

As an ACMer, not only try, but also create.

公告

昵称: wide sea  
园龄: 7个月  
粉丝: 1  
关注: 1  
+ 加关注

日历

<	2012年8月						>
日	一	二	三	四	五	六	
29	30	31	1	2	3	4	
5	6	7	8	9	10	11	
12	13	14	15	16	17	18	
19	20	21	22	23	24	25	
26	27	28	29	30	31	1	
2	3	4	5	6	7	8	

统计

随笔 - 30  
文章 - 0  
评论 - 0  
引用 - 0

导航

博客园  
首页  
发新随笔  
发新文章  
联系  
订阅 XML  
管理

搜索

ACM暑假集训之网络流初章

网络流回顾

网络流问题:

- 1、网络最大流。 2、流量有上下界网络的最大流和最小流。 3、最小费用最大流。
- 4、流量有上下界网络的最小费用最大流等。

在求解网络流问题是求解其他一些图论问题的基础。如求解图的顶点连通度和边连通度、匹配问题等可转化为网络求解。

求解网络流涉及的几个主要概念:

- 1、 容量网络、残留网络、层次网络;
- 2、 增广路、前向弧、后向弧、网络流、割;
- 3、 弧的容量、流量;
- 4、 距离标号、活跃顶点、盈余、推进;
- 5、 可行流、预流、最大流、可改进量、最小割。
- 6、 拆点、路径压缩、最短路、DFS、BFS、二分搜索。

一、网络最大流

- 1、 一般增广路算法 (Generic Augmenting Path)
- 2、 最短增广路算法 (Shortest Augmenting Path)
- 3、 连续最短增广路算法——Dinic算法 (Successive Shortest Augmenting Path Algorithm)
- 4、 一般预流推进算法 (Generic Preflow-Push Algorithm)
- 5、 最高标号预流推进算法 (Highest-Label Preflow-Push Algorithm)
- 6、 SAP算法

详见附件。

## 常用链接

我的随笔  
我的评论  
我的参与  
最新评论  
我的标签

## 我的标签

map(1)  
set(1)  
八数码(1)  
二部图(1)  
高斯消元(1)  
割顶集(1)  
双连通(1)

## 随笔分类

纯搜索(7)  
动态规划(1)  
二部图匹配(8)  
矩阵-高斯消元(1)  
算法回顾(5)  
图的连通(6)  
网络流(3)  
最短路  
最小生成树

## 随笔档案

2013年1月 (13)  
2012年12月 (1)  
2012年11月 (4)  
2012年10月 (6)  
2012年9月 (1)  
2012年8月 (3)  
2012年7月 (2)

## 最新评论

## 阅读排行榜

1. ACM暑假集训之网络流初章(297)  
2. ACM暑假集训之最小生成树(145)  
3. Tempter of the Bone\_ZOJ 2110(121)  
4. poj 2516 Minimum Cost 最小费用最大流(77)  
5. Red and Black\_poj

## 二、最小割的求解

1、 先求得最大流，然后依次枚举。

枚举思路：

- 1) 枚举一条边，并计算没有这条边的最大流，如最大流减少，则删去此边，记下此时的最大流，否则，不删边，枚举下一条边；这样反复枚举，计算，记录，直到最后最大流变为0.枚举结束，求得最小割的边。
- 2) 枚举一个点，并计算失去这个点的最大流，如最大流减少，则删去此点，并记下次时的最大流，否则，不删点，枚举下一个点；这样反复枚举，直至最大流为0.枚举结束，求得最少删去的点数。

这种思路简单，但要多次求最大流，容易超时。

2、 先求得最大流，然后DFS优先搜索遍历。

深搜思路：

在残余网络中从原点Vs出发进行DFS，历遍到的顶点做标记（条件为该边的残留容量不为0），这些点构成集合S，其余顶点构成顶点T。然后连接S到T的所有前向弧（后向弧也是饱和的，这不过它的正流量为0，所以不统计），这些弧的残留为0，也就是说这些弧是饱和的。

这种方法使用DFS完成，也可改用BFS，来求得最小割边集合，所用的时间相对较少。

## 三、流量有上下界的网络最大流和最小流

1、 网络：原始网络（Arc：上界、下届、流量）、伴随网络（Acc：上界、下届、流量）。

2、 Acc的构建：增加一个新的起点S，新的终点T；定点i 的入度 u 大于出度 v 时，增加一条S 到 i 的边，容量为u-v；相反，增加一条i 到 T 的边，容量为v-u；其它边的容量皆为上界 - 下界。

当S 到 各点的流量都满载时，存在可行流。

3、求原网络的可行流：Acc的流量 + Arc 的流量下界。

4、构建有可行流后的残留网络（网络中弧的容量为：流量上界 - 可行流量），然后求解该网络的最大流，最

后与可行流相加就是最大流；求最小流则只要在构建残留网络后，将源点和汇点交换，再求最大流，最后结果的相反值就是最小流。

网上有种Vt 到 Vs 连边的一种简易方式求最大最小流，使用二分搜索，详细参见：<http://wenku.baidu.com/view/0f3b691c59eef8c75fbfb35c.html>

## 四、最小费用最大流（最大流不唯一的情况）

1、 先使用floyd求出任意两点间的最短距离，再求出网络最大流，然后使用二

1979(70)

## 评论排行榜

1. poj 1422 最大匹配基础应用(0)
2. poj 1325 最小点覆盖集(0)
3. poj 1466 最大独立点数(0)
4. poj 3041 二分匹配基础(0)
5. poj 1469 二分匹配基础(0)

## 推荐排行榜

分法设置两点间最长距离的限制，以此为条件重新建立容量网络，求最大流，如最大流减小，则增大距离限制，否则减少，反复操作，直到取到最小的距离（距离最好是整数）。

2、求得从源点到汇点的一条最短路径（约束条件：该边可流通），然后求出这条路上的增广量；然后反复运行，直到没有一条从源点到达汇点的最短路，得出最大流中的最小费用。

附件：

## 1、一般增广路算法（Generic Augmenting Path）

 View Code



```

1 int cap[MAXN][MAXN], flow[MAXN][MAXN];
2 int S, T;
3 int pre[MAXN], aug[MAXN];
4 int queue[MAXN*MAXN], front, rear;
5 void ford()
// GAPA(一般增广路)( poj 1273 47ms )
6 {
7     int u, v, maxf, alpha;
8     maxf = 0;
9     while( true )
10     {
11         memset( pre, -1, sizeof(pre) );
12         pre[S] = S;    aug[S] = INF;
13         front = rear = 0;
14         queue[rear++] = S;
15         while( front != rear && pre[T] == -1 )
16         {
17             u = queue[front++];
18             for( v = S; v <= T; v++ )
19             {
20                 if( pre[v] == -1 )
21                 {
22                     if( cap[u][v] - flow[u][v] > 0 )
23                     {
24                         pre[v] = u;
25                         aug[v] = min( aug[u], cap[u][v] -
flow[u][v] );
26                         queue[rear++] = v;
27                     }
28                     else if( flow[v][u] > 0 )
29                     {
30                         pre[v] = -u;
31                         aug[v] = min( aug[u], flow[v][u] );
32                         queue[rear++] = v;
33                     }
34                 }
35             }
36         }

```

```
37         if( pre[T] == -1 )break;
38         alpha = aug[T];
39         maxf += alpha;
40         for( v = T, u = abs( pre[T] ); v != S; v = u,
u = abs( pre[u] ) )
41         {
42             flow[u][v] += alpha;
43             flow[v][u] -= alpha;
44         }
45     }
46     printf( "%d\n", maxf );
47 }
```



## 2、最短增广路算法 (Shortest Augmenting Path)

View Code



```
1 int cap[MAXN][MAXN], flow[MAXN][MAXN]; //
SAP(最短增广路)( poj 1273 47ms )
2 int rk[MAXN], vis[MAXN], aug[MAXN], pre[MAXN];
3 int S, T;
4 int queue[MAXN*MAXN], front, rear;
5 bool bfs()
6 {
7     memset( rk, -1, sizeof(rk) );
8     memset( vis, 0, sizeof(vis) );
9     int u, v;
10    rk[S] = 0;
11    front = rear = 0;
12    queue[rear++] = S;
13    while( front != rear )
14    {
15        u = queue[front++];
16        if( vis[u] )continue;
17        vis[u]=1;
18        for(v = S; v <= T; v++ )if( cap[u][v] -
flow[u][v] > 0 )
19        {
20            if( rk[v] == -1 )
21                rk[v] = rk[u] + 1;
22            else
23                rk[v] = min( rk[v], rk[u] + 1 );
24            if( !vis[v] )
25                queue[rear++] = v;
26        }
27    }
28    if( rk[T] == -1 )return false;
29    return true;
30 }
31 void BFS()
```

```
32 {
33     int u, v, maxf;
34     maxf = 0;
35     while( true )
36     {
37         if( !bfs() ) break;
38         memset( aug, 0, sizeof(aug) );
39         memset( vis, 0, sizeof(vis) );
40         pre[S] = 0;    aug[S] = INF;
41         front = rear = 0;
42         queue[rear++] = S;
43         vis[S] = 1;
44         while( front != rear && !vis[T] )
45         {
46             u = queue[front++];
47             for( v = S; v <= T; v++ )if( !vis[v] && rk[v]
== rk[u] + 1 )
48             {
49                 if( cap[u][v] - flow[u][v] > 0 )
50                 {
51                     pre[v] = u;
52                     aug[v] = min( aug[u], cap[u][v] -
flow[u][v] );
53                     queue[rear++] = v;
54                     vis[v] = 1;
55                 }
56             }
57         }
58         if( !vis[T] || aug[T] == 0 )break;
59         for(u = pre[T], v = T; u; v = u, u = pre[u] )
60         {
61             flow[u][v] += aug[T];
62             flow[v][u] -= aug[T];
63         }
64         maxf += aug[T];
65     }
66     printf( "%d\n", maxf );
67 }
```



3、连续最短增广路算法——Dinic算法（Successive Shortest Augmenting Path Algorithm）

View Code



```
1 // 矩阵(poj 1273 超时)
2 int cap[MAXN][MAXN]; //
Dinic(连续最短增广路)
3 int rk[MAXN], vis[MAXN];
4 int S, T;
5 int queue[MAXN*MAXN], front, rear;
```

```

6 bool bfs()
7 {
8     memset( rk, -1, sizeof(rk) );
9     memset( vis, 0, sizeof(vis) );
10    int u, v;
11    rk[S] = 0;
12    front = rear = 0;
13    queue[rear++] = S;
14    while( front != rear )
15    {
16        u = queue[front++];
17        if( vis[u] )continue;
18        vis[u]=1;
19        for(v = S; v <= T; v++ )if( cap[u][v] > 0 )
20        {
21            if( rk[v] == -1 )
22                rk[v] = rk[u] + 1;
23            else
24                rk[v] = min( rk[v], rk[u] + 1 );
25            if( !vis[v] )
26                queue[rear++] = v;
27        }
28    }
29    if( rk[T] == -1 )return false;
30    return true;
31 }
32 int DFS( int u, int cur )
33 {
34     if( u == T )return cur;
35     int aug, flow, v;
36     flow = cur;
37     for( v = S; v <= T; v++ )
38     {
39         if( rk[v] == rk[u] + 1 )
40         {
41             aug = DFS( v, min( cap[u][v], cur ) );
42             cap[u][v] -= aug;
43             cap[v][u] += aug;
44             cur -= aug;
45         }
46     }
47     return flow - cur;
48 }
49 int Dinic()
50 {
51     int maxf = 0;
52     while( bfs() )
53         maxf += DFS( S ,INF );
54     return maxf;
55 }
56 // 邻接表( poj 1273 16ms )
57 struct{
58     int v, c, next;
59 }Arc[MAXN*MAXN*2];

```

```


60 int tot, head[MAXN*2];
61 int S, T;
62 int rk[MAXN], vis[MAXN];
63 int queue[MAXN*MAXN*2], front, rear;
64 void init()
65 {
66     tot = 0;
67     memset( head, -1, sizeof(head) );
68 }
69 void add_edge( int u, int v, int c )
70 {
71     Arc[tot].v = v;
72     Arc[tot].c = c;
73     Arc[tot].next = head[u];
74     head[u] = tot++;
75     Arc[tot].v = u;
76     Arc[tot].c = 0;
77     Arc[tot].next = head[v];
78     head[v] = tot++;
79 }
80 bool bfs()
81 {
82     memset( rk, -1, sizeof(rk) );
83     memset( vis, 0, sizeof(vis) );
84     int i, u, v;
85     rk[S] = 1;
86     front = rear = 0;
87     queue[rear++] = S;
88     while( front != rear )
89     {
90         u = queue[front++];
91         if( vis[u] )continue;
92         vis[u] = 1;
93         for( i = head[u]; i != -1; i = Arc[i].next )
94         {
95             v = Arc[i].v;
96             if( Arc[i].c > 0 )
97             {
98                 if(rk[v]==-1)
99                     rk[v]=rk[u]+1;
100             else
101                 rk[v]=min(rk[v],rk[u]+1);
102             if( !vis[v] )queue[rear++] = v;
103         }
104     }
105 }
106 if( rk[T] == -1 )return false;
107 return true;
108 }
109 int dfs( int u, int cur )
110 {
111     if( u == T )return cur;
112     int i, v, flow, aug;
113     flow = cur;

```

```
114     for( i = head[u]; i != -1; i = Arc[i].next )
115     {
116         v = Arc[i].v;
117         if( rk[v] == rk[u] + 1 )
118         {
119             aug = dfs( v, min( Arc[i].c, cur ) );
120             Arc[i].c -= aug;
121             Arc[i^1].c += aug;
122             cur -= aug;
123         }
124     }
125     return flow - cur;
126 }
127 int Dinic()
128 {
129     int maxf = 0;
130     while( bfs() )
131         maxf += dfs( S, INF );
132     return maxf;
133 }
```



#### 4、一般预流推进算法 (Generic Preflow-Push Algorithm)

 View Code



```
1 // 矩阵( poj 1273 47ms )
2 int cap[MAXN][MAXN]; //
3
4 GPFA(一般预流推进)
5
6 3 int ef[MAXN], rk[MAXN];
7 4 int S, T;
8 5 int queue[MAXN*MAXN], front, rear;
9 6 void PUSH()
10 {
11     int u, v, aug, maxf;
12     memset( rk, 0, sizeof(rk) );
13     memset( ef, 0, sizeof(ef) );
14     ef[S] = INF;
15     ef[T] = -INF;
16     rk[S] = T + 1;
17     maxf = 0;
18     front = rear = 0;
19     queue[rear++] = S;
20     while( front != rear )
21     {
22         u = queue[front++];
23         for( v = S; v <= T; v++ )
24         {
25             if( ef[u] > cap[u][v] )
26                 aug = cap[u][v];
27             else
```



```

25         aug = ef[u];
26         if( aug > 0 && ( u == S || rk[u] == rk[v] +
1 ) )
27         {
28             cap[u][v] -= aug;
29             cap[v][u] += aug;
30             if( v == T )maxf += aug;
31             ef[u] -= aug;
32             ef[v] += aug;
33             if( v != S && v != T )
34                 queue[rear++] = v;
35         }
36     }
37     if( u != S && u != T && ef[u] > 0 )
38     {
39         rk[u]++;
40         queue[rear++] = u;
41     }
42 }
43 printf( "%d\n", maxf );
44 }
45 // 邻接表( poj 1273 0ms )
46 struct{
47     int v, c, next;
48 }Arc[MAXN*MAXN*2];
49 int tot, head[MAXN*2];
50 int ef[MAXN], rk[MAXN];
51 int S, T;
52 int queue[MAXN*MAXN*2], front, rear;
53 void init()
54 {
55     tot = 0;
56     memset( head, -1, sizeof(head) );
57 }
58 void add_edge( int u, int v, int c )
59 {
60     Arc[tot].v = v;
61     Arc[tot].c = c;
62     Arc[tot].next = head[u];
63     head[u] = tot++;
64     Arc[tot].v = u;
65     Arc[tot].c = 0;
66     Arc[tot].next = head[v];
67     head[v] = tot++;
68 }
69 void PUSH()
70 {
71     int i, u, v, aug, maxf;
72     maxf = 0;
73     memset( ef, 0, sizeof(ef) );
74     memset( rk, 0, sizeof(rk) );
75     ef[S] = INF;
76     ef[T] = -INF;
77     rk[S] = T + 1;

```

```
78     front = rear = 0;
79     queue[rear++] = S;
80     while( front != rear )
81     {
82         u = queue[front++];
83         for( i = head[u]; i != -1; i = Arc[i].next )
84         {
85             v = Arc[i].v;
86             if( ef[u] > Arc[i].c )
87                 aug = Arc[i].c;
88             else
89                 aug = ef[u];
90             if( aug > 0 && ( u == S || rk[u] == rk[v] +
1 ) )
91             {
92                 Arc[i].c -= aug;
93                 Arc[i^1].c += aug;
94                 if( v == T )maxf += aug;
95                 ef[u] -= aug;
96                 ef[v] += aug;
97                 if( v != S && v != T )
98                     queue[rear++] = v;
99             }
100         }
101         if( u != S && u != T && ef[u] > 0 )
102         {
103             rk[u]++;
104             queue[rear++] = u;
105         }
106     }
107     printf( "%d\n", maxf );
108 }
```



5、 最高标号预流推进算法 (Highest-Label Preflow-Push Algorithm)

6、 SAP算法

View Code



```
1 // 矩阵( poj 1273 32ms )
2 int cap[MAXN][MAXN]; //
SAP+gap[]优化, 矩阵与邻接表
3 int rk[MAXN], cur[MAXN], pre[MAXN], gap[MAXN];
4 int S, T;
5 int SAP( int V )
6 {
7     memset( cur, 0, sizeof(cur) );
8     memset( rk, 0, sizeof(rk) );
9     memset( gap, 0, sizeof(rk) );
10    int u, v, aug, maxf;
```

```

11     u = pre[S] = S;    aug = INF; maxf = 0;
12     gap[0] = V;
13     while( rk[S] < V )
14     {
15 loop:    for( v = cur[u]; v <= T; v++ )
16         {
17             if( cap[u][v] > 0 && rk[u] == rk[v] + 1 )
18             {
19                 aug = min( aug, cap[u][v] );
20                 pre[v] = u;
21                 cur[u] = v;
22                 u = v;
23                 if( v == T )
24                 {
25                     maxf += aug;
26                     for( u = pre[u]; v != S; v = u, u =
pre[u] )
27                     {
28                         cap[u][v] -= aug;
29                         cap[v][u] += aug;
30                     }
31                     aug = INF;
32                 }
33                 goto loop;
34             }
35         }
36     int minr = V;
37     for( v = S; v <= T; v++ )
38     {
39         if( cap[u][v] > 0 && rk[v] < minr )
40         {
41             cur[u] = v;
42             minr = rk[v];
43         }
44     }
45     if( !(--gap[rk[u]]) ) break;
46     gap[rk[u] = minr + 1]++;
47     u = pre[u];
48 }
49 return maxf;
50 }
51 //邻接表( poj 1273 0ms )
52 struct{
53     int v, c, next;
54 }Arc[MAXN*MAXN*2];
55 int tot, head[MAXN*2];
56 int gap[MAXN], cur[MAXN], pre[MAXN], rk[MAXN];
57 int S, T;
58 void init()
59 {
60     tot = 0;
61     memset( head, -1, sizeof(head) );
62 }
63 void add_edge( int u, int v, int c )

```

```

64 {
65     Arc[tot].v = v;
66     Arc[tot].c = c;
67     Arc[tot].next = head[u];
68     head[u] = tot++;
69     Arc[tot].v = u;
70     Arc[tot].c = 0;
71     Arc[tot].next = head[v];
72     head[v] = tot++;
73 }
74 int SAP( int V )
75 {
76     memset( rk, 0, sizeof(rk) );
77     memset( gap, 0, sizeof(gap) );
78     int i, u, v, aug, maxf;
79     for( i = S; i <= T; i++ ) cur[i] = head[i];
80     gap[0] = V;
81     u = pre[S] = S; aug = INF; maxf = 0;
82     while( rk[S] < V )
83     {
84 loop:     for( i = head[u]; i != -1; i = Arc[i].next )
85         {
86             v = Arc[i].v;
87             if( Arc[i].c > 0 && rk[u] == rk[v] + 1 )
88             {
89                 aug = min( aug, Arc[i].c );
90                 pre[v] = u;
91                 cur[u] = i;
92                 u = v;
93                 if( v == T )
94                 {
95                     maxf += aug;
96                     for( u = pre[u]; v != S; v = u, u =
pre[u] )
97                     {
98                         Arc[cur[u]].c -= aug;
99                         Arc[cur[u]^1].c += aug;
100                     }
101                     aug = INF;
102                 }
103                 goto loop;
104             }
105         }
106         int minr = V;
107         for( i = head[u]; i != -1; i = Arc[i].next )
108         {
109             v = Arc[i].v;
110             if( Arc[i].c > 0 && rk[v] < minr )
111             {
112                 cur[u] = i;
113                 minr = rk[v];
114             }
115         }
116         if( !(--gap[rk[u]]) ) break;

```

```
117         gap[rk[u] = minr + 1]++;
118         u = pre[u];
119     }
120     return maxf;
121 }
```



分类: 算法回顾

绿色通道:

好文要顶

关注我

收藏该文

与我联系



wide sea

关注 - 1

粉丝 - 1

+ 加关注

0

推荐

0

反对

(请您对文章做出评价)

« 博主上一篇: ACM暑假集训之最小生成树

» 博主下一篇: ACM暑假集训之二分匹配

posted on 2012-08-16 20:28 wide sea 阅读(298) 评论(0) 编辑 收藏

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](#)。

[博客园首页](#) [博问](#) [新闻](#) [闪存](#) [程序员招聘](#) [知识库](#)



最新IT新闻:

- LG LTE智能手机全球销量突破1000万大关
- 周鸿祎: 我不做教父 我不用微信
- 放翁: 开放2013
- 王垠: 漫谈 Linux, Windows 和 Mac
- 王垠: 你好, 世界。
- » 更多新闻...

最新知识库文章:

- [Facebook如何实现PB级别数据库自动化备份](#)
- [源代码管理十诫](#)
- [如何成为强大的程序员?](#)
- [Xen 虚拟机架构](#)
- [NoSQL的现状](#)
- » [更多知识库文章...](#)

网站建设

一对一服务 省时 省力 省心

全国统一业务咨询电话:

400 188 6666

www.zhubajie.com



Copyright © wide sea Powered by: 博客园 模板提供: 沪江博客