
好呀!
--mool
2. Re:二分图匹配之最佳匹配——KM算法
52 for (int j=1;j<=n;j++) 这里的n是不是小心打错了，应该是m吧？...
--Morning_Glory
3. Re:二分图匹配之最佳匹配——KM算法
写的太好了
--UnicornXi
4. Re:二分图匹配之最佳匹配——KM算法
写的超级好。冬天看您的博客还真是应季呢~
--enceladus
5. Re:二分图匹配之最佳匹配——KM算法
还有楼主，这句一直不理解：如果只是想求最大权值匹配而不要求是完全匹配的话，请把各个不相连的边的权值设置为0。。最大权值匹配不是以完全匹配为前提的吗，而且不相连的边指的是？同问 一直想求一个不是完备匹配...
--stupor

阅读排行榜

1. 二分图匹配之最佳匹配——KM算法(11182)
2. 多叉树转换二叉树(4629)
3. 二分图匹配之最大匹配——匈牙利算法(2601)
4. Noip2016提高组 组合数问题problem(2110)
5. 笛卡尔树——神奇的“二叉搜索堆”(1277)

评论排行榜

1. 二分图匹配之最佳匹配——KM算法(11)
2. 多叉树转换二叉树(3)
3. JZOJ.5246【NOIP2017模拟8.8】Trip(2)

推荐排行榜

1. 二分图匹配之最佳匹
2. 多叉树转换二叉树(3)
3. 博弈论之Nim游戏(2)
4. 强连通分量tarjan缩点——[illegible]2186 Popular Cows(1)

5. 笛卡尔树——神奇的“二叉搜索堆”(1)

根据匈牙利算法，我们做的是将A点与其他点进行连线，但此时的子图里“不存在”与A点相连的其他边，怎么办呢？？

为此，我们就需要加上这些边！

很明显，我们添边，自然要加上不在子图中边权最大的边，也就是和子图里这个边权值差最小的边。

于是，我们再一度引入了一变量d，d=min{Cx[i]+Cy[j]-w[i][j]}

其中，在这个题目里Cx[i]指的是A的标杆，Cy[j]是除D点（即已连点）以外的点的标杆。

随后，对于原先存在于子图的边AD，我们将A的标杆Cx[i]减去d，D的标杆Cy[d]加上d。

这样，这就保证了原先存在AD边保留在了子图中，并且把不在子图的最大权值的与A点相连的边AE添加到了子图。

因为计算机判断一条边是否在该子图的条件是其两端的顶点的标杆满足

Cx+Cy==w[x][y]

对于原先的边，我们对左端点的标杆减去了d，对右端点的标杆加上了d，所以最终的结果还是不变，仍然是w[x][y]。

对于我们要添加的边，我们对于左端点减去了d，即Cx[i]=Cx[i]-d;为方便表示我们把更改后的的Cx[i]视为Cz[i]，即Cz[i]=Cx[i]-d;

对于右端点，我们并没有对其进行操作。那这条我们要添加边的两端点的标号是否满足Cz[i]+Cy[j]=w[i][j]?

因为Cz[i]=Cx[i]-d；d=Cx[i]+Cy[j]-w[i][j];

我们把d代入左式可得Cz[i]=Cx[i]-(Cx[i]+Cy[j]-w[i][j]);

化简得Cz[i]+Cy[j]=w[i][j]。

满足了要求！即添加了新的边。

值得注意的是，这里我们只是对于一条边操作，当我们添加了几条边，要进行如上操作时，要保证原先存在的边不消失，那么我们就要先求出了d，然后

对于每个连边的左端点（记作集合S）的每个点的标号减去了d之后，然后连边的右端点（记作T）加上d，这样就保证了原先的边不消失啦~

实际上这就是一直在寻找着增广路，通过不断修改标杆进行添边实现。

接下来就继续着匈牙利算法，直到完全匹配完为止。

该算法的正确性就在于 它每次都选择最大的边进行连边

至此，我们再回顾KM算法的步骤：

1.用邻接矩阵（或其他方法也行啦）来储存图。

2.运用贪心算法初始化标杆。

3.运用匈牙利算法找到完备匹配。

4.如果找不到，则通过修改标杆，增加一些边。

5.重复3，4的步骤，直到完全匹配时可结束。

是不是清楚了许多？？

因为二分图是网络流的一种特殊情况，在网络流里我们是通过不断的SPFA找到费用最大（小）的路径进行通流，跟这个有点类似。

如果我们要求边权值最小的匹配呢？？？

我们可以把边权值取负值，得出结果后再取相反数就可以了。

至于为什么，正负大小相反了嘛~

至此，这大概是我个人的一点点理解了，希望对您有所帮助。

若有不当之处还请大家指出QwQ。

```
#include<iostream>
#include<cstring>
#include<cstdio>
using namespace std;
const int qwq=0x7fffffff;
int w[1000][1000];  //w数组记录边权值
int line[1000],usex[1000],usey[1000],cx[1000],cy[1000];  //line数组记录右边端点所连的左端点， usex，usey数组记录是否曾访问过，也是判断是否在增广路上，cx，cy数组就是记录点的顶标
int n,ans,m;  //n左m右
bool find(int x){
    usex[x]=1;
    for (int i=1;i<=m;i++){
        if ((usey[i]==0)&&(cx[x]+cy[i]==w[x][i])){    //如果这个点未访问过并且它是子图里面的边
            usey[i]=1;
            if ((line[i]==0)||find(line[i])){    //如果这个点未匹配或者匹配点能更改
                line[i]=x;
                return true;
            }
        }
    }
    return false;
}
int km(){
    for (int i=1;i<=n;i++){  //分别对左边点依次匹配
        while (true){
        int d=qwq;
        memset(usex,0,sizeof(usex));
        memset(usey,0,sizeof(usey));
        if (find(i)) break;  //直到成功匹配才换下一个点匹配
        for (int j=1;j<=n;j++)
            if (usex[j])
             for (int k=1;k<=m;k++)
             if (!usey[k]) d=min(d,cx[j]+cy[k]-w[j][k]);  //计算d值
        if (d==qwq) return -1;
        for (int j=1;j<=n;j++)
         if (usex[j]) cx[j]-=d;
        for (int j=1;j<=m;j++)
         if (usey[j]) cy[j]+=d;      //添加新边
       }
    }
    ans=0;
    for (int i=1;i<=m;i++)
     ans+=w[line[i]][i];
    return ans;
}
int main(){
    while (~scanf("%d%d",&n,&m)){
    memset(cy,0,sizeof(cy));
    memset(w,0,sizeof(w));
    memset(cx,0,sizeof(cx));
    for (int i=1;i<=n;i++){
     int d=0;
     for (int j=1;j<=n;j++){
      scanf("%d",&w[i][j]);
      d=max(d,w[i][j]);    //此处顺便初始化左边点的顶标
       }
    cx[i]=d;
    }
```

```
58      memset(line,0,sizeof(line));
59      printf("%d\n",km());
60  }
61      return 0;
62  }
```

标签: 二分图

好文要顶 关注我 收藏该文

~Lanly~
关注 - 36
粉丝 - 25
+加关注

9 0

« 上一篇: 二分图匹配之最大匹配——匈牙利算法
» 下一篇: 多叉树转换二叉树

posted @ 2017-01-16 22:13 ~Lanly~ 阅读(11181) 评论(11) 编辑 收藏

评论列表

#1楼 2017-04-14 22:01 **Ztraveler**

写得真的不错%

支持(0) 反对(0)

#2楼 [楼主] 2017-04-29 16:00 **~Lanly~**

@ Ztraveler
谢谢^ ^

支持(0) 反对(0)

#3楼 2017-06-10 11:43 **linanwx**

绝世好文

支持(1) 反对(0)

#4楼 2017-07-18 14:42 **Jinke2017**

楼主，请问 最后KM算法的代码中33行，if (d==qwq) return -1;是不是意味着不存在完全匹配。

支持(0) 反对(0)

#5楼 2017-07-18 14:51 **Jinke2017**

还有楼主，这句一直不理解：如果只是想求最大权值匹配而不要求是完全匹配的话，请把各个不相连的边的权值设置为0。。最大权值匹配不是以完全匹配为前提的吗，而且不相连的边指的是?

支持(0) 反对(0)

#6楼 [楼主] 2017-07-27 17:19 **~Lanly~**

@ Jinke2017
对的，if (d==qwq) return -1;是意味着不存在完全匹配
然后我这里的理解是 最大匹配就是最大的匹配数，最佳匹配指完全（最大）匹配的最大值，然后最大权值匹配不关心完全匹配。不相连的边就是左边的某个点和右边的某个点没有联系，或者说不是相互喜欢，就不能连线的意思。有些题目会存在完全匹配并不是权值最大的情况，具体看题目描述吧。^ ^

支持(0) 反对(0)

#7楼 2018-01-16 09:55 **stupor**

还有楼主，这句一直不理解：如果只是想求最大权值匹配而不要求是完全匹配的话，请把各个不相连的边的权值设置为0。。最大权值匹配不是以完全匹配为前提的吗，而且不相连的边指的是？
同问 一直想求一个不是完备匹配的最大权匹配，不知道该怎么弄。。。

支持(0) 反对(0)

#8楼 2018-11-21 07:48 **enceladus**

写的超级好。冬天看您的博客还真是应季呢~

支持(0) 反对(0)

#9楼 2019-01-04 16:37 **kgxpbqbyt**

写的太好了

支持(0) 反对(0)

#10楼 2019-01-05 17:32 **Morning_Glory**

```
52 for (int j=1;j<=n;j++)
```

这里的n是不是小心打错了，应该是m吧？

支持(0) 反对(0)

#11楼 2019-07-31 16:34 **mool**

好呀！

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册， 访问 网站首页。

相关博文：
· 【原创】我的KM算法详解
· 二分图的最佳匹配(KM 算法)
· Kuhn-Munkres算法（二分图最大权匹配）
· 加权二分图之km算法
· 二分图最佳匹配