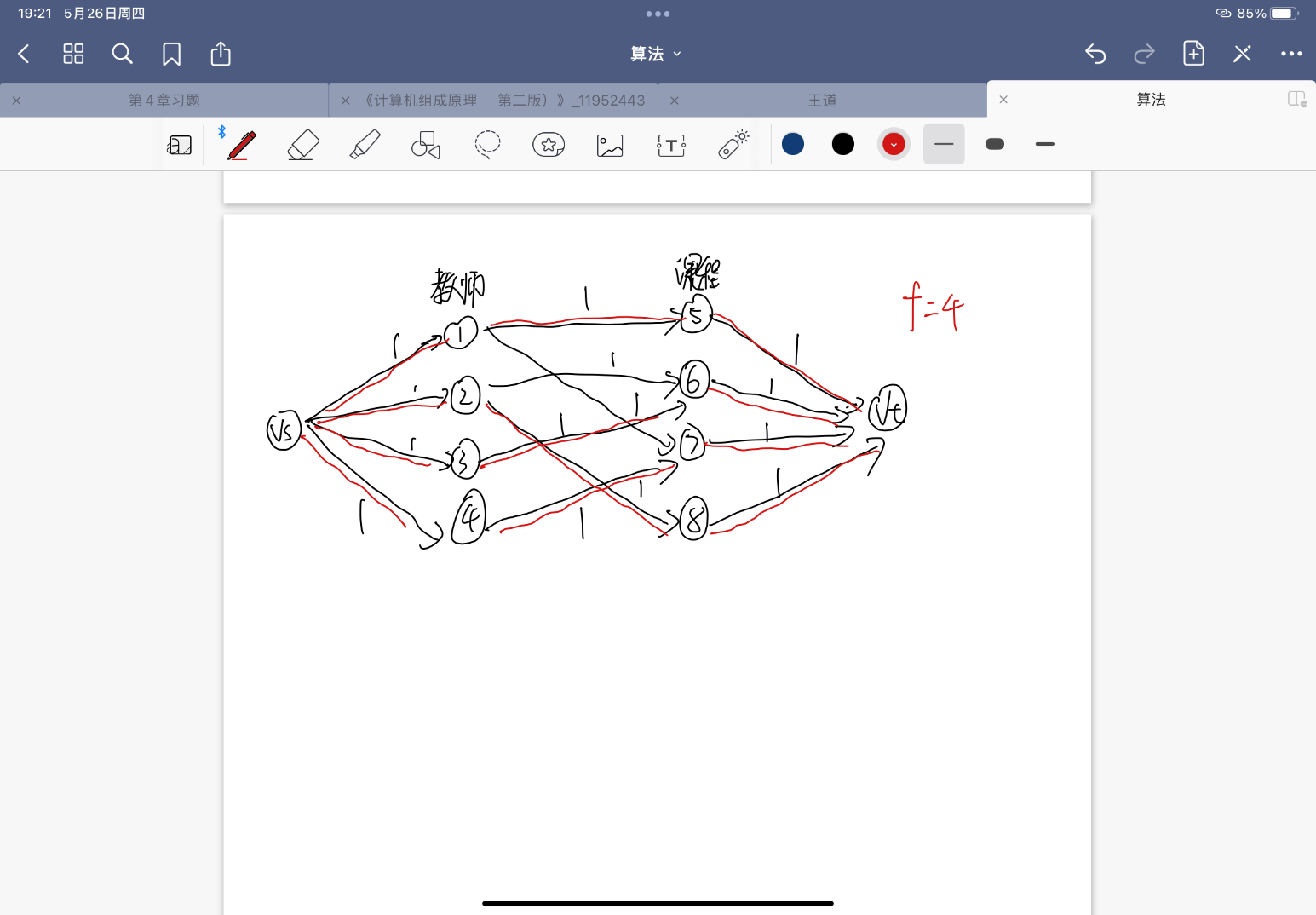
**教师与课程问题**

可将该问题转化成二分图的完备匹配问题，即将教师和课程分别作为二分图的两个端点，做出如下二分图：

对于二分匹配问题，我们可以采用最大流算法和匈牙利算法实现。

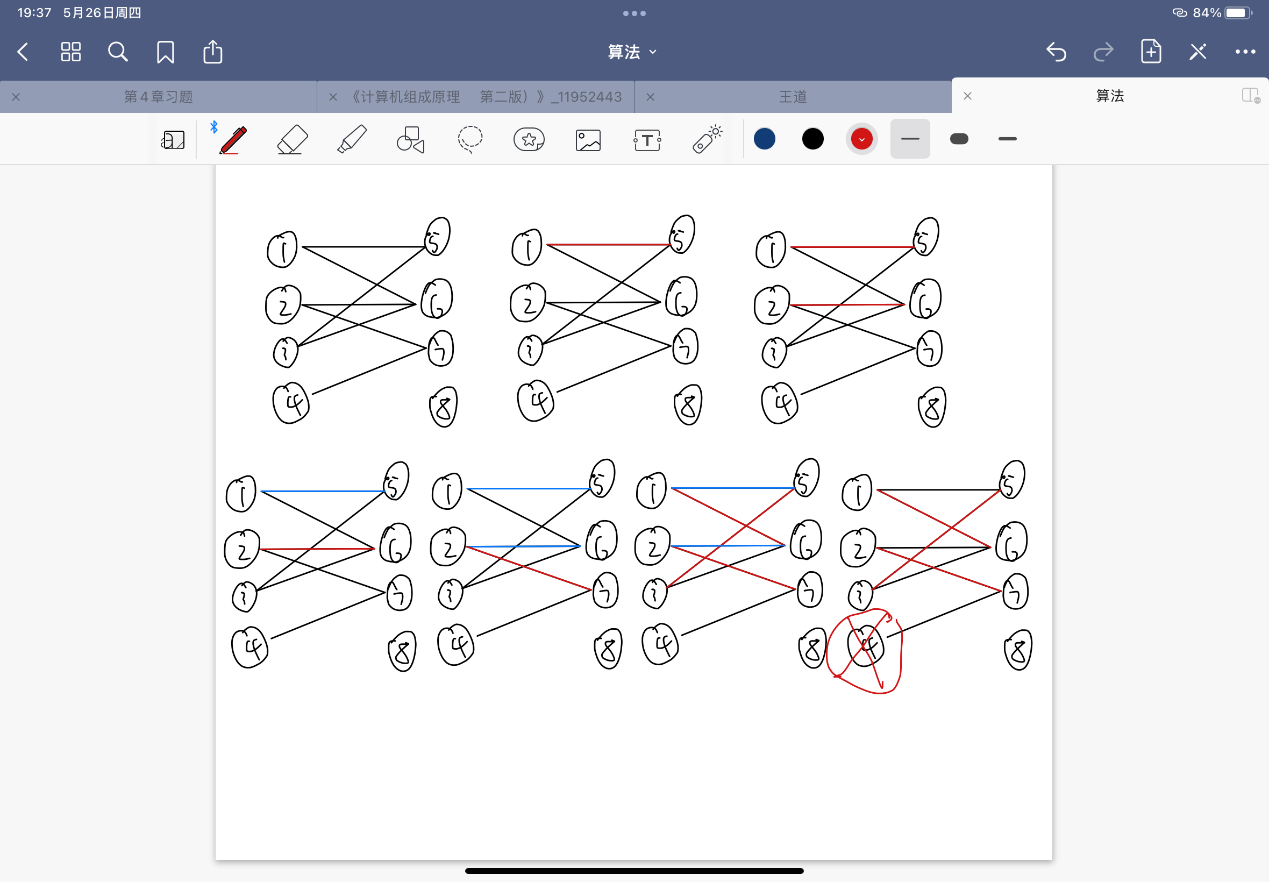
若采用最大流算法，我们需要将教师指向课程的边看作有向边，再做出一个源点和一个汇点，源点指向每一个教师，容量为1，所有课程指向汇点，容量也为1。

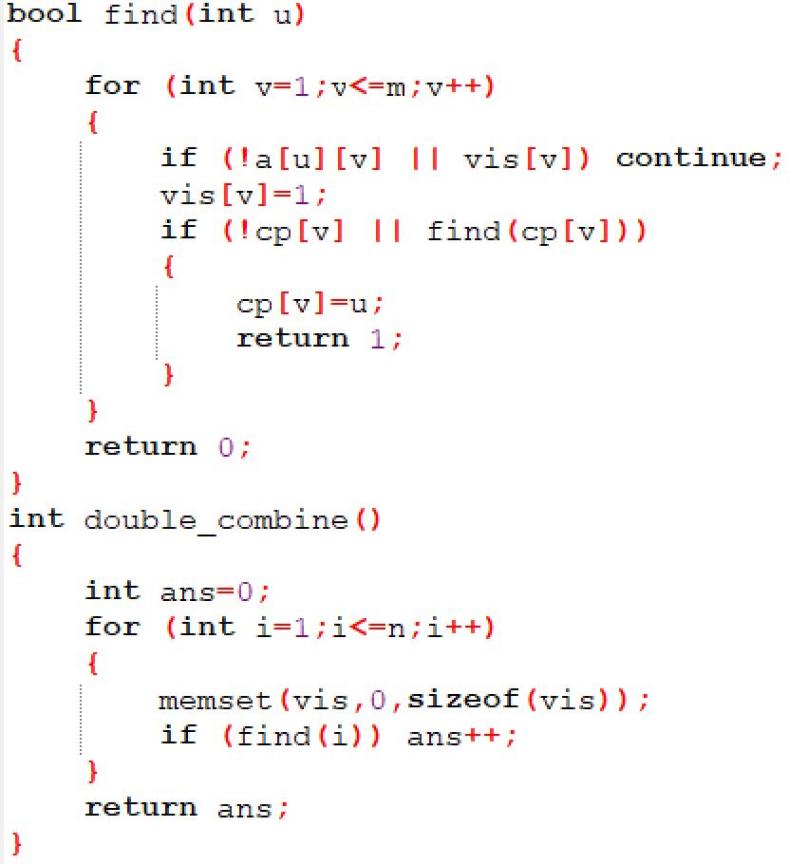


在此图的基础上求解最大流问题即可，最终所得的最大流值即为最大匹配数，如果该数等于n，即可认为可以实现完备二分图匹配，同时流量为1的边表示该教师讲授该门课

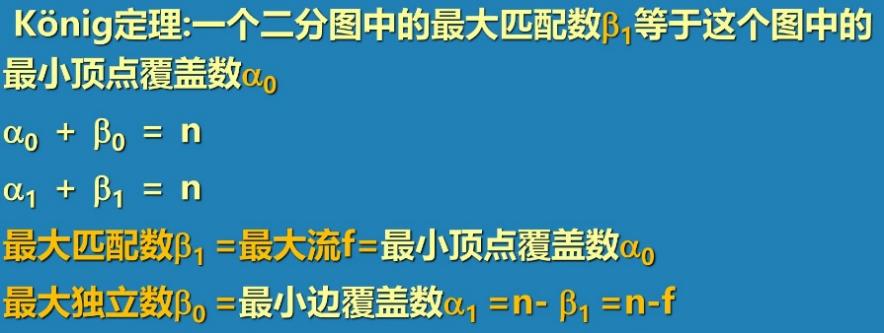
若采用匈牙利算法，不需要建立网络模型，直接将所有边看作无向边即可，也无需添加源点和汇点。

匈牙利算法采用了递归的思路，若当前左端点无法匹配，则去试图更改之前已匹配好的点，尝试着腾出来右端点进行匹配，并将该点也看做已经匹配。若无法腾出来右端点，则证明无法二分完备匹配。





该问题也可看做最大独立集问题，求出最大独立子集，由



可知，最大匹配数=n-最大独立数

**分配问题**

对二分图的每个顶点设一个可行顶点标号,结点函数l,对任意边(x,y)满足l(x)+l(y)>=w(x,y),再使用图中满足足l(x)+l(y)=w(x,y)的所有边生成一个新的子图，新的子图的完美匹配，便是原图的最大权匹配。

其中，对于可行顶点标号，初始化为l(x)=Max w(x,y)即X结点的顶标为从它出发所有边的最大权值，l(y)=0即结点顶标为0。随后对生成子图求完备匹配，若未找到，则按照如下规划修改标号：

将一个未被匹配的顶点u(u in {x})做一次搜索，构成交错树，记下哪些结点被访问那些结点没有被访问。求出d=min{lx[i]+ly[j]-w[i,j]}其中i结点被访问，j结点没有被访问。对于所有访问过的x顶点，将它的可行标减去d，对于所有访问过的y顶点，将它的可行标增加d。然后再次求解完备匹配，若还未找到，再次修改，一直重复直到找到为止。

**医生排班问题**

该问题经过合理的建模之后，可以通过网络流来解决：

该网络流包含如下结点：

源点Vs

汇点Vt

医生i结点

医生i的假期j结点

假期j的第k天结点

包含如下边：

Vs指向每一个医生i结点，容量为C

医生i结点指向医生i的可以值班的假期j结点，容量为1

医生i的假期j结点指向假期j中医生i可以值班的天的结点，容量为1

每天的结点指向Vt，容量为1

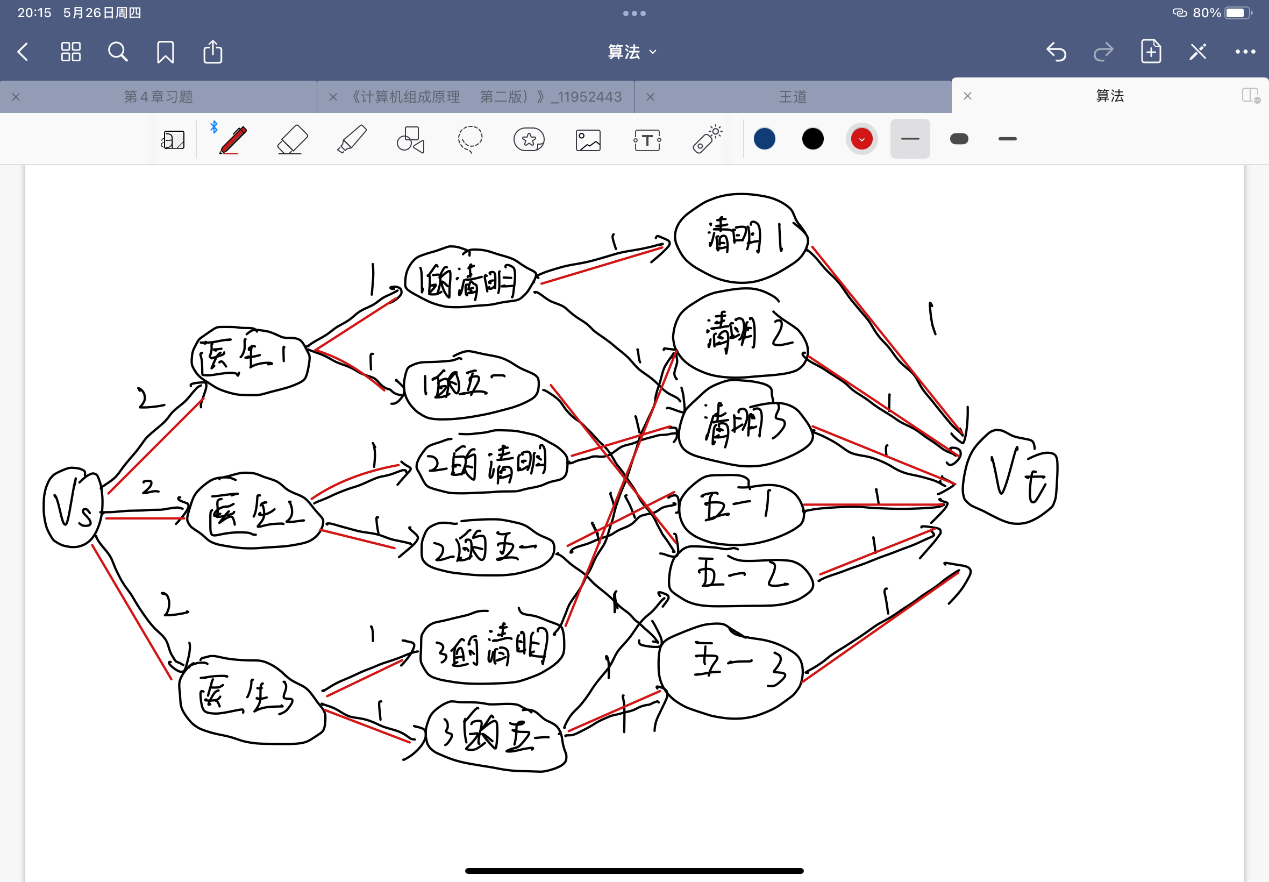
在该图的基础上求解网络流的最大流f，若f等于假期总天数，即可得到问题的解。

这里给出一个例子：

|  |  |  |
| --- | --- | --- |
| 医生1 | 清明第1，3天 | 五一第2天 |
| 医生2 | 清明第3天 | 五一第1，3天 |
| 医生3 | 清明第2天 | 五一第2，3天 |

其中C=2.

针对该例，可构建如图，其中黑色为构建出的图，红色为一种最大流解

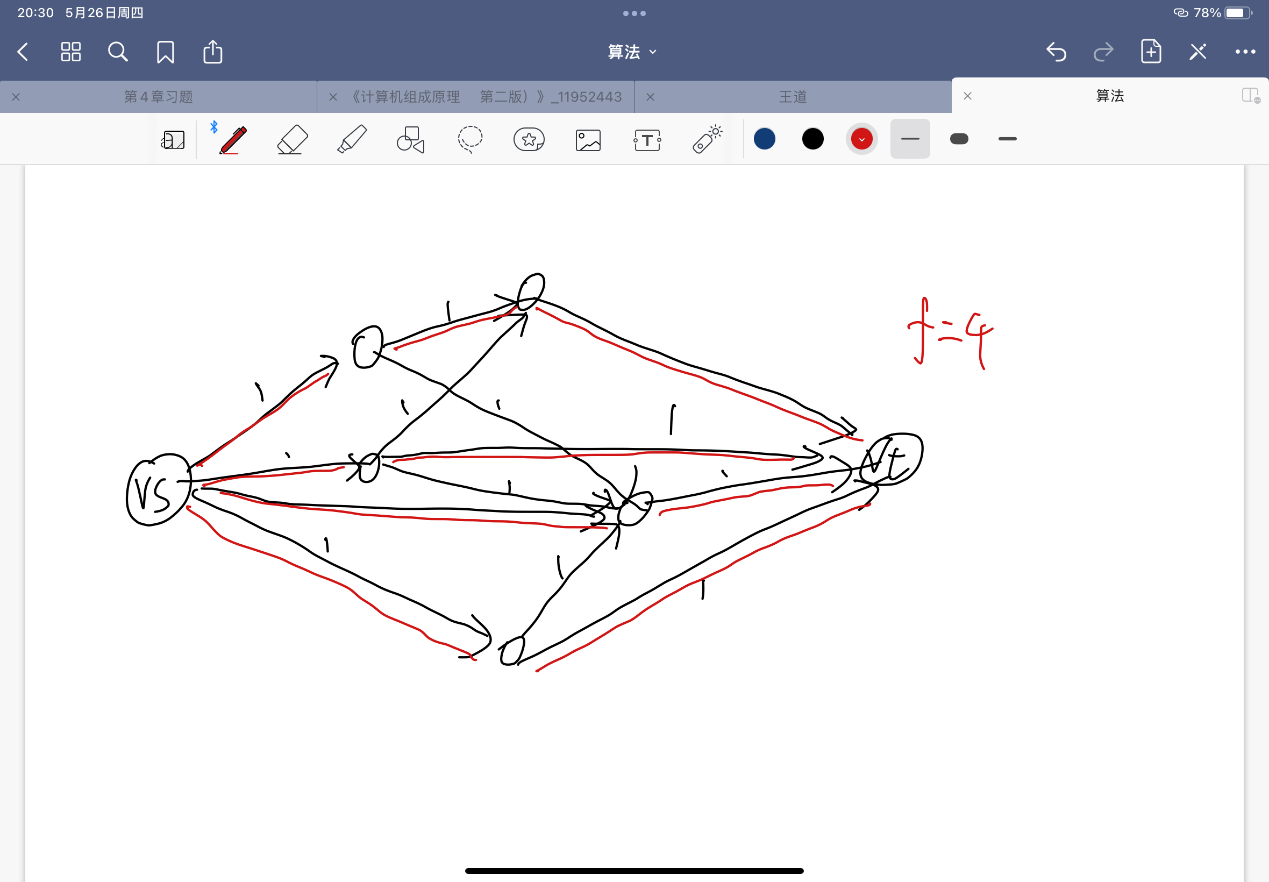


最终得出如下排班

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 清明1 | 清明2 | 清明3 | 五一1 | 五一2 | 五一3 |
| 医生1 | 医生3 | 医生2 | 医生2 | 医生1 | 医生3 |

**最大边不交路径数**

此题可通过简单的网络流建模即可得到，即将起点s作为源点Vs，将终点t作为汇点Vt，将图作为网络流模型，并且每条边的容量均为1，因为每条边只能使用一次，即流过1次，在此图的基础上求最大流f，f即为最大边不交路径数



**试题库问题**

该构造如下网络模型：

该网络流包含如下结点：

源点Vs，汇点Vt，类型结点，试题结点

包含如下边：

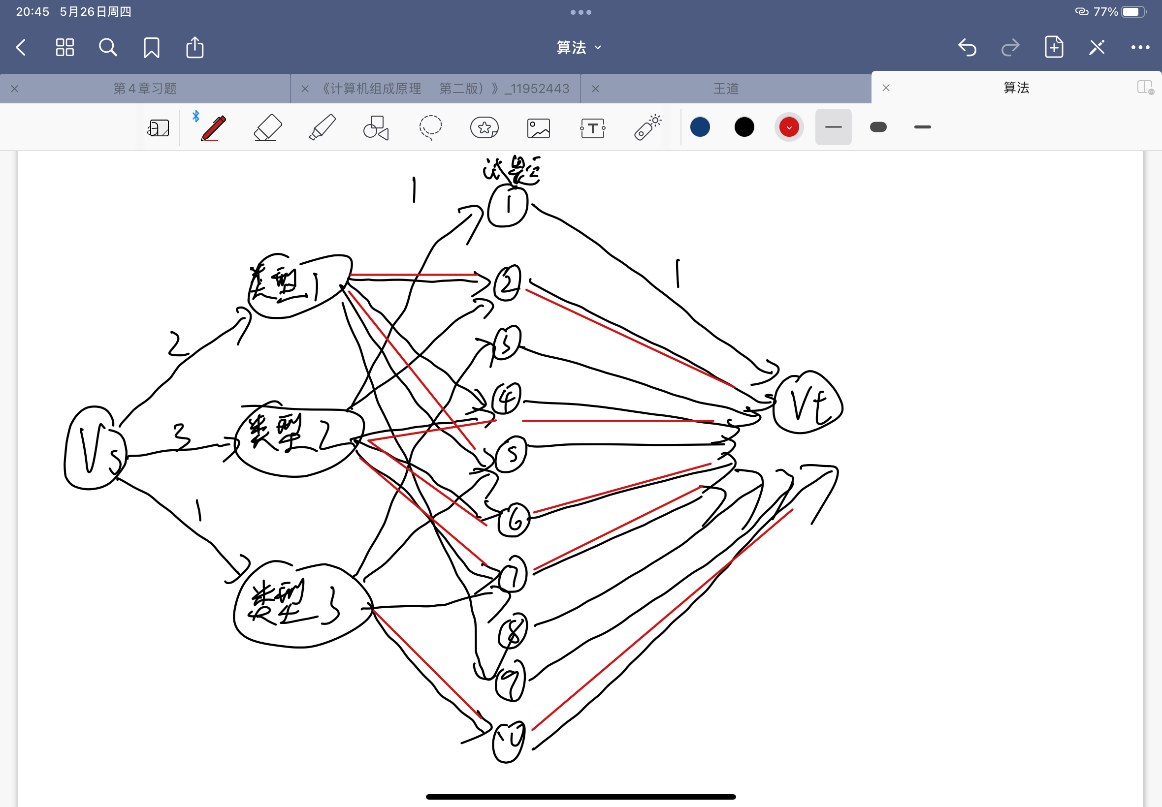
Vs指向类型结点的边，容量为该类试题的需求量

类型结点指向属于这个类型的试题结点，容量为1

试题结点指向Vt，容量为1

在此网络图上求最大流f，若f等于所需试题总数，则满足组卷要求，经过的试题结点构成了整套试题

假设某套试题，一共需要6道题，类型1需要2道，类型2需要3道，类型3需要1道，则可构建如下网络图：



求解最大流f=6，可知满足条件，试卷为2，4，5，6，7，10