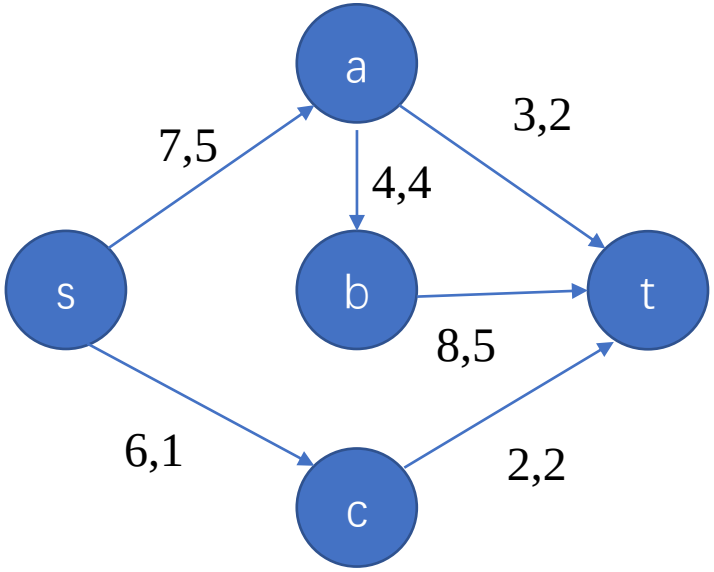
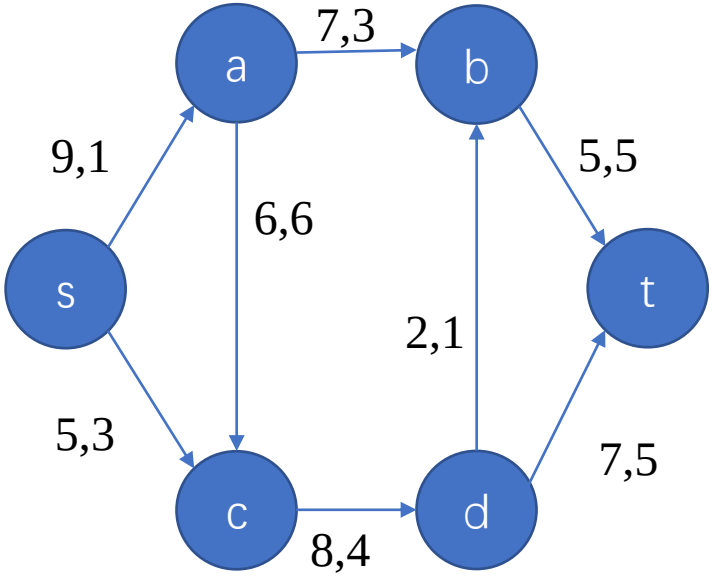
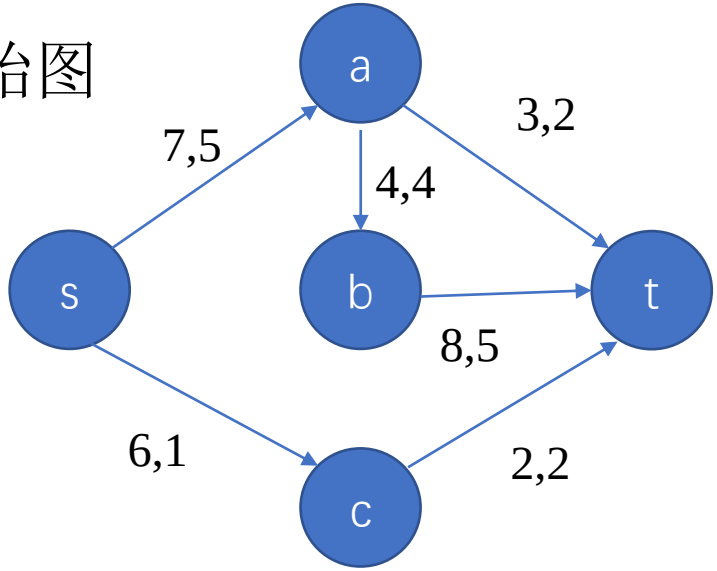
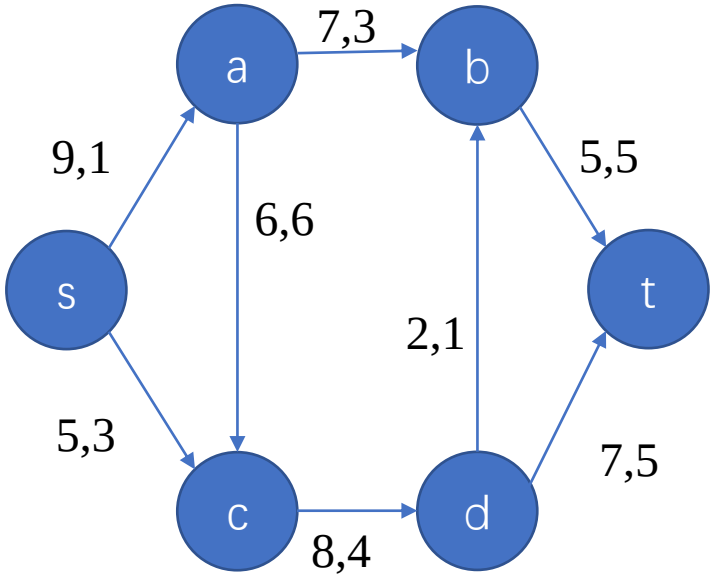
1. 给定如下两个带流的网络，画出对应的剩余图。



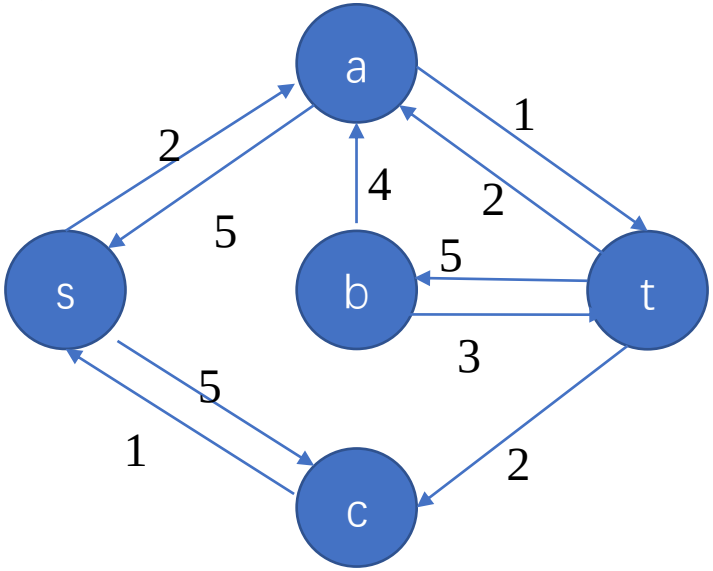


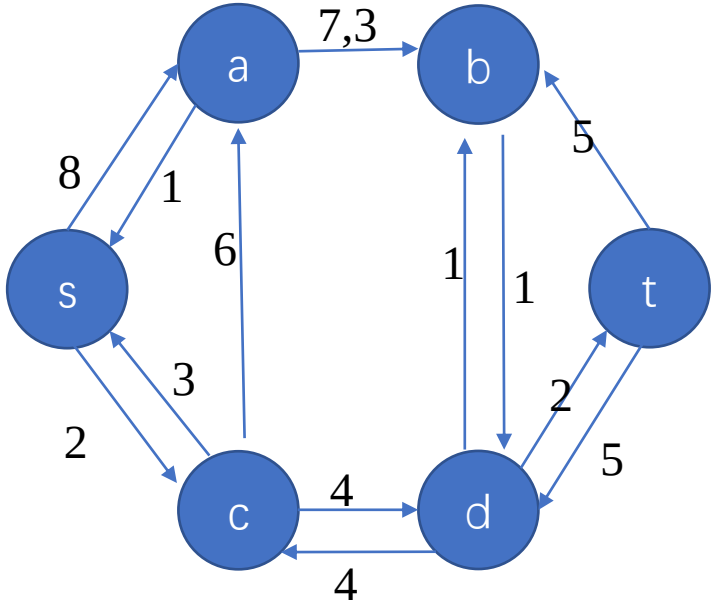
原始图


a
s
b
t
c
7,5
4,4
3,2
8,5
6,1
2,2

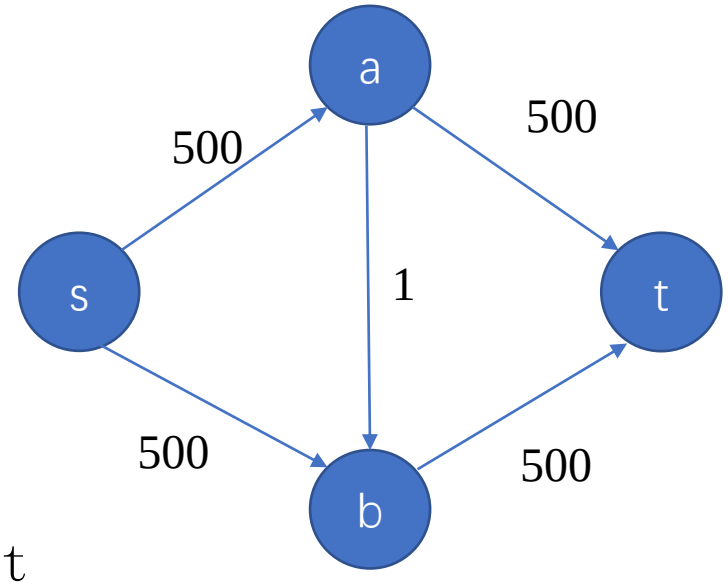

a
b
s
t
c
d
7,3
9,1
6,6
5,5
2,1
5,3
7,5
8,4

剩余图


a
s
b
t
c
2
1
4
5
2
5
3
5
1
2


a
b
s
t
c
d
7,3
8
1
5
6
1
1
3
2
2
5
4
4

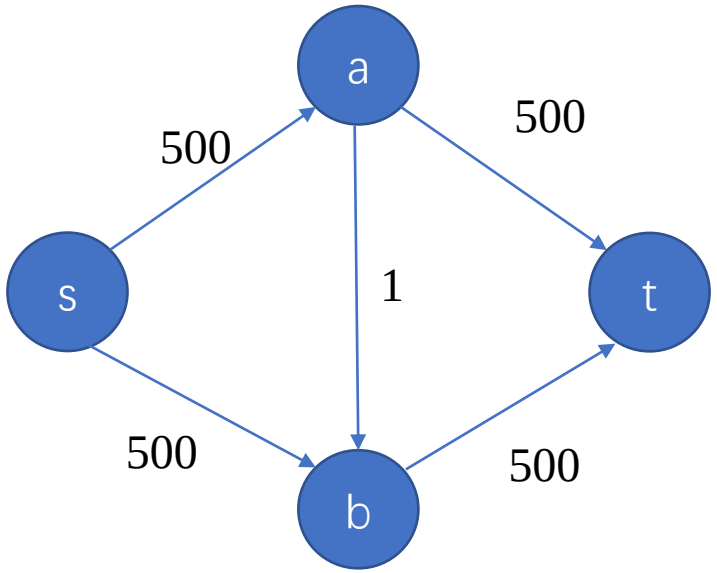
2. 给定如下网络，采用Ford-Fulkerson算法按照增广路径1,2交替迭代，并更新剩余图。
增广路径1：s-a-b-t，增广路径2：s-b-a-t



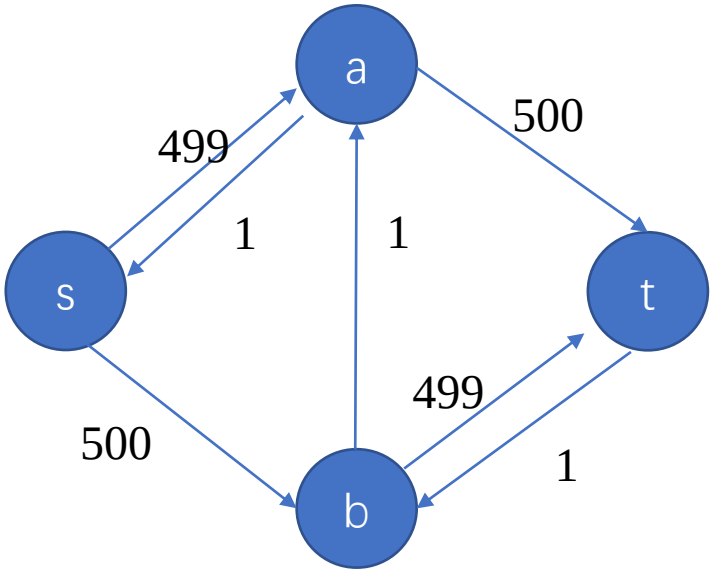
Ford-Fulkerson算法：
输入：网络（G,s,t,c）
输出：G中的一个流
1. 初始化剩余图，设R=G
2. for 边(u,v)∈E
3. f(u,v) <- 0
4. end for
5. While 在 R中有一条曾广路径p=s,…,t
6. 设△为p的瓶颈容量
7. for p 中的每条边(u,v)
8. f(u,v) <- f(u,v)+△
9. end for
10. 更新剩余图R
11. End while

2. 给定如下网络，采用Ford-Fulkerson算法按照增广路径1,2交替迭代，并更新剩余图。

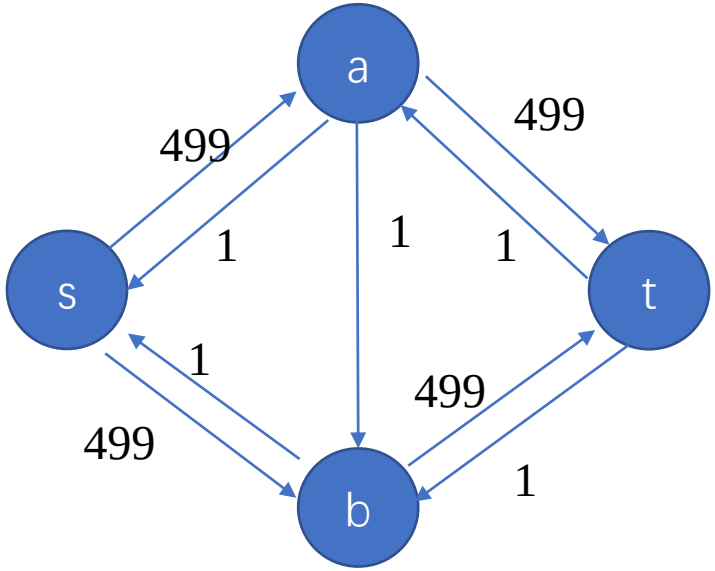
增广路径1：s-a-b-t
增广路径2：s-b-a-t



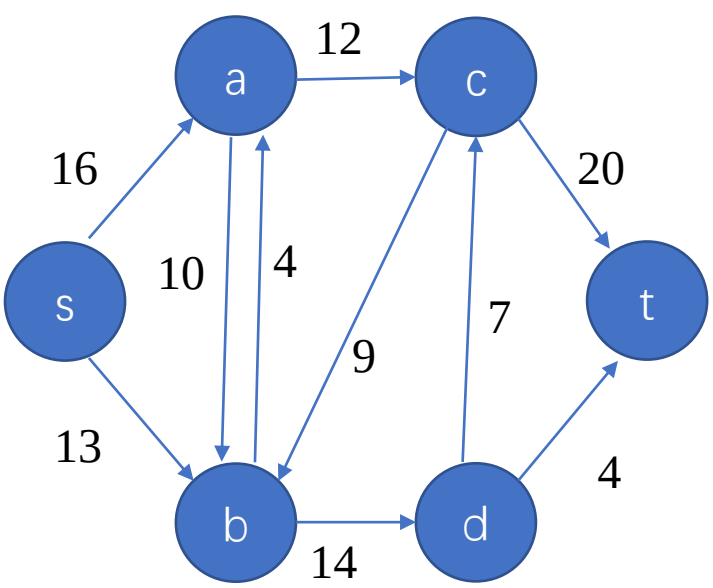
第一次迭代，路径s-a-b-t

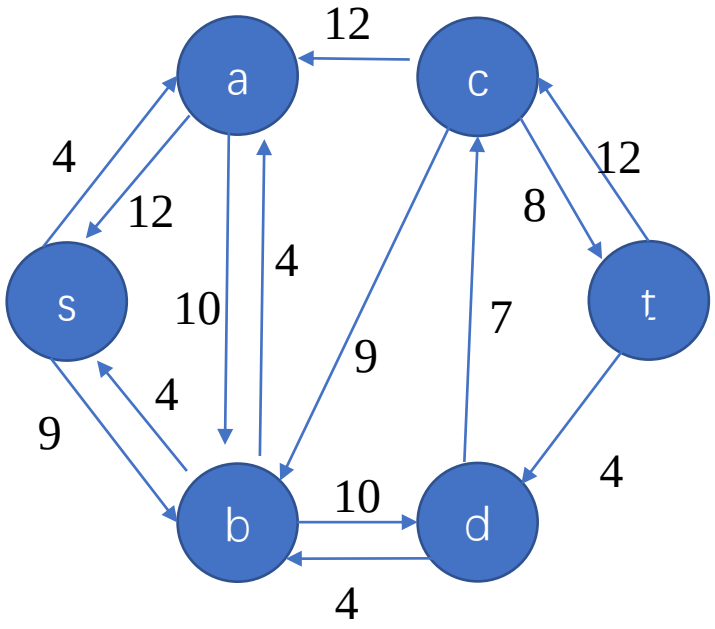


第二次迭代，路径s-b-a-t

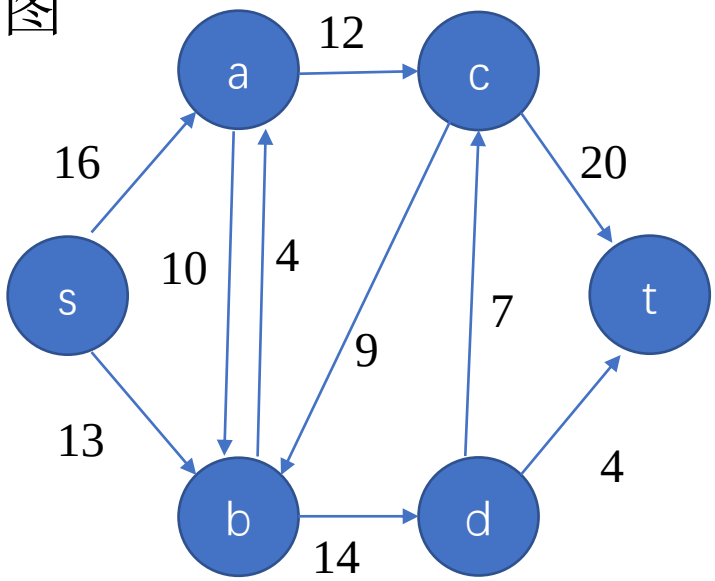


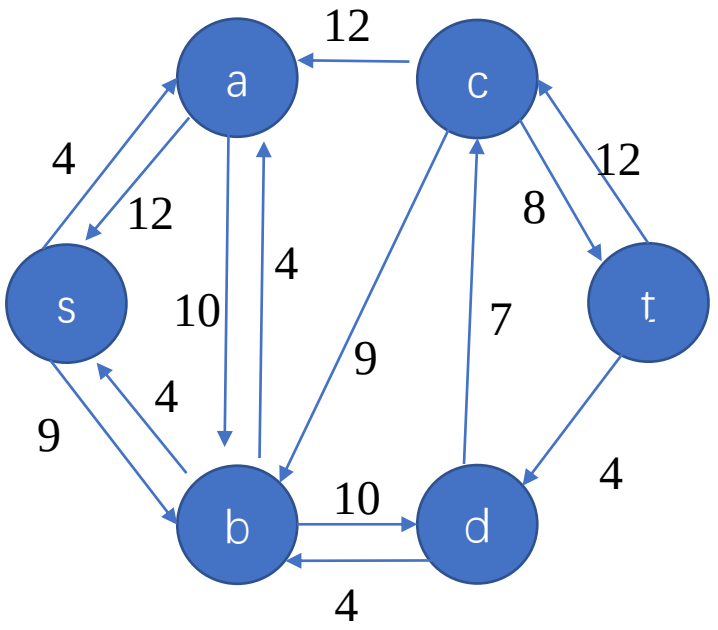
3. 给定如下网络，画出相应的层次图。



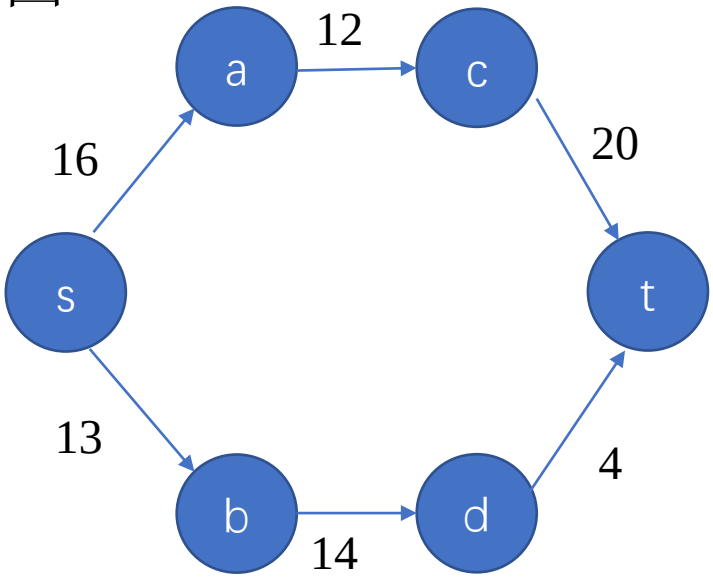


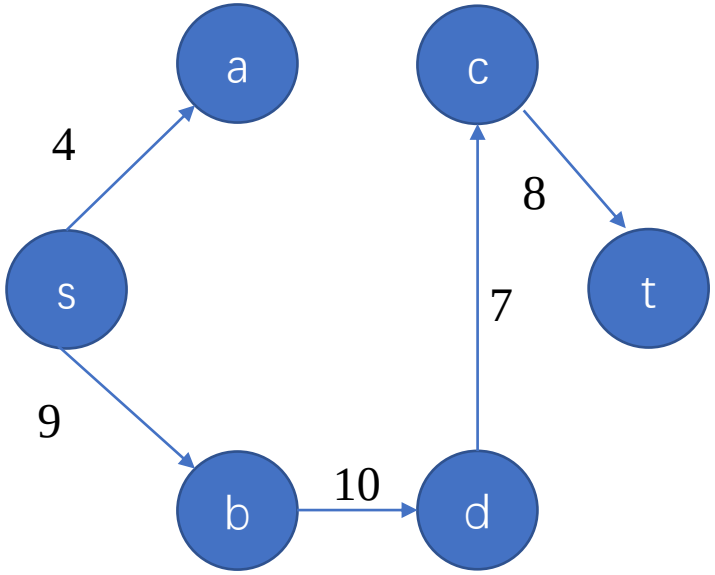
原始图


a
c
s
t
b
d
12
16
20
10
4
7
9
13
4
14


a
c
s
t
b
d
12
4
12
8
12
10
4
7
9
4
9
10
4
4

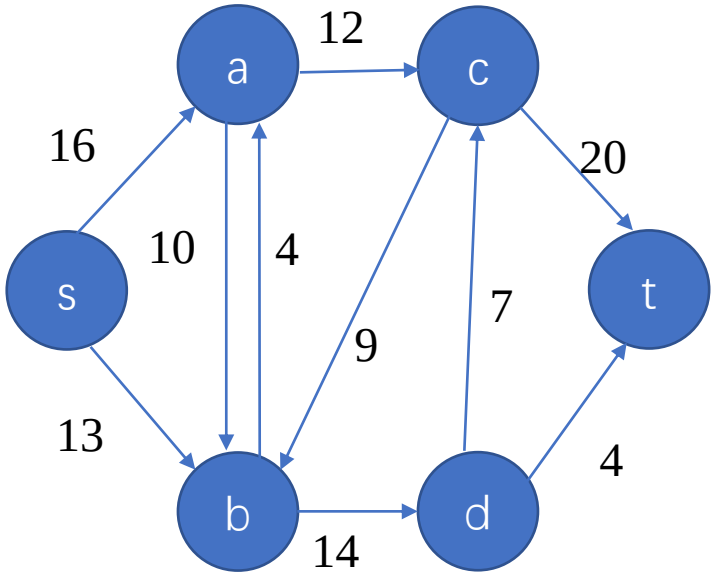
层次图

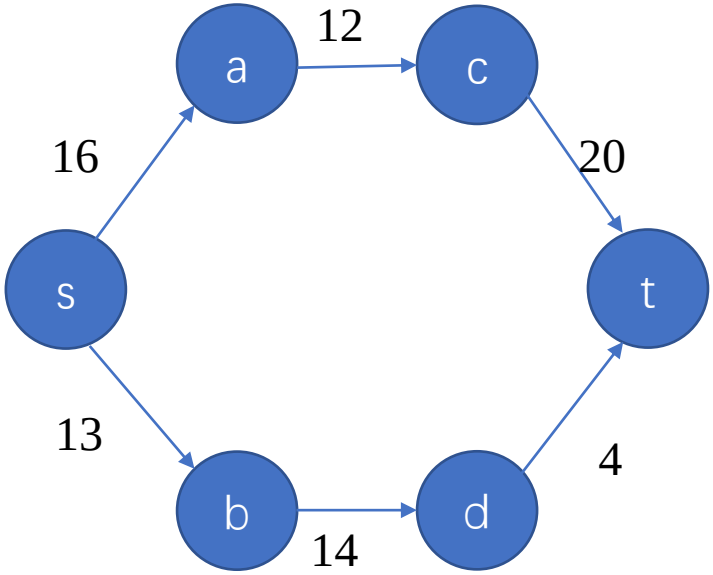

a
c
s
t
b
d
12
16
20
13
4
14


a
c
s
t
b
d
4
8
7
9
10

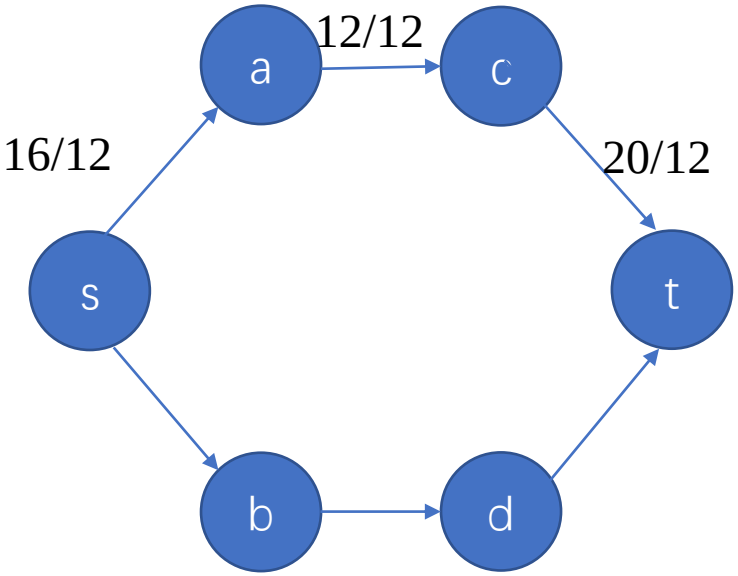
4. 给定如下网络，采用最小路径长度增值法（MPLA）计算最大流。

MPLA算法：
输入：网络(G,s,t,c)
输出：G中的最大流
1. for 每条边(u,v)∈E
2. f(u,v) <- 0
3. end for
4. 初始化剩余图，设R=G
5. 查找R的层次图L
6. while t 为 L中的顶点
7. while t 在L中能从s到达
8. 设p为L中从s到t的一条路径
9. 设△为p的瓶颈容量
10. 用△增值当前流f
11. 沿着路径p更新L和R
12. end while
13. 用剩余图R计算新的层次图L
14. end while

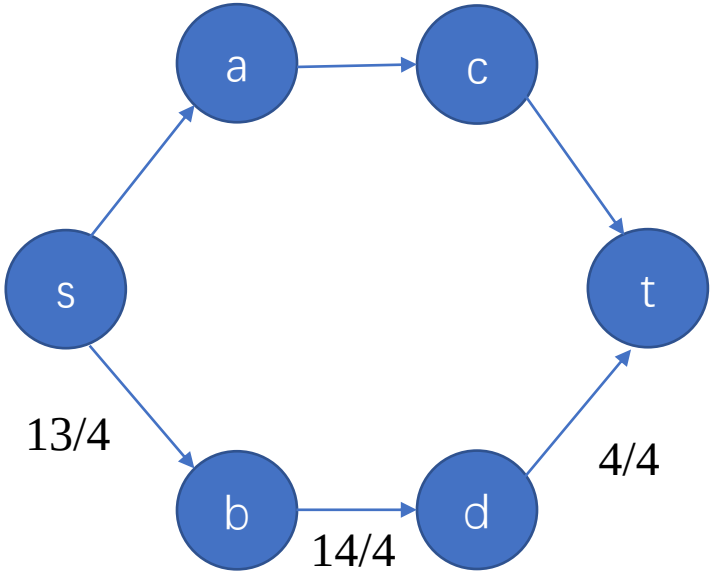




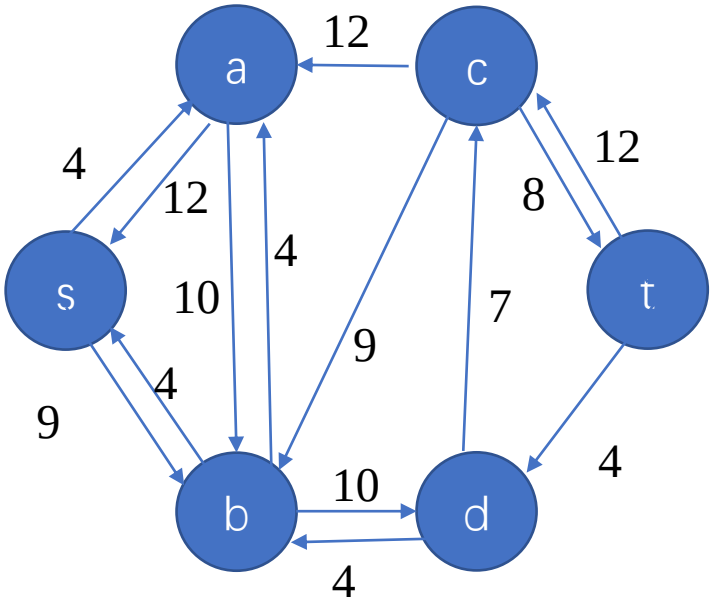
第一层次图



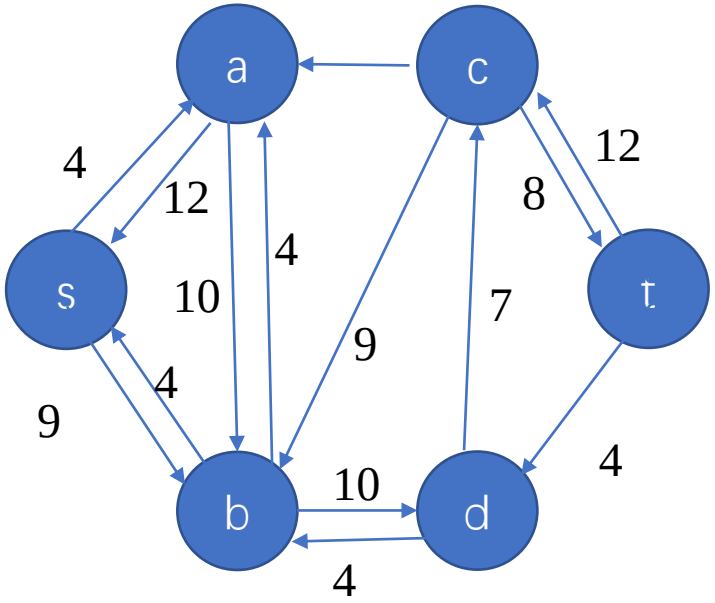
增值s, a, c, t



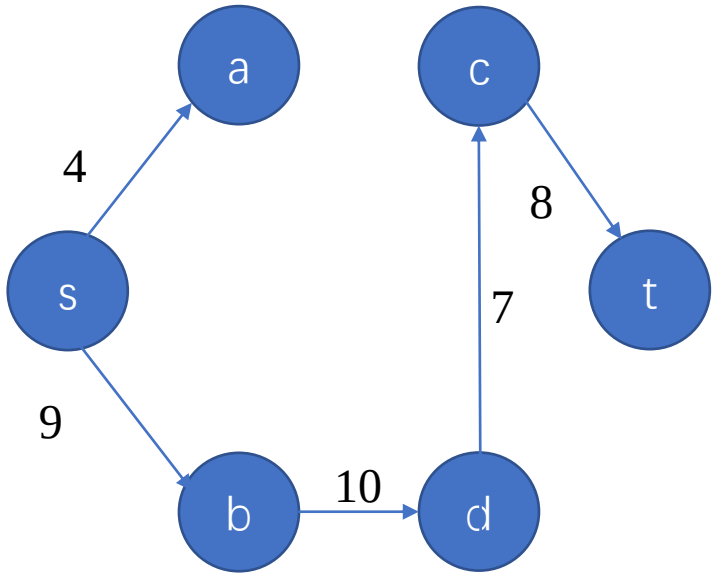
增值s, b, d, t



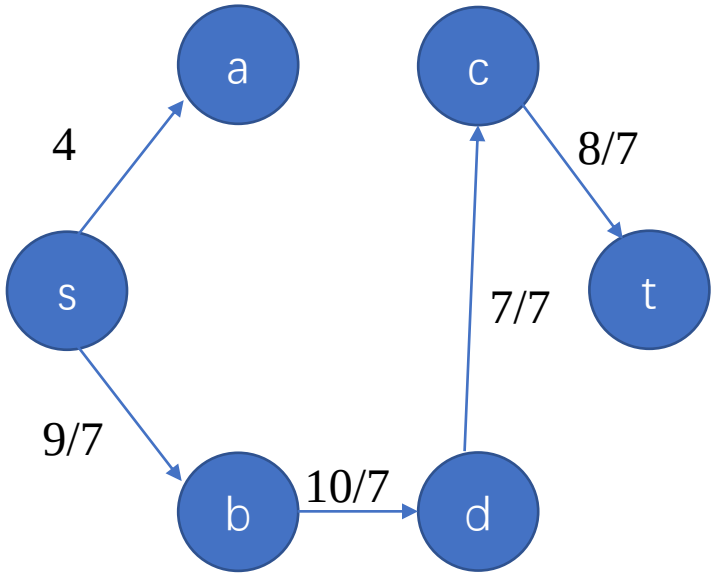
剩余图



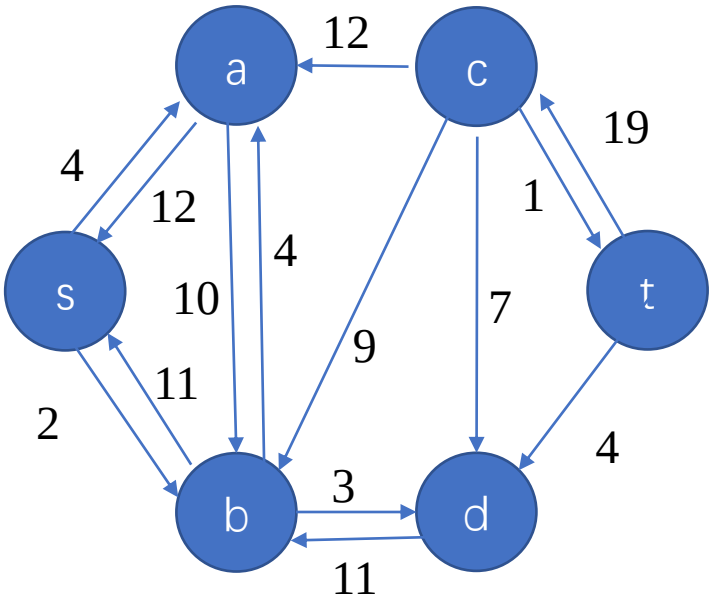
剩余图



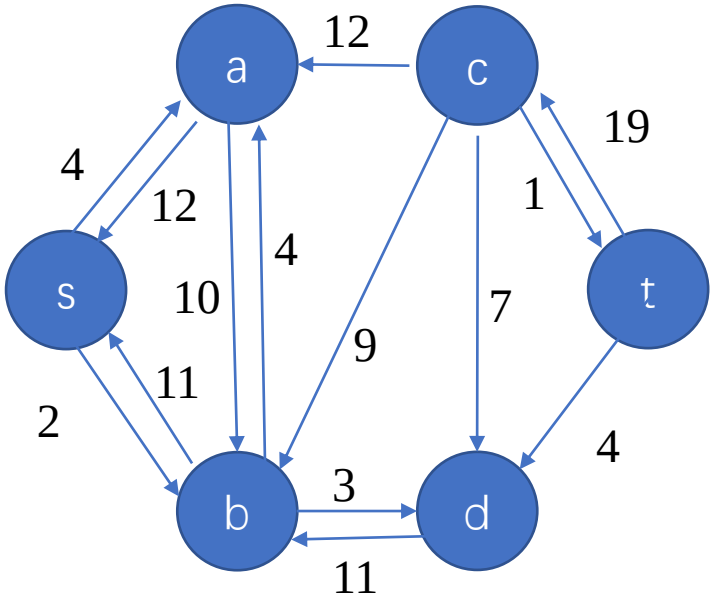
第二层次图



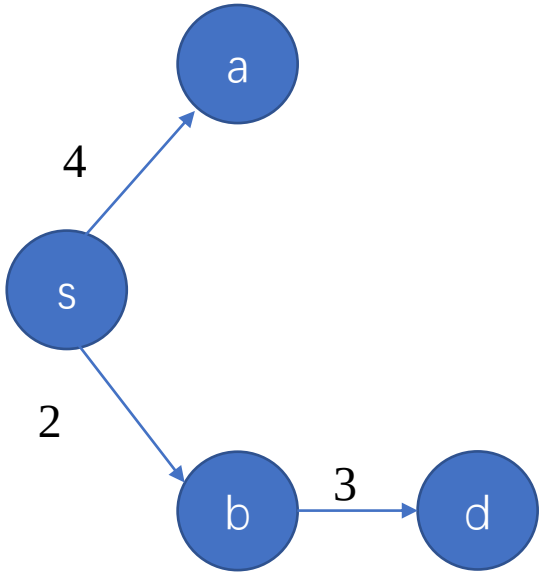
增值s, b, d, c, t



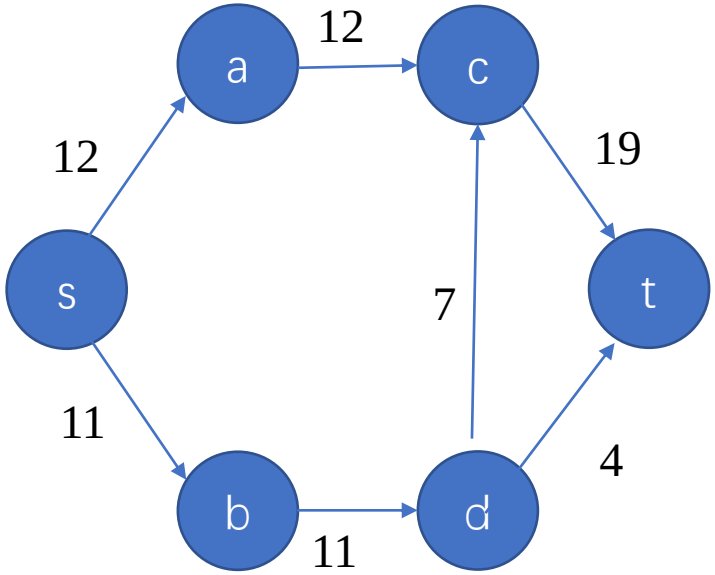
剩余图



剩余图

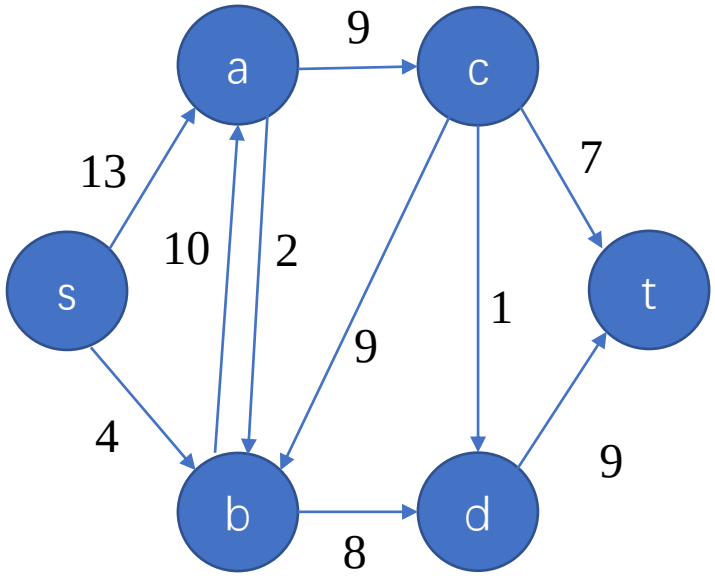


第三层次图



最后的流

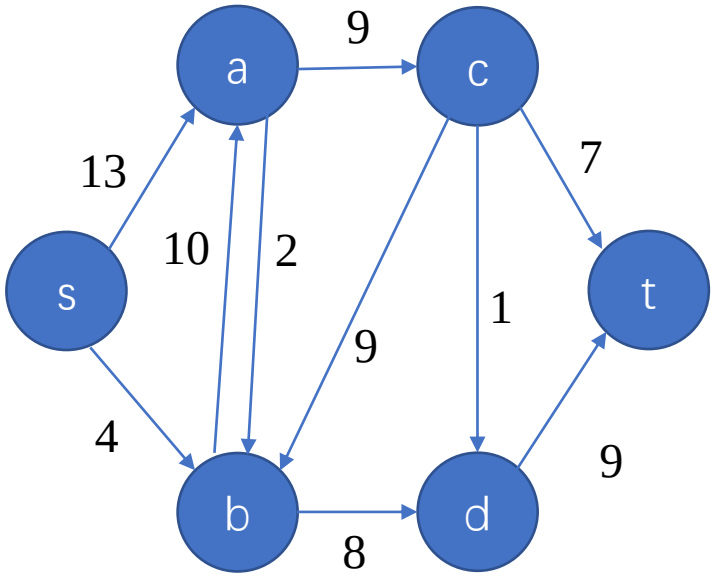
5. 给定如下输入图，采用给定的DINIC算法找到最大流。

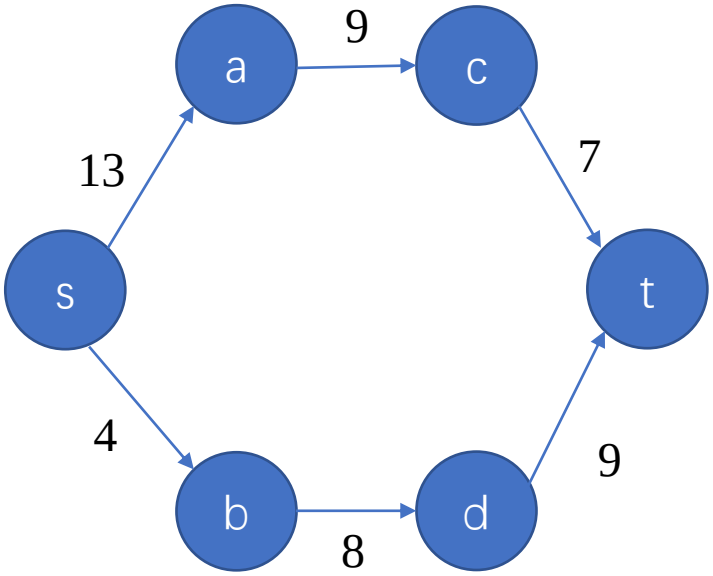


DINIC算法

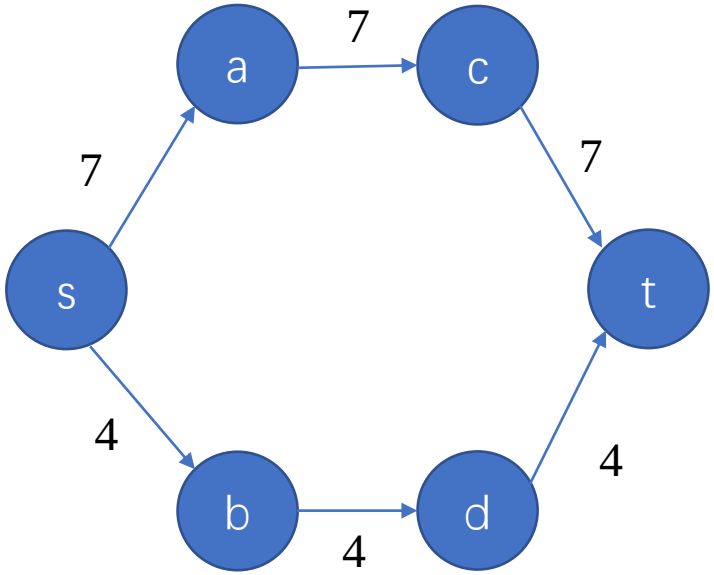
输入：网络（G，s，t，c）
输出：G中的最大流

```
1.  for 每条边（u,v）∈E
2.        f(u,v)←0
3.  end for
4.  初始化剩余图，设R=G
5.  查找R的层次图L
6.  while t 为L中的顶点
7.        u←s
8.        p←u
9.        while outdegree(s) > 0 { 开始阶段}
10.             while u ≠ t
11.                  if outdegree(u) > 0 then {前进}
12.                       设(u,v)为L中的一条边
13.                       p← v
14.                       u← v
15.                  else  {退出}
16.                       删除 u和L中所有的邻接边
17.                       从 p的末尾删除u
18.                       将u设为p中的最后一个顶点
19.                       (u可能是t)
20.                  end if
21. if u = t then {增值}
22.       设△为p中的 瓶颈容量，用
23.       △增值p当前的流，在剩余
24.       图和层次图中调整p的容量，
25.       删除饱和边，设u是p中从s
26.       可到达的最后顶点，注意u
27.       可能是s
28.end if
29.end while
30.从当前剩余图R计算新的层次图L
31.end while
```

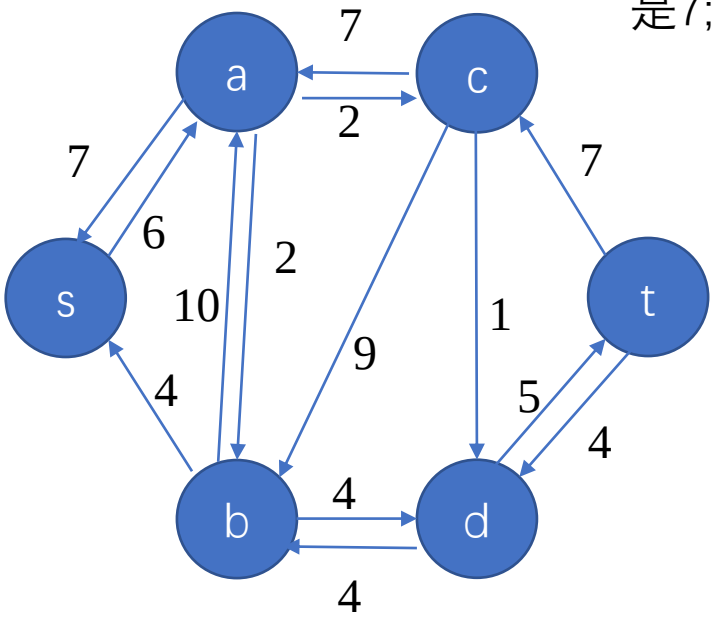




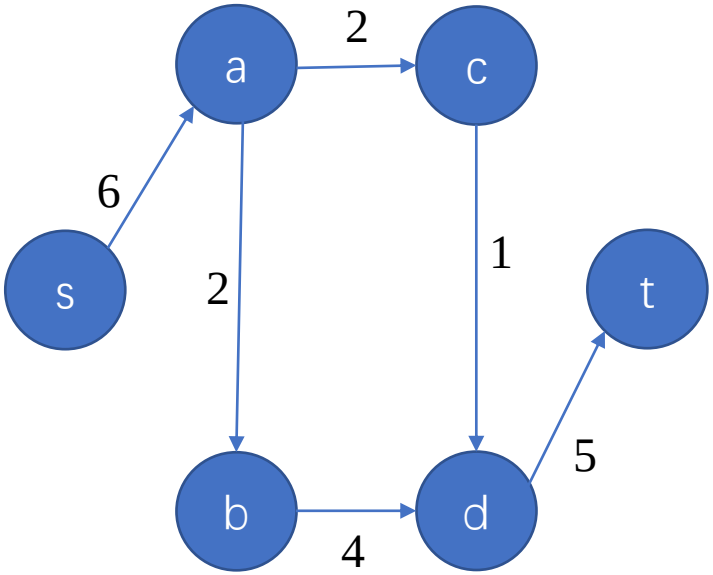
第一层次图



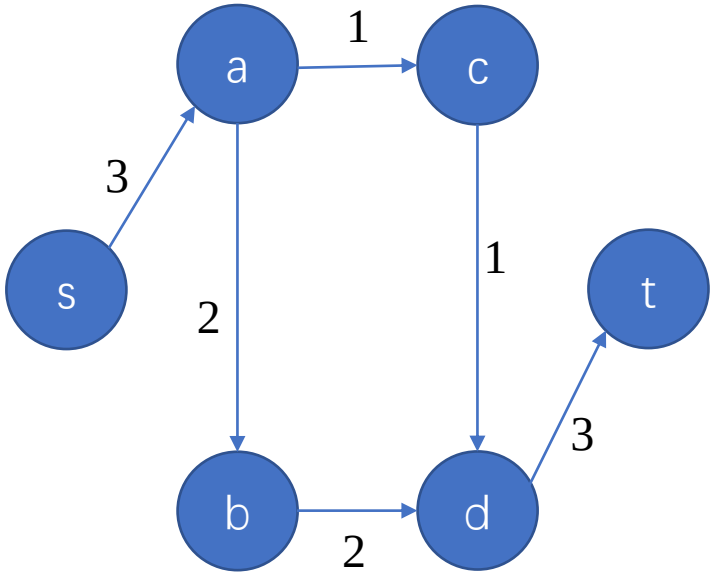
阻塞流:s,a,c,t上的瓶颈容量是7; s,b,d,t上的瓶颈容量为4



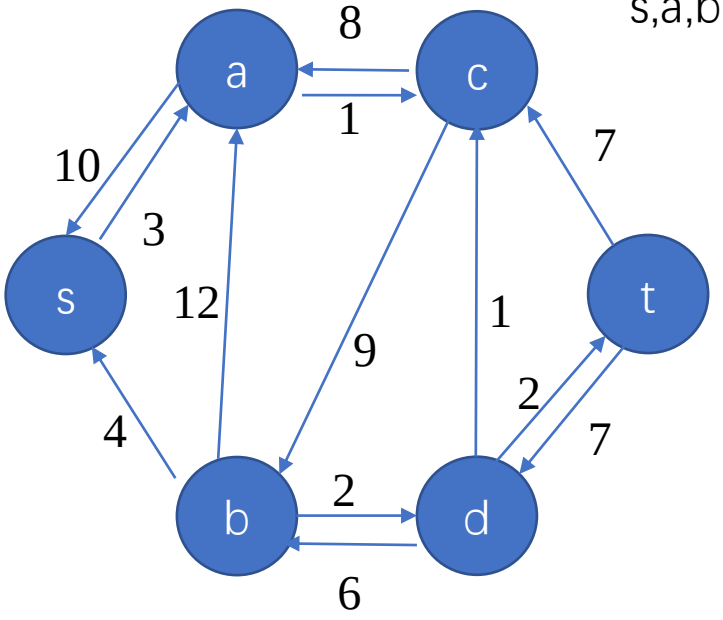
用瓶颈容量更新流，得到剩余图



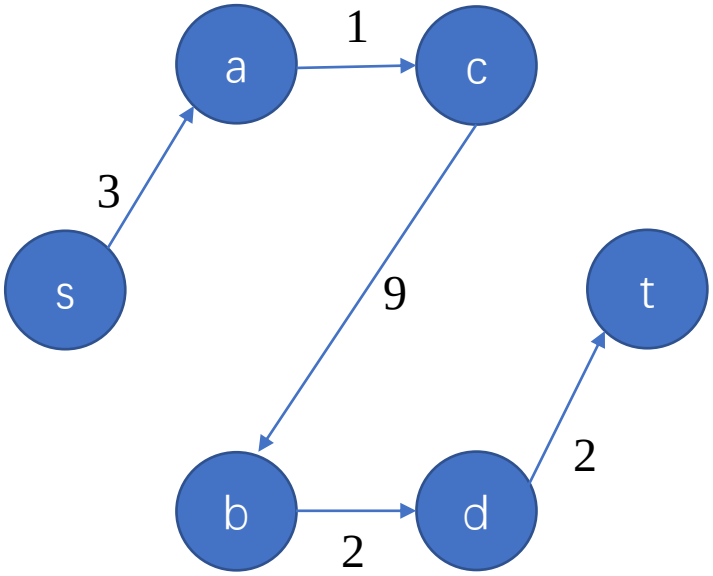
第二层次图



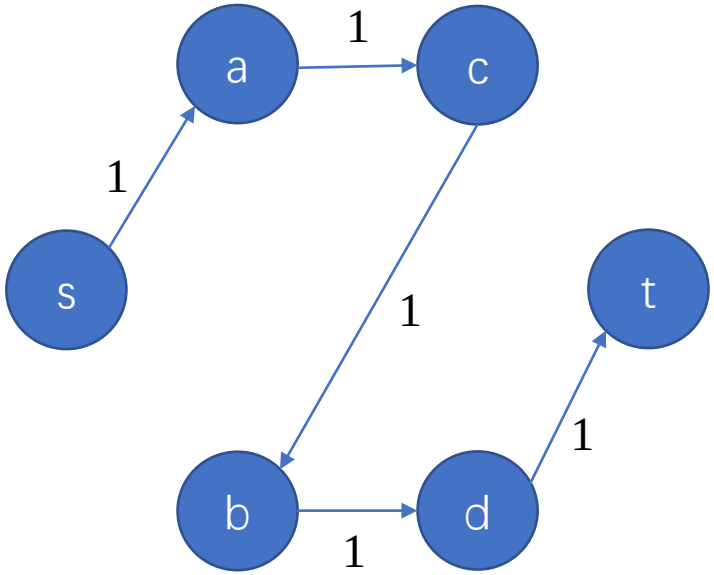
阻塞流:s,a,c,t上的瓶颈容量是1;
s,a,b,d,t上的瓶颈容量为2



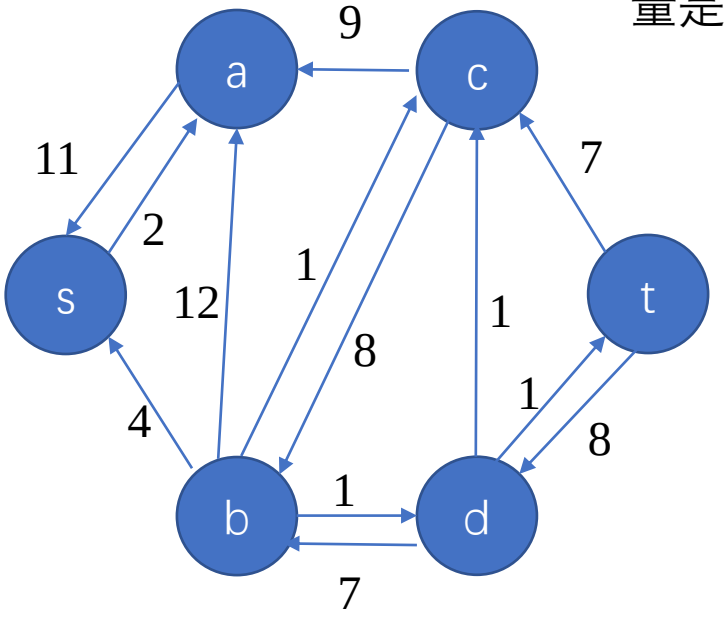
用瓶颈容量更新流，得到剩余图



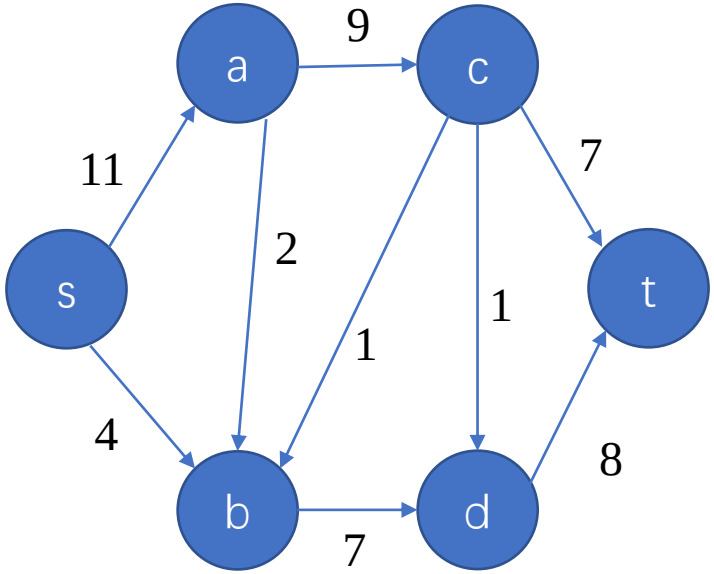
第二层次图



阻塞流:s,a,c,b,d,t上的瓶颈容量是1



用瓶颈容量更新流，得到剩余图



最后的层次图中s不能到达t，因此有最大流