红黑树课堂

专业的海外IT辅导机构

扫码添加小助手
获取更多内容

扫码关注B站
了解更多课程

网络流算法

Flow Network

一个有向图G，每个边都有一个正数的容量。

有两个特殊的顶点，source s和 sink t，没有箭头指向s，没有箭头从t中指出。
（s是源头，t是最终水流流入的水箱）

G中的flow是一个函数 $f : E \to \mathbb{R}^+$, $f(u, v) \geq 0$，满足：

1. Capacity constraint: for all edges e(u, v) ∈ E we require

   f(u,v) ≤ c(u,v).

2. Flow conservation: For all v ∈ V − {s, t} we require

流入值 流出值

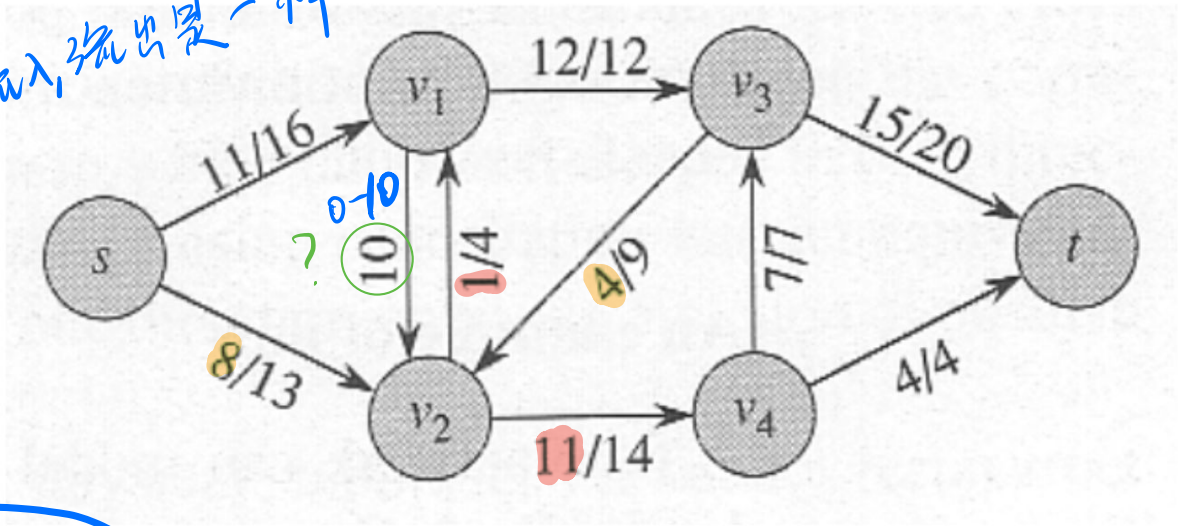
$$\sum_{(u,v)\in E} f(u,v) = \sum_{(v,w)\in E} f(v,w).$$

value of flow: $|f| = \sum_{v:(s,v)\in E} f(s,v).$

$$|f| = \sum_{v:(v,t)\in E} f(v,t).$$

例子：

除去s，t，其他点，流入，流出是一样的

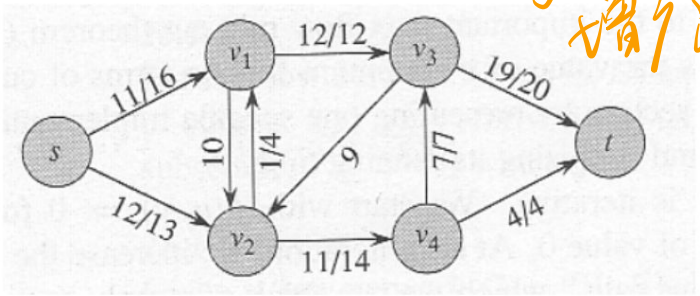


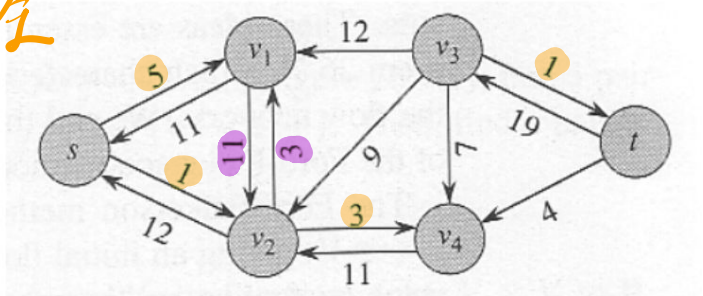
通过的流量/最大容量

Residual Flow Network

另一种表示形式
把原本通过的流量用一根相反的箭头标出。
把原本剩余的流量用一根和之前相同方向的箭头标出。
通过建立argmenting path 可以增加总通过量

增广路径



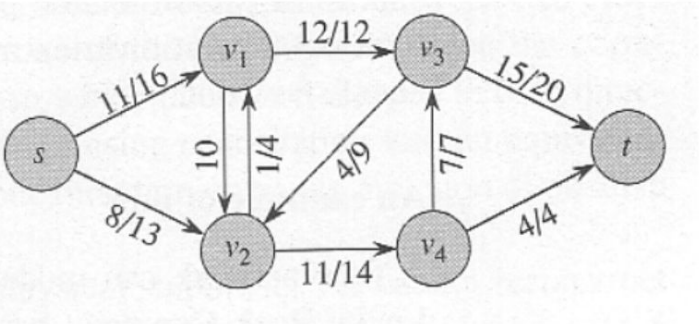


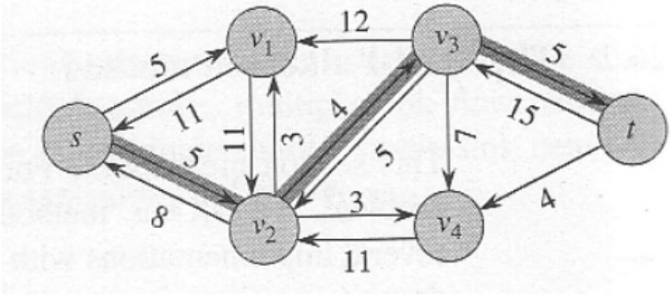
Augmenting Path

augmenting path的最大通过量为一条路径上的“瓶颈”边的容量大小。

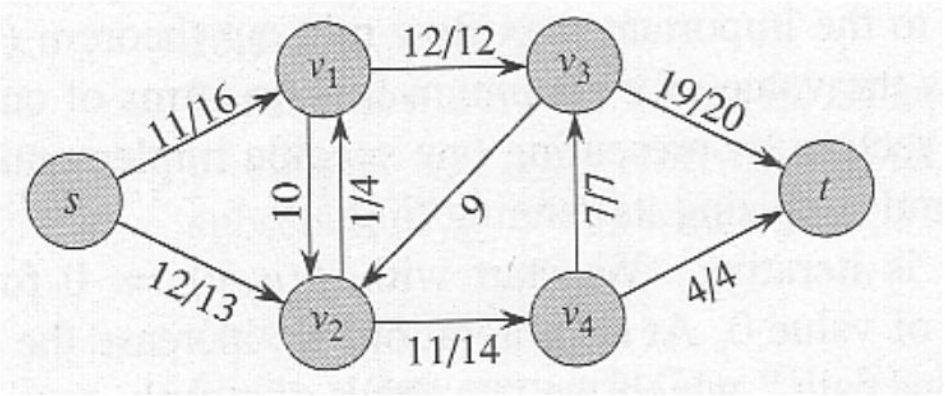
i.e., the capacity of the smallest capacity edge on that path.

现在，我们可以重新计算通过增强路径所有边的流量，如果通过增强路径的流量与原始流量的方向相同，则添加通过路径的额外流量，如果方向相反，则减去



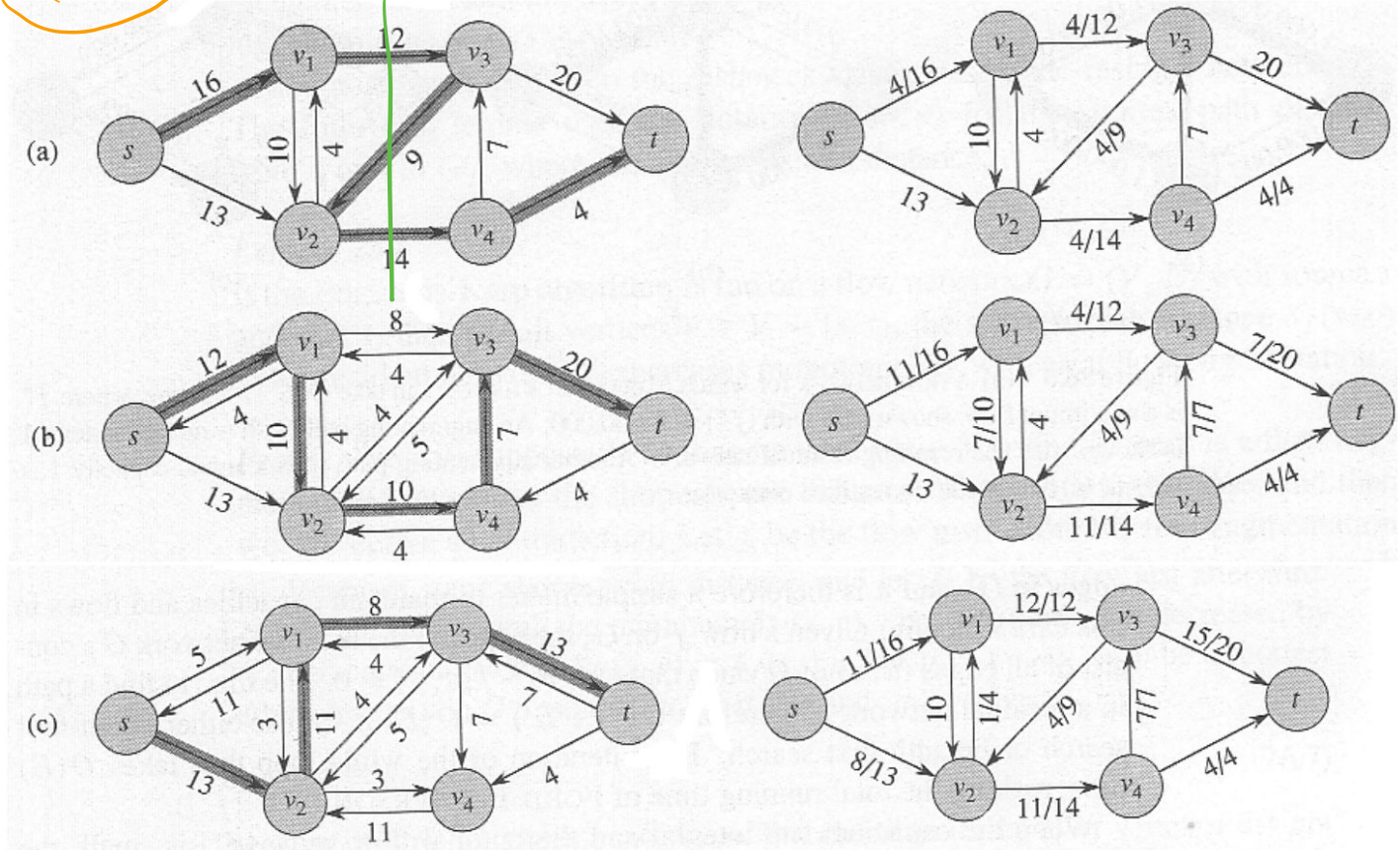


增加通过量之后的图:

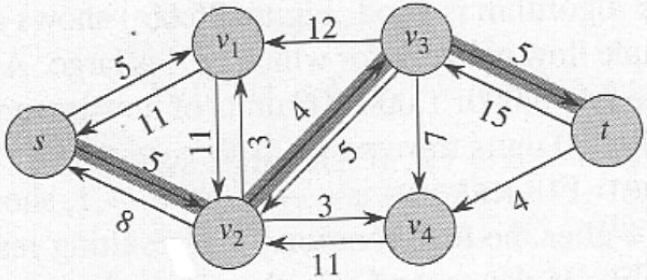


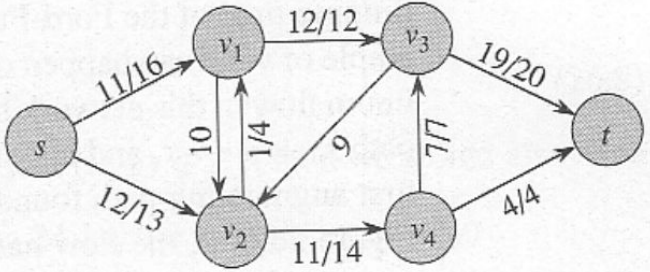
Ford Fulkerson method找最大流

1. 尽可能增加augmenting path
2. 当没有可加的augmenting path时，为最大流量。

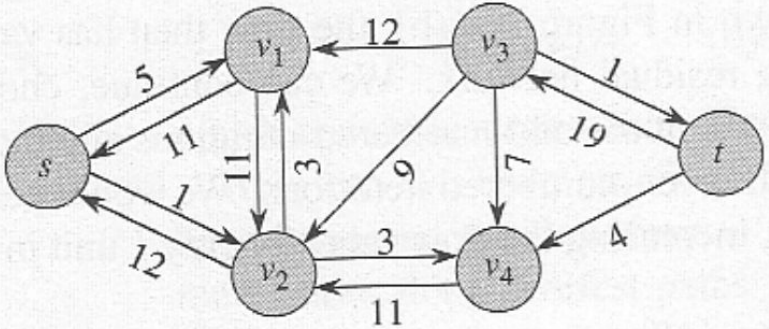

(a)
s
v_1
v_2
v_3
v_4
t
16
13
10
4
12
9
7
20
4
14
4/12
4/16
4/9
4/14
4/4
(b)
8
12
4
20
5
10
(c)
11
3
11/16
12/12
15/20
8/13
11/14

(d)


v_1
v_3
s
t
v_2
v_4
12
5
5
11
11
3
4
5
7
15
5
8
3
4
11


v_1
12/12
v_3
11/16
19/20
s
t
10
1/4
9
7/7
12/13
v_2
v_4
4/4
11/14

(e)


v_1
12
v_3
5
1
11
19
s
t
11
3
9
7
1
12
v_2
3
v_4
4
11

为什么这个过程可停止？
为什么我们不能陷入循环中，而循环不断地添加扩展路径呢？

如果所有容量都是整数，则每个扩充路径都会增加至少1个单位通过网络的流量；
总流量不能大于离开源的所有边的总容量之和，因此最终必须终止该过程。

如何证明我们可以得到最大解？证明将基于网络流的最小割。

Cut

流网络中的割线是将基础图的顶点分为两个子集S和T的任意分区，从而：

1 S∪T=V

2 S∩T=∅

3 s∈S and t∈T.

切口（S，T）的容量c（S，T）是离开S并进入T的所有边的容量之和，即

$$c(S,T) = \sum_{(u,v)\in E} \{c(u,v) : u \in S \ \& \ v \in T\}$$

Flow through a cut

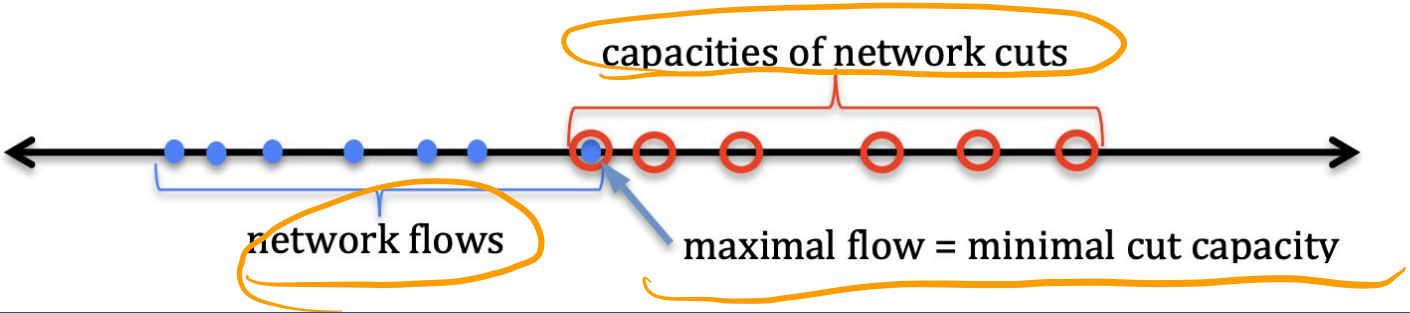
通过切口f（S，T）的流量是从S到T的通过边的总流量减去从T到S的通过边的总流量：

$$f(S,T) = \sum_{(u.v)\in E} \{f(u,v) : u \in S \ \& \ v \in T\} - \sum_{(u.v)\in E} \{f(u,v) : u \in T \ \& \ v \in S\}$$

显然 f (S, T) ≤ c(S, T ） 因为对任意 (u, v) ∈ E我们假设f(u,v) ≤ c(u,v), 并且 f(u,v) ≥ 0.

最大流 = 最小割

定理:流网络中的最大流量等于最小容量切割(cut)的容量。
由于任何流量都必须穿过每个切口，因此任何流量都必须小于任何切口的容量:f = f(S, T)≤c(S, T)。
因此，如果我们发现流量f等于某个切口的容量(S, T)，则该流量必须最大，而该切口的容量必须最小。



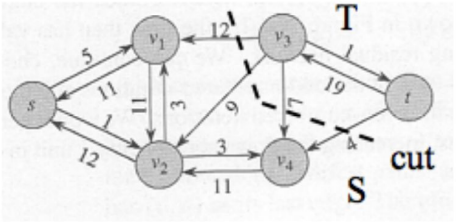
现在，我们证明了这个算法会停止且正确。

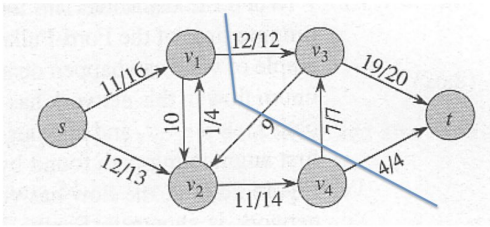
假设福特-富尔克森（Ford Fulkerson）算法已终止，并且在最后的剩余网络流中没有从源s到宿t的更多增广路径。

将S定义为源s和所有顶点u，以便在残留网络流（residual network flow）中有一条路径从源s到该顶点u。

将T定义为没有此类路径的所有顶点的集合。

由于不再有从s到t的增广路径（ augmenting paths），因此汇点t显然属于T。





复杂度？

Ford-Fulkerson算法可能会在时间上与最大流量的值成比例地运行，该最大流量的值可以与输入大小成指数关系。

Edmonds-Karp Max Flow Algorithm

Edmonds-Karp算法以一种简单的方式改进了Ford Fulkerson算法：始终选择从源s到接收器t的最短路径，其中“最短路径”意味着边的数量最少，而不论它们的容量（即每个边缘具有相同的单位重量）。

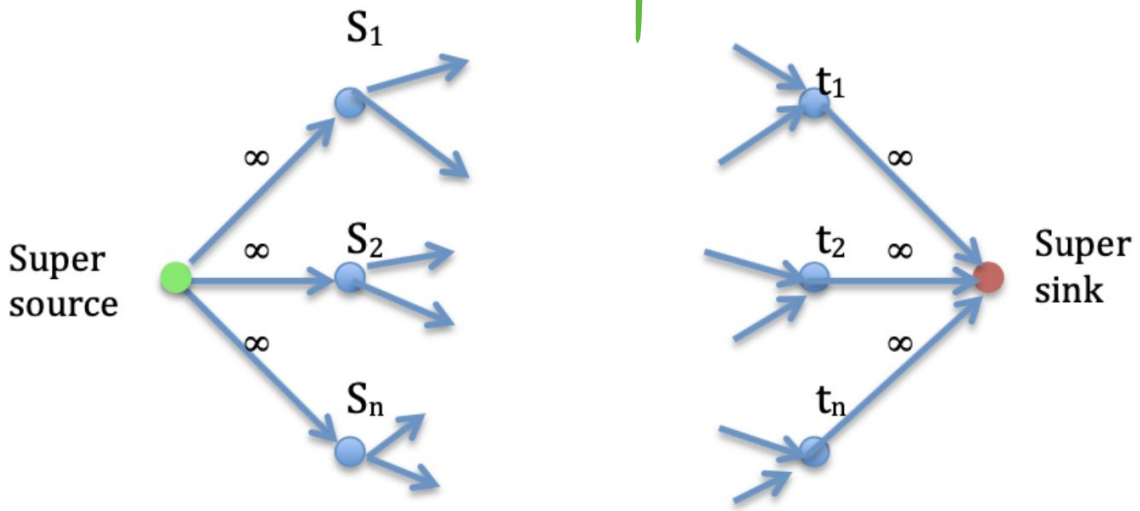
请注意，这种选择在某种程度上与直觉相反：我们最好选择容量较小的边缘，而不是容量较大的边缘，只要它们沿着从s到t的最短路径即可。

为什么这样的选择会加快福特-Fulkerson算法¥的速度？要看到这一点，需要一个棘手的数学证明，**请参阅教科书**。可以证明这种算法在时间$O(|V||E|^2)$上运行。

迄今为止最快的最大流量算法，是Preflow-Push 的扩展算法，运行时间在| V |^3。

Networks with multiple sources and sinks

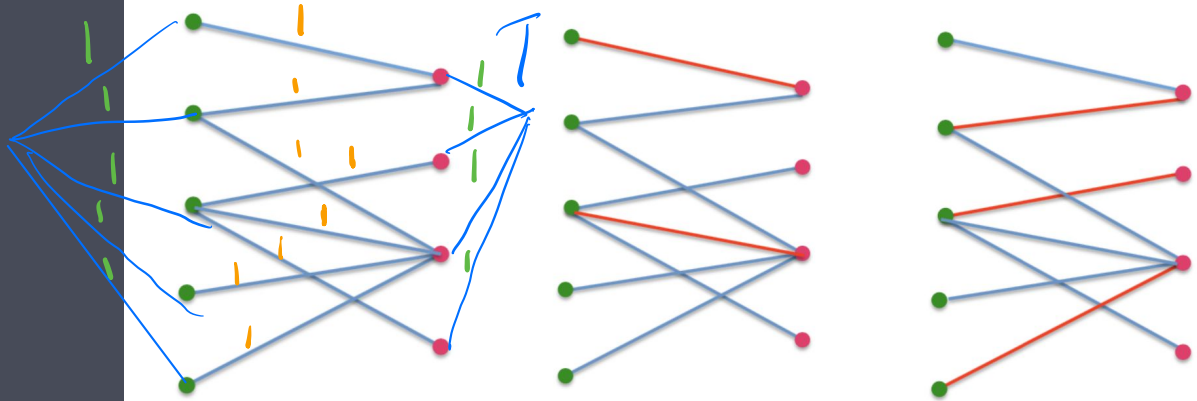
通过添加“超级接收器”（super-sink）和“超级源”（“super-source”），并通过无限容量的边缘将它们分别连接到所有源和接收器，可以将具有多个源和接收器的流网络简化为具有单个源和单个接收器的网络。



Maximum matching in bipartite graphs

我们将考虑二分图；即图的顶点可以分为两个子集L和R，使得每个边e∈E在集合L中具有一端，在集合R中具有另一端

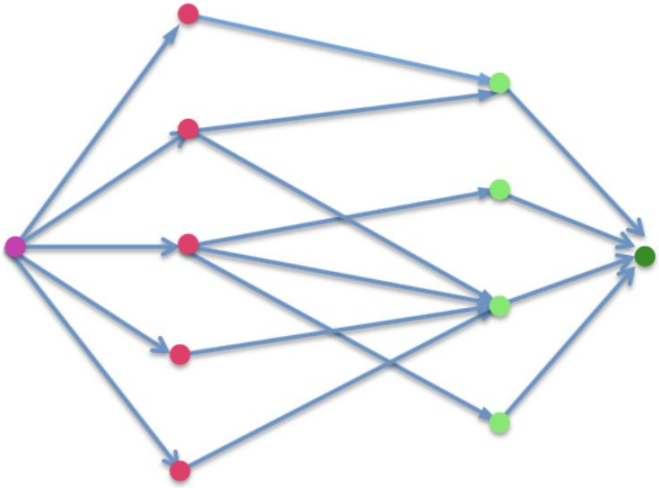
图G中的匹配是所有边E的子集M，使得图的每个顶点最多属于匹配M中的一个边。

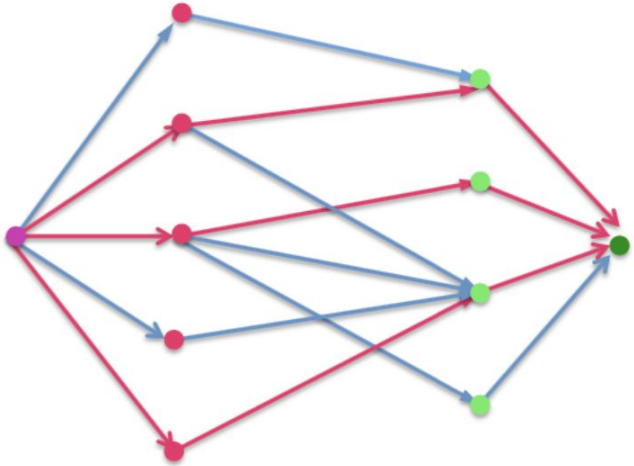


二分图G中的最大匹配是包含尽可能多的边的匹配。

通过添加超级源和超级接收器，并将所有边的容量设置为1，我们将“最大匹配”问题转变为“最大流”问题。

通过添加超级源和超级接收器，并将所有边的容量设置为1，我们将“最大匹配”问题转变为“最大流”问题。





Max Flow with vertex capacities

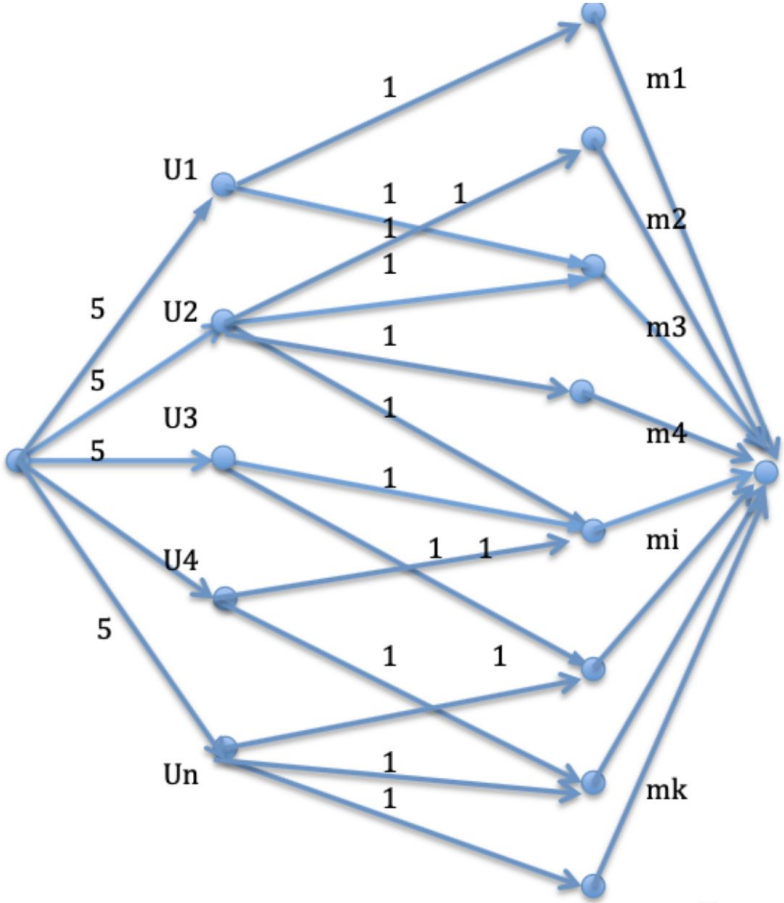
有时，流程图的不仅边而且顶点vi都可能具有容量C（vi），这限制了到达顶点（并也限制了离开顶点）的流的总吞吐量：

$$\sum_{e(u,v)\in E} f(u,v) = \sum_{e(v,w)\in E} f(v,w) \leq C(v)$$

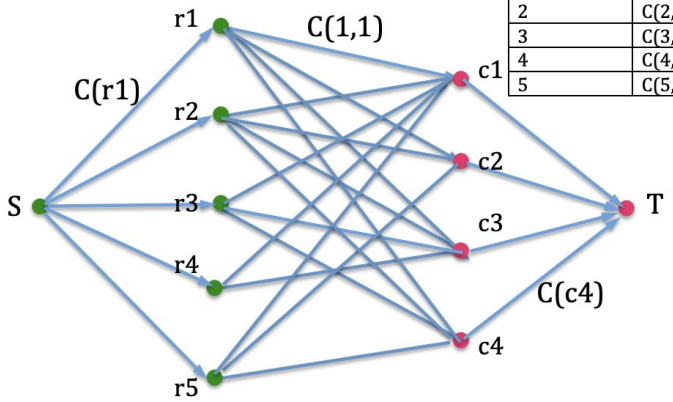
通过将具有有限容量C（v）的每个顶点v分为两个顶点vin和vout，从而使所有进入v的边缘都进入vin，所有离开v的边缘现在都离开vout，将这种情况简化为只有边缘具有容量的情况连接新顶点vin和vout的边e_* =（vin，vout），其容量等于原始顶点v的容量：

Applications of Max Flow algorithm

假设您有一家电影租赁公司。目前，您有k部电影库存，其中有i部电影的mi副本。n位客户中的每位客户一次最多可以租借5部电影。客户已将他们的喜好发送给您，这些喜好是他们想要看的电影列表。您的目标是分发尽可能多的电影



- The storage space of a ship is in the form of a rectangular grid of cells with n rows and m columns. Some of the cells are taken by support pillars and cannot be used for storage, so they have 0 capacity. You are given the capacity of every cell; cell in row r_i and column c_j has capacity $C(i, j)$. To ensure the stability of the ship, the total weight in each row r_i must not exceed $C(r_i)$ and the total weight in each column c_j must not exceed $C(c_j)$. Find how to allocate the cargo weight to each cell to maximise to total load without exceeding the limits per column, limits per row and limits per available cell.



row\column	1	2	3	4
1	C(1,1)	C(1,2)	C(1,3)	C(1,4)
2	C(2,1)	C(2,2)	C(2,3)	C(2,4)
3	C(3,1)	C(3,2)	0	C(3,4)
4	C(4,1)	0	C(4,3)	0
5	C(5,1)	C(5,2)	0	C(5,4)

row	capacity
r1	C(r1)
r2	C(r2)
r3	C(r3)
r4	C(r4)
r5	C(r5)

column	capacity
c1	C(c1)
c2	C(c2)
c3	C(c3)
c4	C(c4)