第六章 组合优化问题的有效算法

6.2 最大流问题的有效算法

四、最大流问题的一个多项式算法

算法的基本框架：

初始化：求一个可行流 f

第 1 步：构造 $N(f)$;

第 2 步：构造 $AN(f)$;

第 3 步：寻找 $AN(f)$ 中的极大流 g；

第 4 步：如果 $AN(f)$ 中不存在 $s-t$ 路，当前的流为最大流，算法终止；

否则转第 5 步

第 5 步：增广流 $f := f + g$：

对 $AN(f)$ 中所有的前向弧 (u,v)，置 $f(u,v) := f(u,v) + g(u,v)$

对 $AN(f)$ 中所有的后向弧 (u,v)，置 $f(u,v) := f(u,v) - g(u,v)$

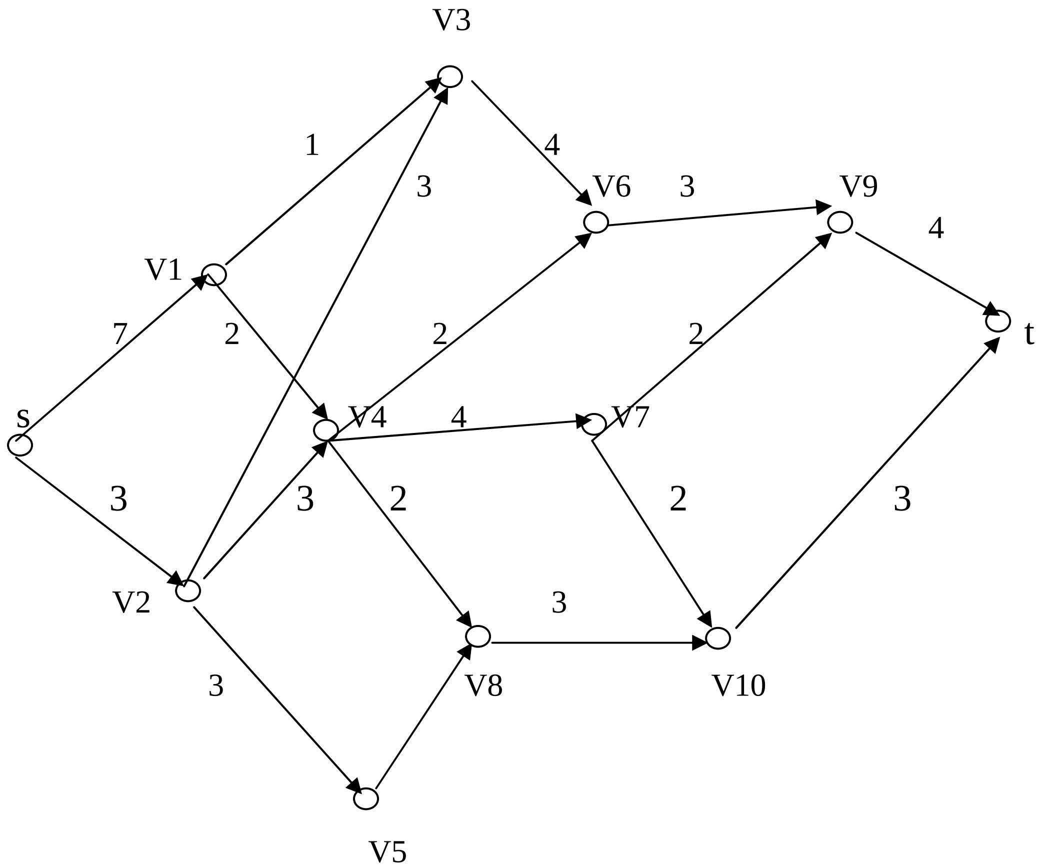
转第 1 步。

算法的主要部分：在分层网络 $AN(f)$ 中寻找极大流 g。

思路：不是沿一条一条路逐步增加流的值，而是让流沿许多路同时增加。

节点v的 throughput：min{进入v的弧之容量和，离开v的弧之容量和}

$v=s$，则为∞　　　$v=t$，则为∞



如果给v_3输送 4 个单位的流，那么它必须将这 4 个单位的流输送个v_6，而 throughput$[v_6]=3$，所以，必须有一部分沿原路返回，就产生了时间上的浪费。

原因：出发点v_3的 throughput 较大，从而在 throughput 较小的节点v_6发生了阻塞。

所以：若从 throughput 最小的节点出发，就不会发生这种现象。

throughput$[v_1]=3$，从 throughput 最小的节点v_1出发，(v_1,v_3)上流分配为 1，(v_1,v_4)上流分配为 2。进入v_3的流必须通过(v_3,v_6)发出去。此时，按深探法，我们能够将这一个单位的流逐步输送到t。但是，不这样做，而是把<u>所有必须通过v_6的流累积起来，进行一次处理</u>。这意味着，<u>在处理一个节点之前，必须把它前面各层次的节点都处理完</u>；或者，等价地，<u>我们从一个节点到另一个节点是按广探法进行的</u>。

如何处理一个节点：如v_1的下一个节点v_4？

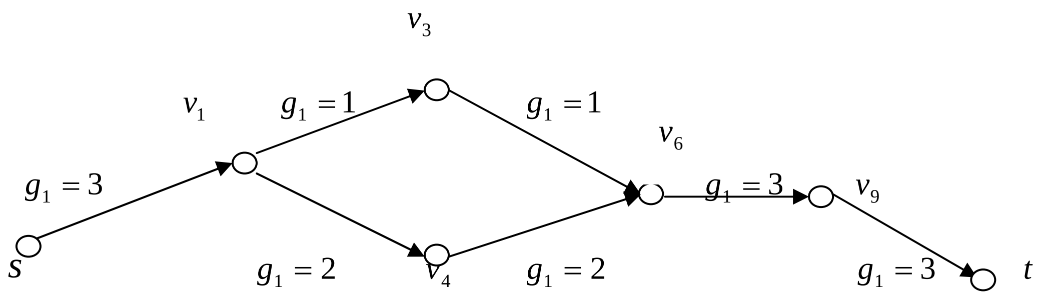
由于v_1有 2 个流送给v_4，所以v_4必须发出 2 个流；

对 v_4 发出去的弧逐一检查并填上流，使其达到容量的上界或者流入 v_4 的流已经发完。

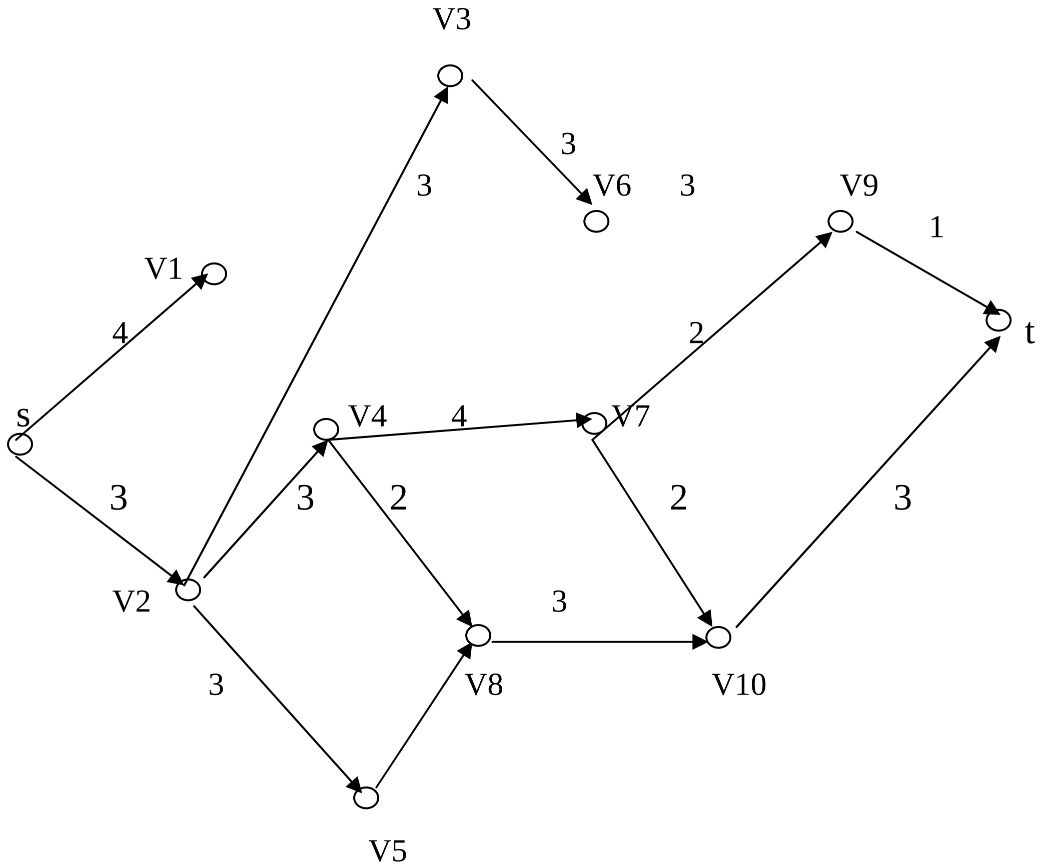
如首先检查 (v_4,v_6)，让其达到其容量 2。所以：从 v_6 向 v_9 发送 3 个流，从 v_9 发出 3 个流到 t。

注： 这一步的整个过程，我们不必担心被处理节点是否有足够的通过能力，这是因为在每一层各节点被分配给流量之和等于 $\text{throughput}[v_1]$。由于 v_1 是 throughput 是最小的节点，所以送到每一个节点的流量，决不会大于该点的通过能力，故不会产生阻塞，从而也就不学需要沿远路返回。

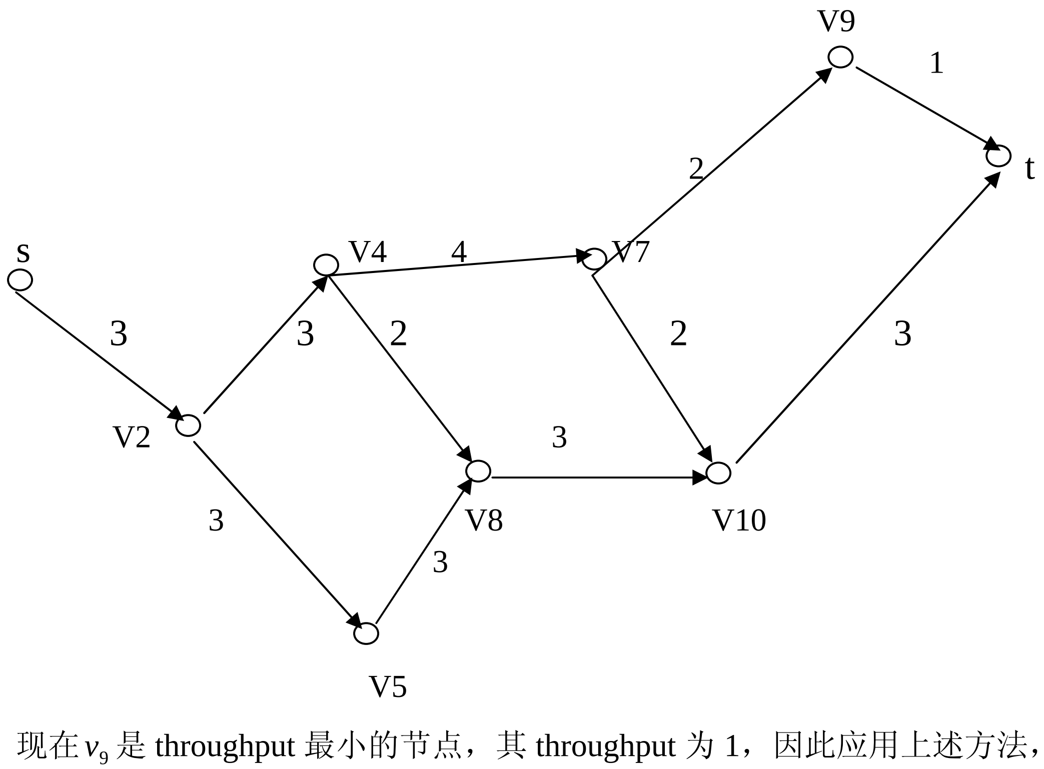
为了完成这一步而得到一个合理的流，我们还需要从 s 出发发送 3 个流到 v_1。与上述过程对称，即自 v_1 沿反向弧“送”3 个流给 s 即可。



此时，g_1 不是极大流，还要重复上述步骤。但是为了反映出已经得到的流 g_1，对弧的容量进行修正，修正后去掉容量为 0 的弧，得到新的分层网络：

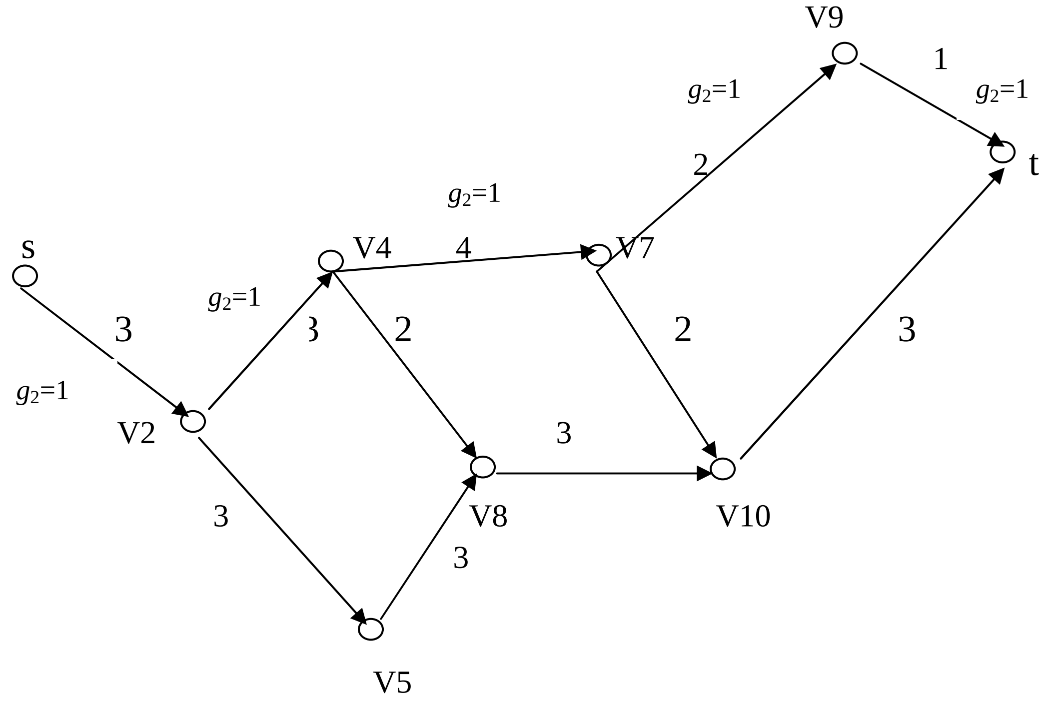


将 throughput 为 0 的节点以及与之关联的弧去掉：

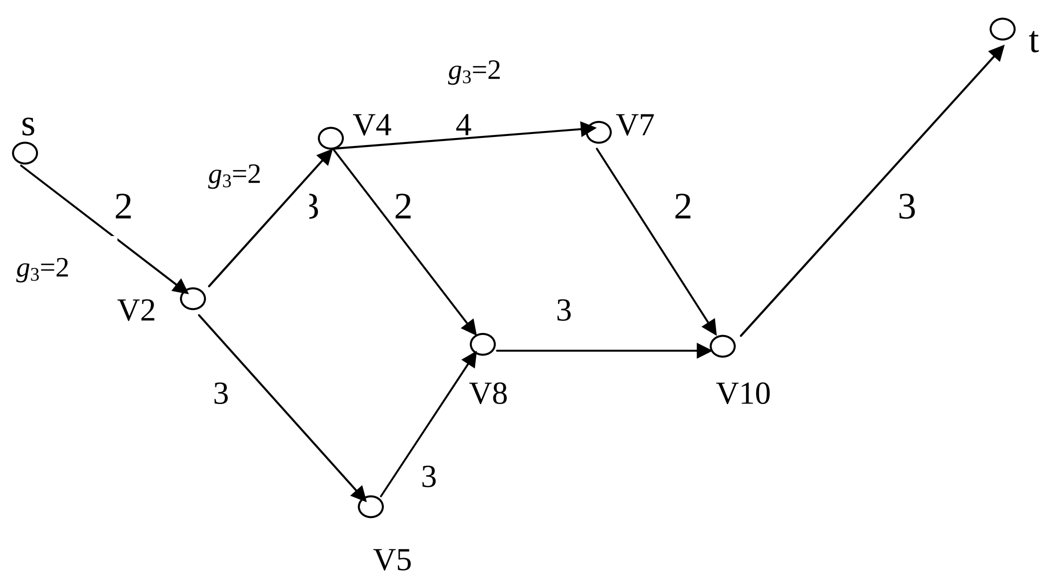


现在 v_9 是 throughput 最小的节点，其 throughput 为 1，因此应用上述方法，可以

找出流 g_2（事实上该流正好是一条增广路）。



然后，修正容量，并且去掉无用的弧和节点。应用上述方法，可以找出流 g_3（事实上该流正好是一条增广路）。



然后，修正容量，并且去掉无用的弧和节点。由于 s 已经去掉，所以我们有极大流 $g = g_1 + g_2 + g_3$，至此，这一阶段已经完成。

整个算法描述如下：
最大流算法：

Input: 网络 $N = (s, t, V, A, b)$

Output: N 的最大流 f

```
begin
   f: = 0, done := "no"; //初始值
   while done := "no" do
   begin //一个新阶段
     g := 0;
     构造辅助网络 AN(f) = (s, t, U, B, ac)
    if 在 AN(f) 中自 s 不能达到 t then done := "yes"
      else repeat
        begin
          while 存在 v 使得 throughput[v] = 0 do
          if v = s 或 t then goto iner
            else 从 AN(f) 中去掉 v 及其关联弧；
              设 v 是 AN(f) 中一个节点使得 throughput[v] 是最小的；//非 0 的
                push (v,throughput[v]);
                 pull (v,throughput[v]);
   iner:  f := f + g
        end
    end
```

procedure push (y, h)//自 y 到 t 把流 g 增加 h 个单位，pull (v, h)程序类似

```
begin
Q := {y};
for all u ∈ U − {y} do req[u] := 0;
req[y] := h; //req[u] 表示自 u 中必须发出的数量
```

while $Q \neq \phi$ do

begin

令v是Q中的一个元素；

从Q中去掉v；//Q中的元素以进入Q的顺序排队

for all $v^{'}$使得$(v, v^{'}) \in B$ and until $\mathrm{req}[v] = 0$ do

begin

$m := \min(ac[v, v^{'}], \mathrm{req}[v]);$

$ac[v, v^{'}] := ac[v, v^{'}] - m;$

if $ac[v, v^{'}] = 0$ then 从B中去掉$(v, v^{'})$；

$\mathrm{req}[v] := \mathrm{req}[v^{'}] - m;$

$\mathrm{req}[v^{'}] := \mathrm{req}[v] + m;$

将$v^{'}$加到Q里；

$g[v, v^{'}] := g[v, v^{'}] + m;$

end

end

end

引理 6.1 在某一阶段，$AN(f)$中一条弧a，仅当$AN(f)$中不存在过a的关于流g的前向增广路时，才把a从B中去掉。

证明：在某一阶段，$AN(f)$中一条弧a，如果被去掉，则必有：或者$g(a) = ac(a)$，或者$a = (u, v)$且u和v至少有一个的 throughput 为 0。

（1） 当$g(a) = ac(a)$时：在$AN(f)$中a只能做为后向弧出现在关于g的增广路上。

（2） 当$a = (u, v)$的一个端点的 throughput 为 0 时，如v，设进入v的弧其容量为 0，因此进入v的一切弧不能在进入v的前向路上。

所以，a不能在任意的前向路上。

证毕

引理 6.2 每一阶段结束时，g 是 $AN(f)$ 中的极大流。

证明：由于，一个阶段的操作是：去掉弧和减少容量，

所以，如果一条弧关于流 g 的前向增广路是无用的，

则在整个这一阶段都是无用的；

因此，本阶段结束时，$AN(f)$ 中不存在关于 g 的前向增广路，

使得它通过此阶段已去掉的弧和节点。

又由于，仅当 s 或 t 被去掉时，本阶段才结束，

所以，一个阶段结束时，$AN(f)$ 中不存在前向增广路，

故，g 是 $AN(f)$ 中的极大流。

引理 6.3 在任意一个阶段，$AN(f+g)$ 中 $s-t$ 的距离严格地大于其前一个阶段 $AN(f)$ 中 $s-t$ 的距离。

证明：显然，辅助网络 $AN(f+g)$ ＝辅助网络 $AN(f)(g)$（$AN(f)$ 关于 g 的辅助网络）。

但是，因为 g 是极大流，所以在 $AN(f)$ 中不存在关于 g 的前向增广路，

所以，所有增广路的长度> $s-t$ 距离，

所以，$AN(f+g)$ 中 $s-t$ 的距离> $AN(f)$ 中的 $s-t$ 的距离。

证毕

定理 6.3 上述算法求解网络 $N=(s,t,V,A,b)$ 的最大流是正确的，其计算复杂度为 $O(|V|^3)$。

证明：正确性：在最后一个阶段，s 和 t 是不连通的，因此 $N(f)$ 中最大流的值为 0。所以，这个最大流值为 $|f^*|-|f|$，其中 $|f^*|$ 为 N 中最大流的值。所以，$|f|=|f^*|$，即算法结束时得到最大流。

复杂性：

（1）迭代步数（阶段数）：至多为 $O(|V|)$。因为，在 $AN(f)$ 中从一个阶段到下一个阶段 $s-t$ 的距离是严格递增的，且不能超过 $|V|$。

（2）在一个阶段的计算量：处理不同弧的总次数 T；记

$$T = T_s + T_p$$

其中，T_s表示在同一个阶段里经过一次处理可达到饱和的弧的数目，

T_p表示在同一个阶段里经过一次处理只得到部分流的弧的数目。

因为：一条弧一旦被饱和，则要去掉，所以每一条弧最多经过一次这样的处理，所以：$T_s = O(|A|)$。

当一条弧经过一次处理未被饱和，则在这一阶段可能需要处理多次。

因为：每处理一个 throughput 最小的节点，最多需要处理$|V|$条这样的弧，

而每一个阶段最多处理$|V|$个这样的节点，

所以：$T_p = O(|V|^2)$。

所以：$T = T_s + T_p = O(|A|) + O(|V|^2) = O(|V|^2$。

所以：整个算法的计算复杂度为$O(|V|^3)$。

证毕

五、具有单位容量的最大流问题的多项式算法

显然：如果弧容量为整数，则经过上述算法每次得到的可行流也都是整数。所以，如果弧容量为 1，那么，每一弧上的流值为 1 或 0，并且在辅助网络中每次增广流的值为 1。

单位容量下的最大流算法，具有很好的性能。

引理 6.4 上述最大流算法应用于单位容量网络$N = (s,t,V,A)$时，算法的每个阶段的计算复杂度为$O(|A|)$。

证明：每一条增广路上的弧容量为 1，每一条弧处理一次就成为饱和弧。

算法所需要的阶段数目也可以改进。

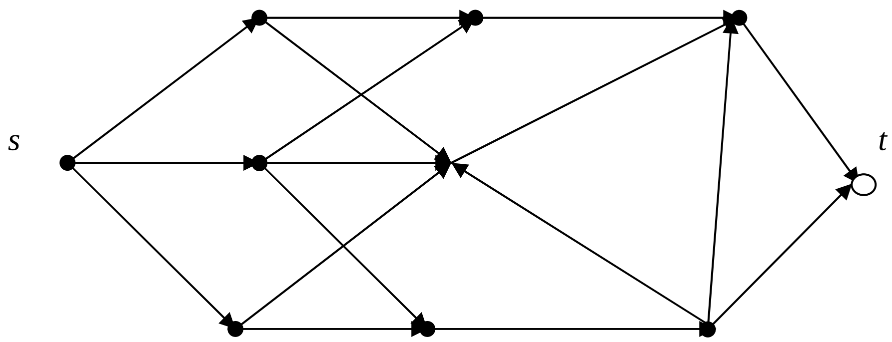
引理 6.5 在具有单位容量网络$N = (s,t,V,A)$中，$s-t$距离不可能大于$2|V|\sqrt{|f^*|}$，其中$|f^*|$为最大流的值。

定理 6.3 对于具有单位容量网络$N = (s,t,V,A)$，上述算法的计算复杂度为$O\left(|V|^{\frac{2}{3}}|A|\right)$。

六、简单网络最大流问题的多项式算法

当一个网络除弧容量为 1 外，它还有某些特殊结构时，上述最大流算法的计算复杂度可能会更好。例如，对于简单网络。

定义 6.2：具有单位容量网络 $N=(s,t,V,A)$，它的节点的或者入次为 0 或 1，或者出次为 0 或 1，则称这样的网络为简单网络。



引理 6.6 在一个简单网络中，$s-t$ 距离不可能大于 $\frac{|V|}{|f^*|}$，其中 $|f^*|$ 为最大流的值。

定理 6.4 对于简单网络，上述算法的计算复杂度为 $O\left(|V|^{\frac{1}{2}}|A|\right)$。