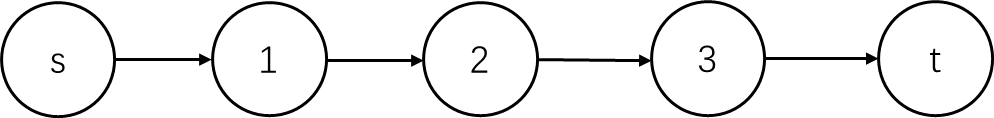
1. **剩余网络与原网络的可达性**
   1. 原图G中可达的节点，在更新G’的过程中有可能变得不再可达。

比如汇点t，在更新结束后一定不再可达；

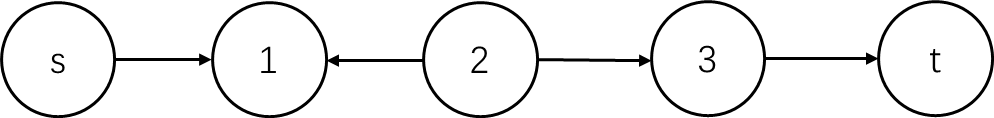
* 1. G’中可达的节点，G中一定可达。

证明：

假设G’如下



我们假设2号节点是第一个G中无法达到的节点，则原图G应如下：



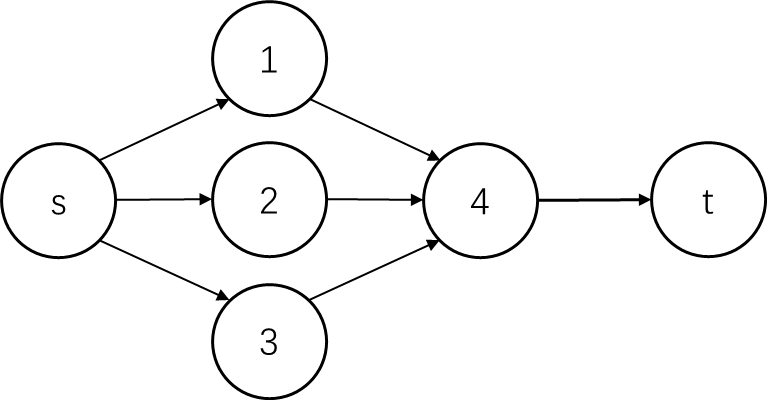
我们将G’中1到2的边称为e’，则e’在原图中的e一定是一条正向边，而在G’中变为了反向边。由于我们在G’中将e更新成了e’，说明e一定在一条增广路径上，所以节点2是该增广路径上的一个结点，所以节点2也是s可达的。

综上，G’与G相比，可达性只会变差。

1. **最大流与最小割的唯一性**

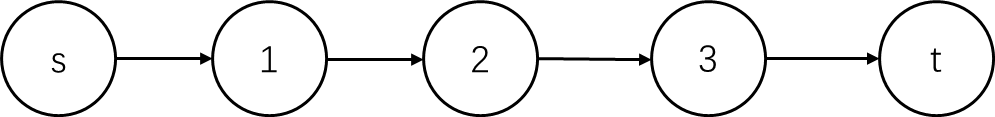
根据最大流-最小割定理，我们知道最大流的值等于最小割的容量，但我们还需要进一步讨论一下最大流与最小割唯一性的关系。

* 1. 最大流值唯一，但流法不唯一。最小割容量唯一，但划分方式不唯一。



如上图所示，所有边权重为1，则最大流值为1，但流法有三种。上图中最小割容量为1，且只有1中分法，即s、1、2、3、4。

继续考察下面的图：



图中，最大流流法只有1种，最小割容量也为1，但最小割有4种，{s}，{s, 1}，{s, 1, 2}，{s, 1, 2, 3}。

* 1. 上面两个例子还能证明：最大流流法唯一，最小割不唯一。最小割唯一，最大流流法不唯一。即二者并没有形式上的联系，仅仅是值相等。

1. **寻找最大瓶颈边的增广**
   1. 使用二分查找

方法类似容量缩放，初始化delta为(1+C)/2，然后删去所有容量小于delta的边，看能否找到增广路径，如果可以则继续增大delta为(C+delta)/2，否则减少delta为(1+delta)/2。不断查找，直到上界小于下界为止。

找到一条增广路径的时间复杂度为logC，

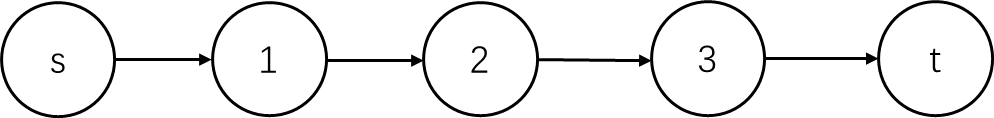
* 1. 使用dijkstra算法直接找到瓶颈边最大的路径

初始化为无穷大，pre[v]=0。dis[v]数组存储s到v的最大瓶颈边，即更新时

dis[v]=max{ dis[v], min( dis[u], c(u, v) ) } , 如果第二项更小，则pre[v]=u。

找到一条增广路径时间复杂度为m\*logn

1. **Blocking flow的回退操作**

****

如果3-t是瓶颈边，则退回到3，重新寻找；

如果3-t以及1-2都是瓶颈边，则退回到1重新寻找。当然，直接退回到s也是可以的。

1. **关于层次图、剩余图以及原图中，增广路径长度的关系**

题目要求：计算原图中，源点到汇点的路径长度小于等于k的最大流。

首先，dinic算法是按照路径长度对剩余图中的增广路径进行了排序，所以一般的想法是求层次图，直到层次图中求得的路径长度大于k时停止，则后面的路径一定也大于k。

但上面的想法存在一个问题，就是层次图是根据剩余图求得的，而剩余图中的路径长度与原图本身就不是相同的。即如果使用了反向边，则反向边的路径长度应该是-1。

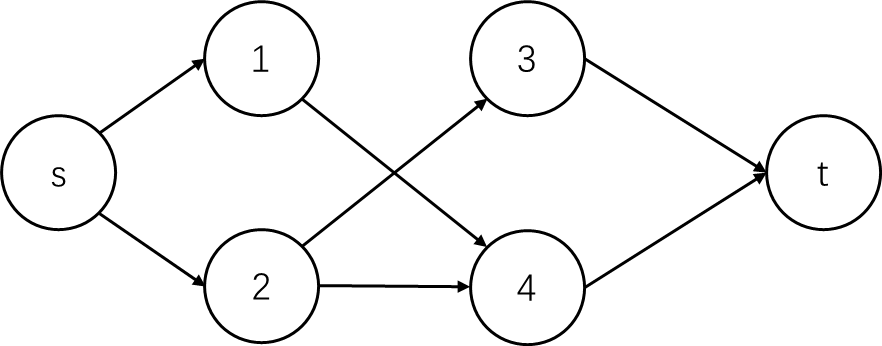
这里我们先直观上说明一下，根据dinic算法，在剩余图和层次图中不断求得的增广路径长度是不减的，因为层次图中会不断使用到剩余图中指向同一层或者前几层的边，这样会使路径长度变长，对应原图路径也会变长。而如果层次图使用了剩余图中的反向边，则其对应原图中的路径也是不减的，因为如果使用了反向边路径减小了，则这条路径应该在之前的层次图中就被找到了。

所以dinic可以保证，不断求得的增广路径在原图中对应的路径也是不减的。

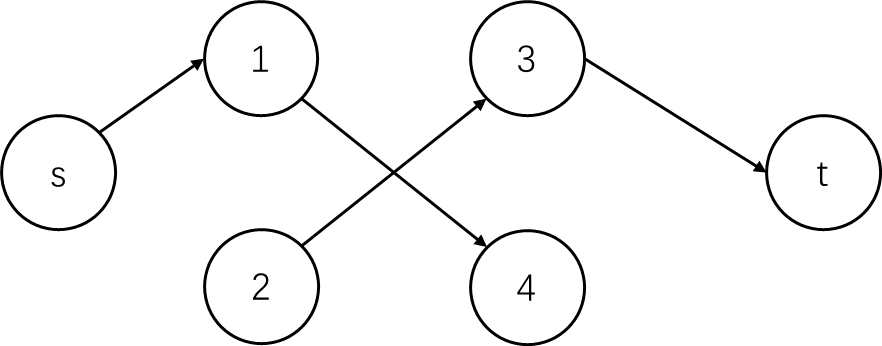
但剩余图和原图增广路径的长度并不是对应的。

反例：

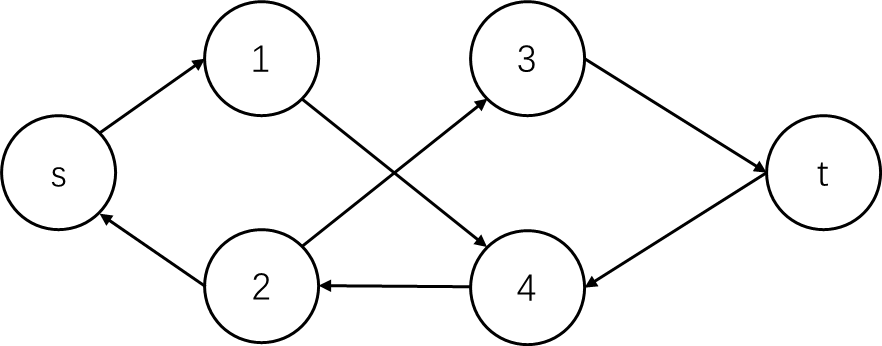
假设原图如下，每条边容量为1，求边长小于等于3的最大流：



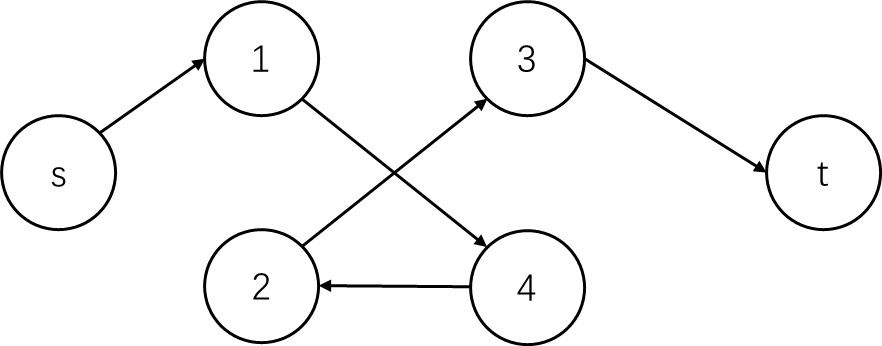
则其本身就是一个层次图，所以层次图也一样。如果我们第一条增广路找的是s-2-4-t，则会删去所有的边，并且无法再到达t。



第一次增广后对应剩余图为：



所以此时我们需要计算一个新的层次图：



此时求得的增广路长度为5，如果我们将增广路长度和原图长度理解为相同的概念，则此时已经不满足要求，应该舍弃。但是，原图中，这条增广路径对应的路径长度其实为3，符合要求。

所以如果按照相同概念，我们只能得到大小为1的最大流，而实际答案应该为2。

所以求得的路径长度都是不减的，但长度本身并不是互相对应的。

这道题可以用网络流来解，只是思路上要用到我们明天课上要讲到一些内容。

可以分成三步来考虑这道题：

1.      求最大流的时候，如果规定流经的路径长度不能超过k，该怎么解？这可以用基于level graph的思路来解。我们明天课上会讲到。

2.      给定一个流图，问图中去掉多少条边，会使得源点到汇点之间不存在长度不超过k的路径。这该怎么解？这在网络流第二部分的slides中会讲：disjoint paths。感兴趣的同学可以提前看一下那部分slides。我不知道明天是不是来得及讲到那部分，因为之前还有二部图匹配要讲。

3.      如果上一问中不是问去掉多少条边，而是问去掉多少个顶点，该怎么解？这个思路很简单，你们自己想想就能明白。教科书上这一章的第13题就是这种顶点有容量限制的网络流版本。

把上面三个思路结合起来就能解出这道题。

**6 Blocking flow与最大流的关系**

Blocking flow指在当前的层次图中，只用正向边已经无路可走时得到的流。Blocking flow就是一个局部最优解，类似于贪心法得到的解，所以blocking flow不一定是最大流，最大流是全局最优解。

因为最终求得的最大流中，不包含反向边（因为反向边原图中没有，结果不能包含图中不存在的边），所以最大流是blocking flow，最大流也不能走到汇点。