Network Flow

Load balance

假设有n个任务和m台机器，对每一个任务可由两台机器中的一个来执行

建图：

把每一个任务和对应的两个机器建立连边(每条边 容量为1)

设一个源点S指向n个任务(每条边容量为1)

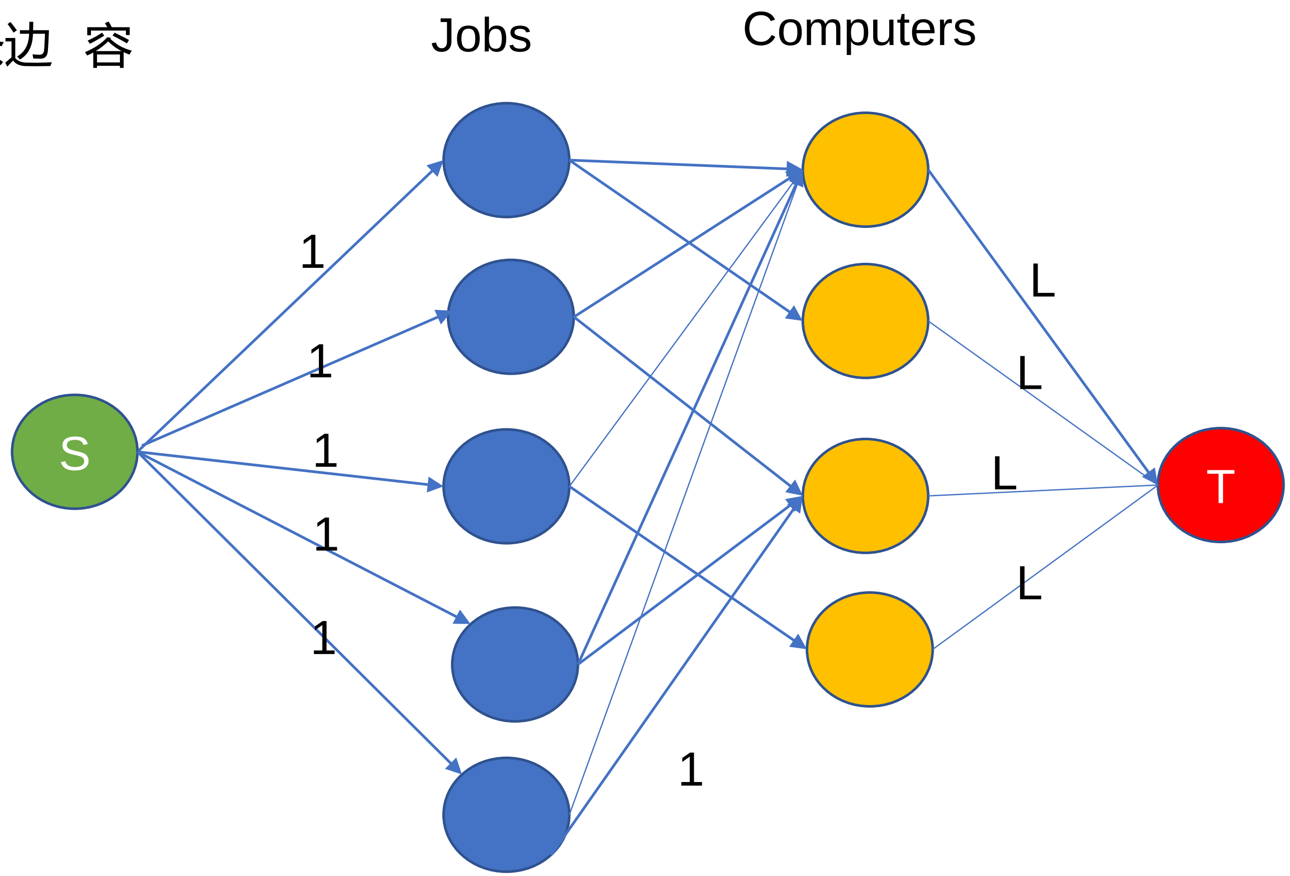
设m台机器指向一个汇点T(每条边容量为L)

求解:

最小的最大负载，因为最大负载是单调的，所以我们可以二分最大负载

时间复杂度:

$O(n^3 \log n)$



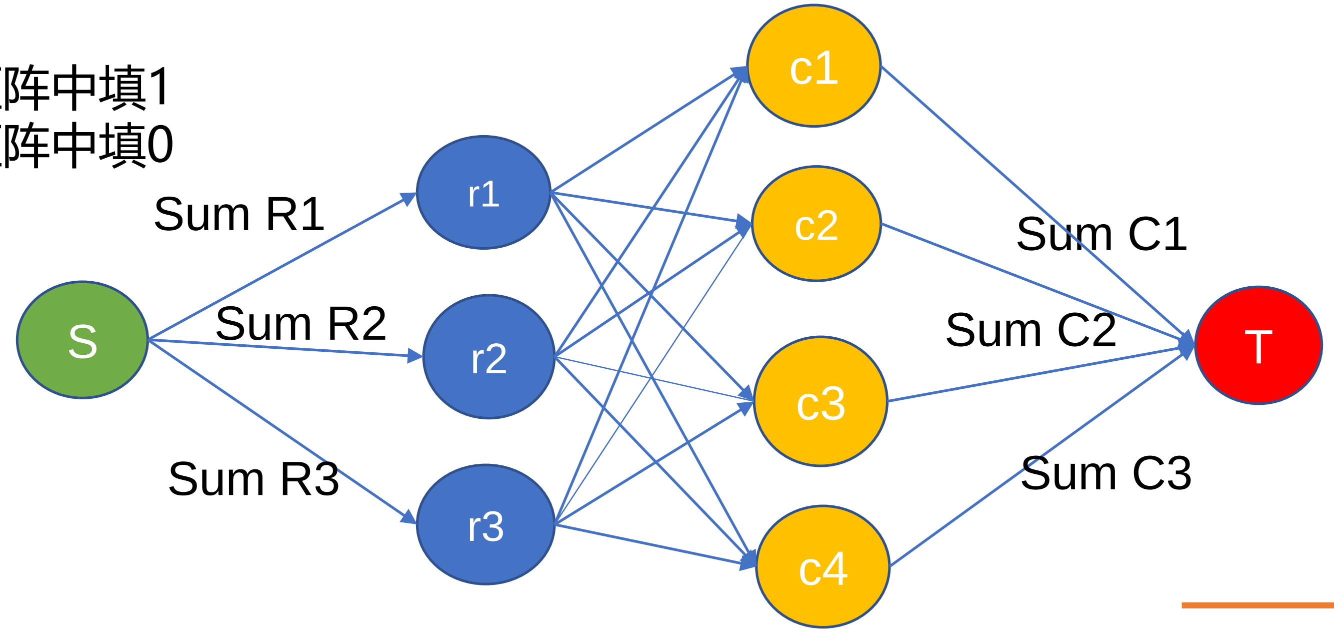
Matrix

假设有n行和m列
建图：
每一行作为一个结点，n个行结点，
每一列也作为一个结点，m个列结点，n个行结点依次和m个列结点相连(共m*n条边)，容量为1

设一个源点S指向n个行结点，容量为对应的行和
设一个汇点T，把m个列结点连接到T，容量为对应列和

求解：
最大流
若第i个行结点到第j和列结点流量为1，则在矩阵中填1
若第i个行结点到第j和列结点流量为0，则在矩阵中填0

	c1	c2	c3	c4
r1				
r2				
r3				



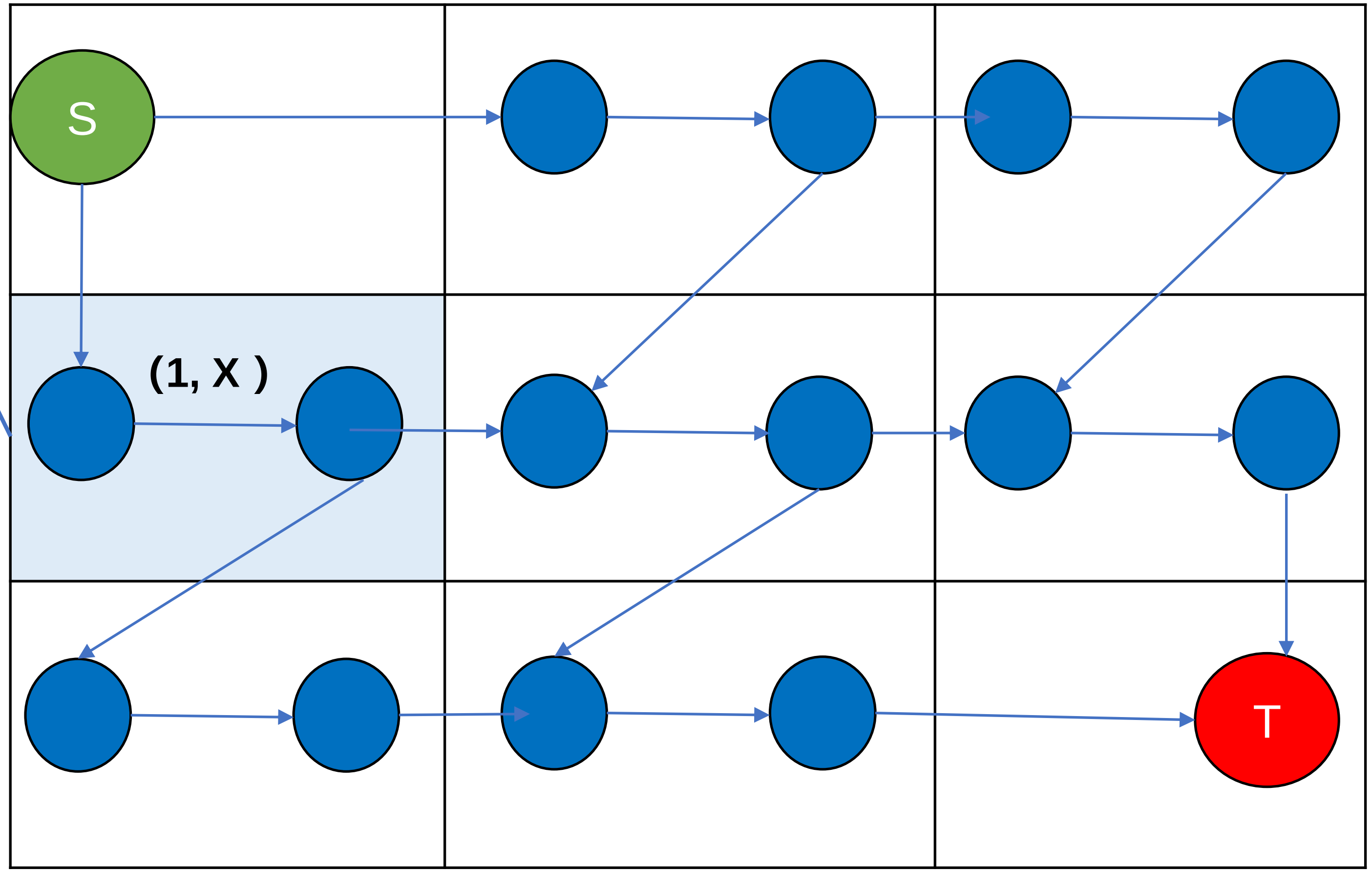
时间复杂度:
$(n+m+2)$ 个点
$(n+m+nm)$ 个边
使用Dinic’s 算法时间复杂度是
$O((n+m+2)^2(n+m+nm)) = O(n^4)$

Problem Reduction

假设有n行和m列

建图：

把矩阵的每个方格**拆为两个点**，
点与点之间的边**容量**为1，
权值 X 为原有的格子的花费。



求解：

从顶点S到顶点T，找两条没有公共边的路径，求流量为2的最小费用流

时间复杂度:

如果使用Bellman Ford算法，时间复杂度为O(FVE)
如果使用Dijkstra算法 ，时间复杂度为O(FElogV)

Network Cost

c必须是整数而且比较小才能做

建图:

每条边的花费不是固定的，而是与经过的流量相关，不能简单的使用最小费用流。

对于每一条边由于容量不唯一，所以在流量不同的时候对应着不同的花费，我们可以将容量修改为1，这样花费随之固定。

将平方拆分为奇数之和，每当流量增加1，选择一条比上条大一点的边，随之的花费也会增加，这个拆分一定是存在的。

$$a + 3a + \cdots + (2n - 1)a = \frac{(a + (2n - 1)a) * n}{2} = an^2$$

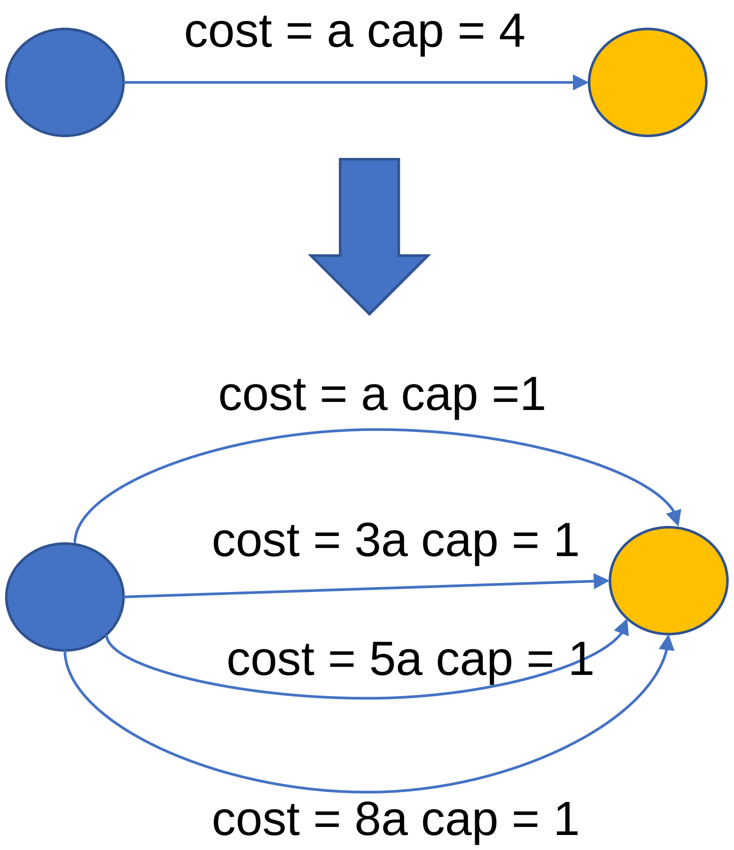
求解:

最小费用流

时间复杂度:

如果使用Bellman Ford算法，时间复杂度为O(FVE)

如果使用Dijkstra算法，时间复杂度为O(FElogV)



Choose Numbers

最小权之配集
假设有n行和m列
建图：

把矩阵里面的元素分为奇数和偶数的集合

建立一个二分图，把源点S到奇点集合，偶数点集合到汇点T的容量设为M_{ij}。

在奇点和偶点之间的边的容量为INF

1	2	3
4	5	6
7	8	9

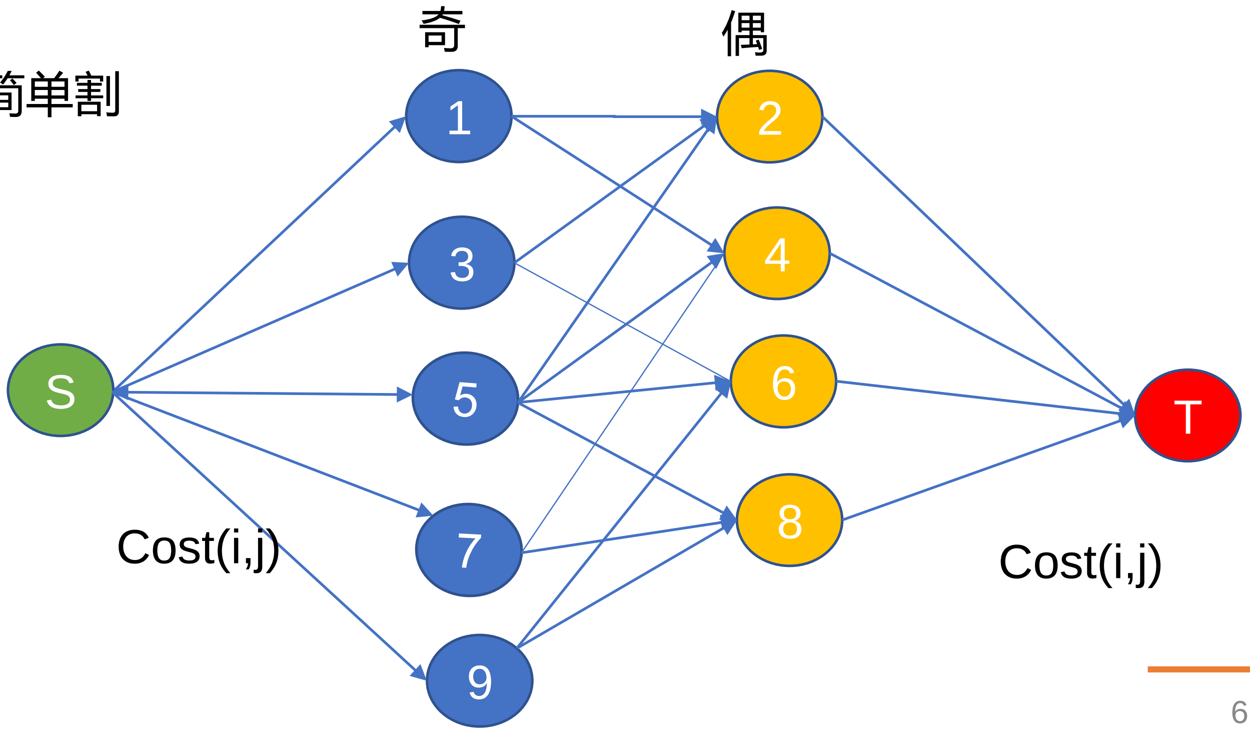
求解：

求最小割，中间权值都是**INF**，所以一定是一个简单割

最小割保证了

1，保证相邻至少选一个

2，求出解就是可行解



时间复杂度：

$(nm + 2)$ 个点

$(n + m + 3nm)$ 个边

MaxFlow((nm + 2，n + m + 3nm))

如果使用Dinic’s 算法时间复杂度是

$O((nm+2)^2(n+m+3nm)) = O(n^6)$

Maximum Weight Subgraph

最大权闭合子图
假设有n个点和m个边
建图：
　　左边给的点代表边集，建立源点S到左边点边，容量是对应的边权
　　右边给的点代表点集，建立右边点到汇点T的边，容量是对应的点权

　　点集结点和边集结点之间的边的容量为INF

求解：
　　用最大流求解最大权闭合子图
　　最后的解就是S割集，边权-点权

时间复杂度：
　　$(n+m+2)$ 个点
　　$(3n+m)$ 个边
　　MaxFlow((n + m + 2，3n + m))
　　如果使用Dinic’s 算法时间复杂度是
　　$O((n+m+2)^2(3n+m)) = O(n^3)$