SAP算法心得

网络最大流算法是网络流算法的基础，实现方法很多，但时间复杂度与编程复杂度难于兼顾。一方面，诸如预流推进等高效算法难于编写调试，有时实际效果欠佳（参见dd_engi的评测）；另一方面，基于增广路的算法大多时间效率不高。于是许多人选择了相对简单的Dinic算法。事实上，SAP算法更易于理解，时间效率更高，编程更简单，思路更清晰。

（名词解释）SAP（Shortest Augmenting Paths）：　　最短增广路

算法基本框架：

- 定义距离标号为到汇点距离的下界。
- 在初始距离标号的基础上，不断沿可行弧找增广路增广，一般采用深度优先。可行弧定义为:

$$\{(i,j)\,|\,h[i]=h[j]+1\}$$

- 遍历当前节点完成后，为了使下次再来的时候有路可走（当然要满足距离标号的性质：不超过真实距离），重标号当前节点为:

$$\min\{h[j]\,|\,(i,j)\}+1$$

- 重标号后当前节点处理完毕。当源点被重标号后，检查其距离标号，当大于顶点数时，图中已不存在可增广路，此时算法结束；否则再次从源点遍历。
- 理论复杂度为:

$$O(n^2m)$$

我的心得：

- 理论上初始标号要用反向BFS求得，实践中可以全部设为0，可以证明：这样做不改变渐进时间复杂度。
- 理论上可以写出许多子程序并迭代实现，但判断琐碎，没有层次，实践中用递归简单明了，增加的常数复杂度比起SAP速度微乎其微却极大降低编程复杂度，易于编写调试。

- **★GAP优化★（性价比极高，推荐！**详见程序“//GAP”部分**）**
  注意到我们在某次增广后，最大流可能已经求出，因此算法做了许多无用功。可以发现，距离标号是单调增的。这启示我们如果标号中存在“间隙”，则图中不会再有可增广路，于是算法提前终止。实践中我们使用数组vh[i]记录标号为i的顶点个数，若重标号使得vh中原标号项变为0，则停止算法。

附效率测试与比较：

算法	高标推进	SAP(GAP)	Dinic	SAP(NOGAP)	BFS+EK
时间(s)	5.67	6.02	11.35	23.90	55.77

注:高标推进程序在180行左右。Dinic用的是dd_engi带BFS的标程，带GAP优化的SAP参见程序。

附源程序与注释（注意以下程序未用BFS，只用到递归，实现简单）：

```
program ditch;                                  //以USACO的ditch为例

var
 c:array[0..1000,0..1000] of cardinal;          //邻接矩阵
 h,vh:array[0..1000] of cardinal;
 n,augc,rd1,rd2,m,i:cardinal;                   //augc为增广路容量
 flow:cardinal=0;
 found:boolean;                                 //记录是否已达汇点

procedure aug(const m:cardinal);
var
 i,augco,minh:cardinal;
begin
 minh:=n-1;
 augco:=augc;
 if m=n then begin
  found:=true;
  inc(flow,augc);
  exit;
 end;
 for i:=1 to n do
  if c[m,i]>0 then begin
```

```
   if h[i]+1=h[m] then begin
    if c[m,i]<augc then augc:=c[m,i];
    aug(i);
    if h[1]>=n then exit;                    //GAP
    if found then break;
    augc:=augco;
   end;
   if h[i]<minh then minh:=h[i];
  end;
 if not found then begin                     //重标号
  dec(vh[h[m]]);                             //GAP
  if vh[h[m]]=0 then h[1]:=n;                //GAP
  h[m]:=minh+1;
  inc(vh[h[m]]);                             //GAP
 end else begin                              //修改残量
  dec(c[m,i],augc);
  inc(c[i,m],augc);
 end;
end;

begin
 assign(input,'ditch.in');
 assign(output,'ditch.out');
 reset(input);
 rewrite(output);

 readln(m,n);
 fillchar(c,sizeof(c),0);
 for i:=1 to m do begin
  readln(rd1,rd2,augc);
  inc(c[rd1,rd2],augc);
 end;
 fillchar(h,sizeof(h),0);
 fillchar(vh,sizeof(vh),0);
 vh[0]:=n;
 while h[1]<n do begin
  augc:=$FFFFFFFF;
```

```
    found:=false;
    aug(1);
  end;
  writeln(flow);

  close(input);
  close(output);
end.
```

Reference:

1.几种高级网络流算法的实测, dd_engi

2.*Network Flows: Theory, Algorithms, and Applications,* Ahuja, R.K.

3.http://www.gnocuil.cn/blog/2007/11/网络流的sap算法, gnocuil.