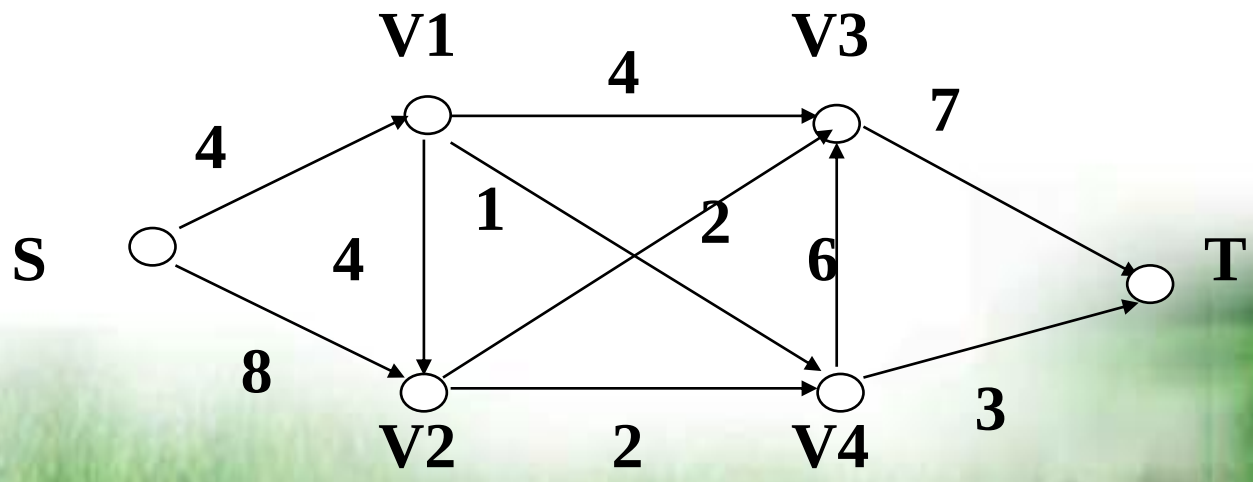
HNFMS

网络流

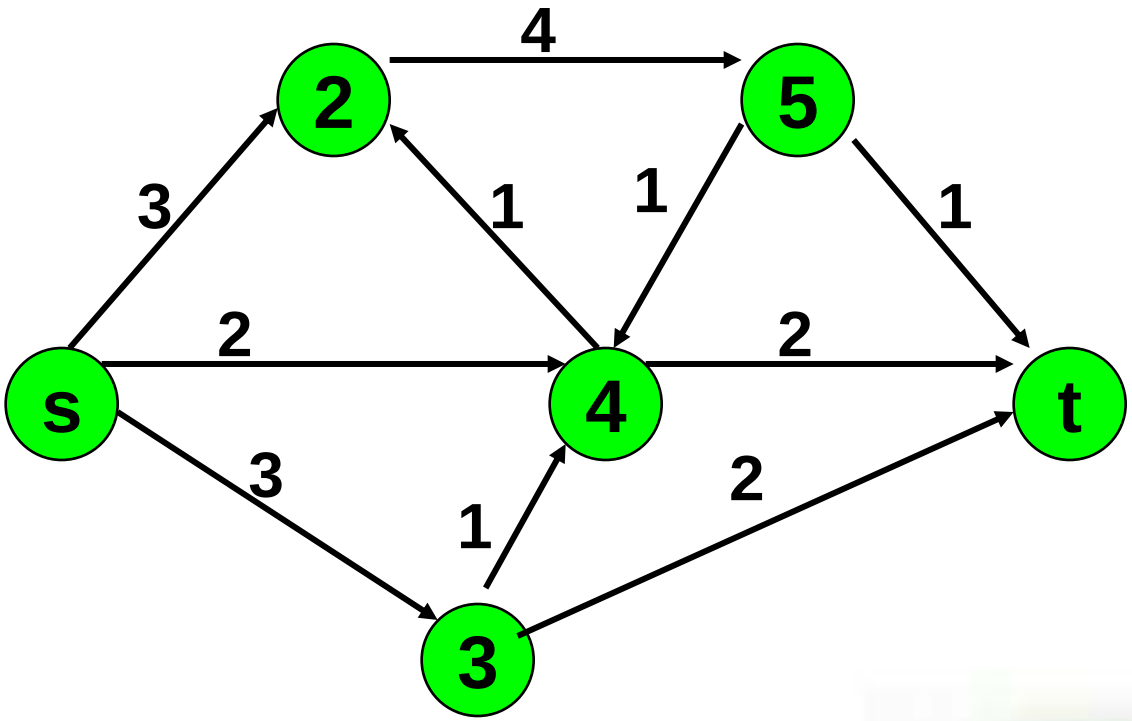
授课人：长沙市一中　周祖松

运输网络

- 现在想将一些物资从S运抵T，必须经过一些中转站。连接中转站的是公路，每条公路都有最大运载量。
- 每条弧代表一条公路，弧上的数表示该公路的最大运载量。最多能将多少货物从S运抵T？



公路运输图


4
2
5
3
1
1
1
2
2
s
4
t
3
1
2
3

基本概念

- 这是一个典型的网络流模型。为了解答此题，我们先了解网络流的有关定义和概念。
- 若有向图G=(V,E)满足下列条件：
  1. 有且仅有一个顶点S，它的入度为零，这个顶点S便称为源点，或称为发点。
  2. 有且仅有一个顶点T，它的出度为零，这个顶点T便称为汇点，或称为收点。
  3. 每一条弧都有非负数，叫做该边的容量。边(vi, vj)的容量用cij表示。
- 则称之为网络流图，记为G = (V, E, C)

可行流

- 在网络的一个可行流中，即任意一个实际运输情况中，实际运输量必须满足一些关系：
- 1—实际运输量不能是负的
- 2—每条弧的实际运输量不能大于该弧的容量
- 3—除了起点V_s和终点V_t外，对其他顶点来说指向V_i的弧上的运输量之和，应该等于所有从V_i出发的弧上的运输量之和

增广链P

- 若给定一个可行流F=(F_{ij}),我们把网络中$F_{ij}=C_{ij}$的弧称作饱和弧， $F_{ij}<C_{ij}$的弧称作非饱和弧，$F_{ij}=0$的弧称作零流弧， $F_{ij}>0$的弧称作非零流弧
- 若P是网络中联结源点s和汇点t的的一条路(不用管边的有向性)，我们定义路的方向是从V_s到V_t，则路上的弧被分为两类－一类与路的方向一致，称为前向弧；另一类和路的方向相反，称为后向弧

增广链P

- 如果对于P上的每一条前向弧都是非饱和弧，每一条后向弧都是非零流弧，则称P为一条可改进路。因为可以通过修正P上各弧的流量来使得总流量变得更大。修正的方法是：
- 1—不属于P的弧一概不变
- 2—对于P上的所有前向弧加上a，后向弧减去a。这里a是一个可改进量。

求网络的最大流

- a既要尽量大，又要保证变化后0<=F_{ij}<=C_{ij}。因此a=min(min($C_{前向弧ij}$－$F_{前向弧ij}$),min($F_{后向弧ij}$))。
- 如果不存在V_s到V_t的增广链，则该网络已经形成了最大流。
- 那么求最大流的算法可以大致表示为：不断寻找增广链P，并修改P，直到找不到P为止。
- 下面介绍一种用标号法来寻找可改进路的算法。

求最大流的标号法

- 每个顶点的标号分为两个部分：记录是否标号的部分和记录是否检查的部分。
- 标号法流程：
- 1－先标记源点s为已标号未检查
- 2－找出一个已标号未检查的点i
- 3－寻找所有未标号的和i相连的点j，弧F_{ij}必须满足增广链中对弧的要求。记j为已标号未检查。
- 4－记i为已标号已检查，并重复2－4的过程，直到汇点t已标号。

求最大流的标号法

- 由此我们得到了一条由s到t的可改进路。
- 先修改增广链，然后重复上述标号过程，直到找不出增广链为止。
- 最后我们便得到了整个网络的最大流。
- 至于增广链的记录和改进量a的计算等，可以在标号的过程中同时完成，这里可以参考程序。

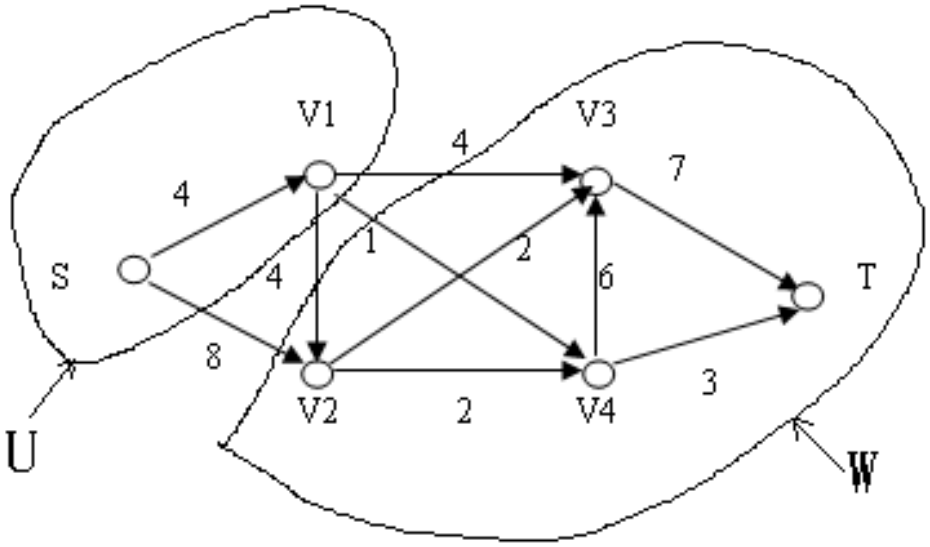
```
Int main()// 最大流
{ int i, j, delta, x;
while (1)
 {int  last={0} ;  //可改进路中的前趋
  int check[max]={0} ; //检查数组
   last[1] = maxint;   //给源点初始化
    while(last[n]==0)
    {  i = 1;
     while((i<=n)&&((last[i]==0)||check[i])
       i++;   //未找到已标号而未检查的点
     if (i > n)  break;
    for (j = 1;j<=n;j++)
      if (last[j] == 0)
         if (flow[i, j]<limit[i, j])  last[j] =i
         else if (flow[j, i]>0)       last[j] = -i;
    check[i] = 1;
   } //while(last[n]==0)
   if (last[n]== 0)  break;
   delta = maxint;    i = n;
   while (i!=1)
   {  j = i; i = abs(last[j]);
     if (last[j] > 0)
           x = limit[i, j] - flow[i, j]
      else  x = flow[j, i];
      if (x < delta)  delta = x;
   };     //求改进量
   i= n;
   while(i!=1)      //放大网络流
   {   j = i; i = abs(last[j]);
      if (last[j] > 0)  flow[i, j]+= delta;
      else              flow[j, i]-= delta;
    }   //while(i!=1)
  };   //while (1)
};
```

切割

- G = (V, E, C)是已知的网络流图，设U是V的一个子集，W = V\U，满足SU，TW。即U、W把V分成两个不相交的集合，且源点和汇点分属不同的集合。
- 对于弧尾在U，弧头在W的弧所构成的集合称之为**切割**，用（U，W）表示。把割切（U，W）中所有弧的容量之和叫做此切割的容量，记为C（U，W），即：

$$C(U,W)=\sum_{\substack{i\in U\\ j\in W}} c_{ij}$$

切割



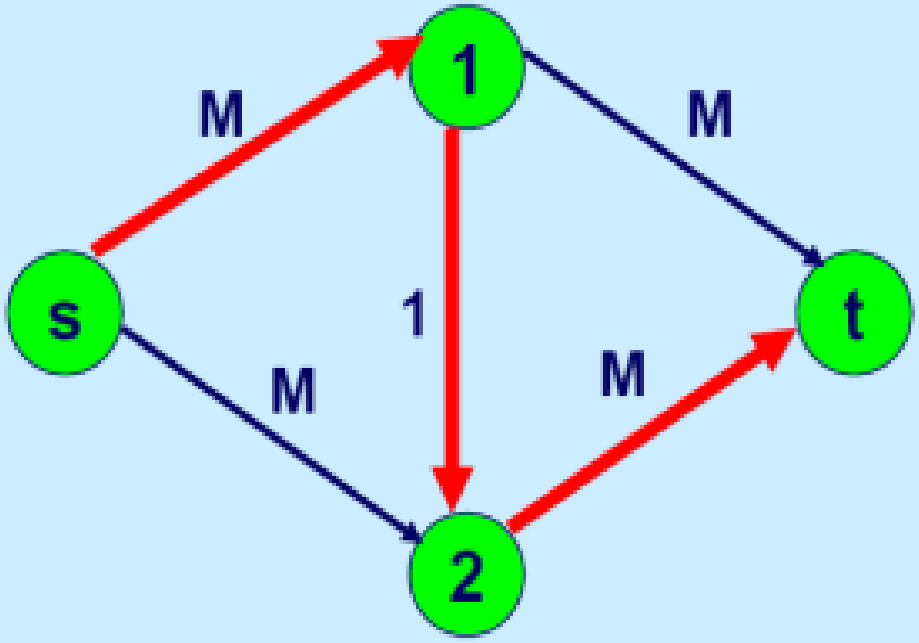
- 上例中，令U = {S, V1}，则W = {V2, V3, V4, T}，那么，
- C(U, W) = <S, V2> + <V1, V2> + <V1, V3>+<V1, V4> =8+4+4+1=17

流量算法的基本理论

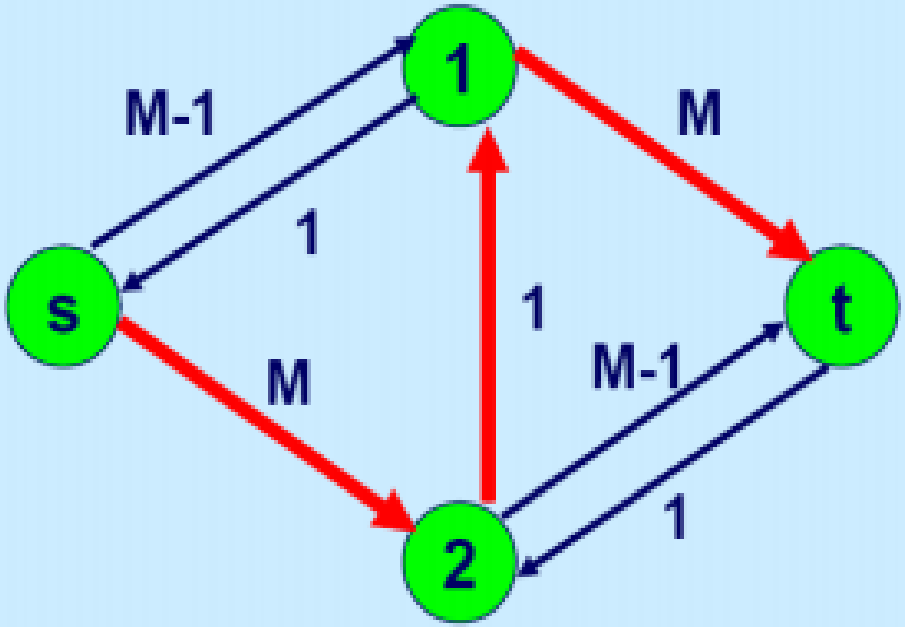
- **最大流最小割定理。**

  最大流等于最小割，即max V(f) = min C(U, W)。

一个简单但是非常坏的例子



1次增广后



两次增广后