光网络业务连续性优化难题-复赛正式赛

> 时间限制：90s　空间限制：512MB
>
> 决赛相对复赛正式赛的改动用红色字体标出，蓝色字体为加粗字体

题目描述

光网络可以看成是一个由 N 个节点，M 条边组成的无向连通图，在本题中，光网络通过对 Gabriel图 增减边来生成，两个节点之间可能存在多条边，图上的每条边代表现实世界的一条光纤，每条光纤上存在 $K = 40$ 条光通道；

在光网络上初始时运行着多条光业务，一条起点为 S，终点为 T，宽度为 W 的光业务可以看作是一条从起点 S 到终点 T 的**简单路径**(即路径上不存在重复节点或重复边)，其中对于路径上的每条边，使用了其中的 W 个**编号连续**的通道；多条业务可以使用同一条边的不同通道，但是不能使用同一条边的相同通道；另外，每条业务会有对应的业务价值，用 V_i 表示第 i 条业务的业务价值；

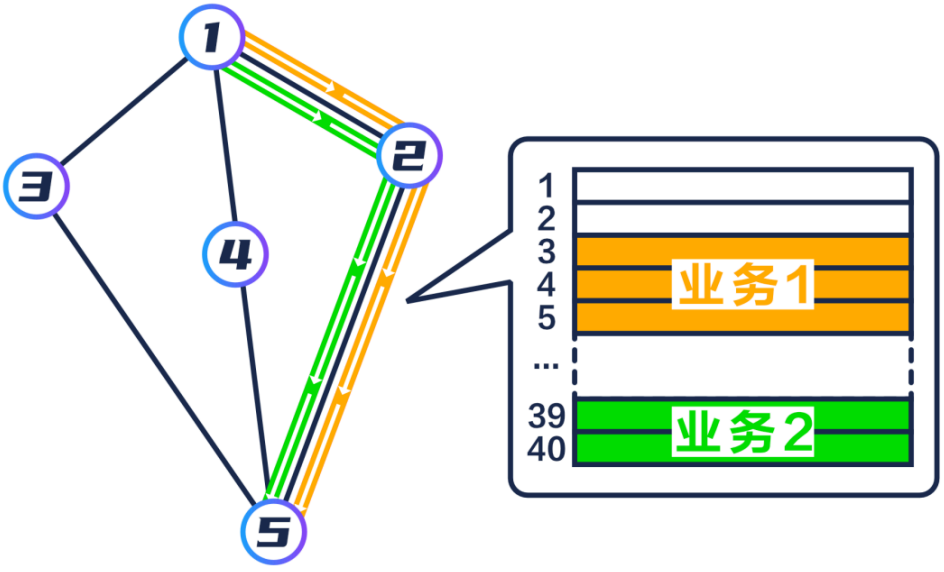

图 1

如图1所示，此时光网络中运行着两条光业务；业务 1 和业务 2 的起点为 1，终点为 5，其中业务 1 的宽度为 3，使用了通道 [3, 5]，业务 2 的宽度为 2，使用了通道 [39, 40]，它们的业务路径为 (1 → 2 → 5)；

在普通情形下，一条业务在其顺序经过的所有边 $e_1, e_2 \ldots e_n$中，使用的通道编号集合应该是相同的；具体的，令 C_{e_k} 为该业务在路径上第 k 条边的使用通道集合，则 $C_{e_{k+1}} = C_{e_k}$；但有些节点上存在若干次变通道的次数 P_k，令 $e_k = (u, v)$，表示这条边连接了 u 和 v 节点，$e_{k+1} = (v, w)$，如果此时节点 v 的变通道次数还有剩余（$P_v > 0$），则可以使用一次 P_v 使得 $C_{e_{k+1}} \neq C_{e_k}$；

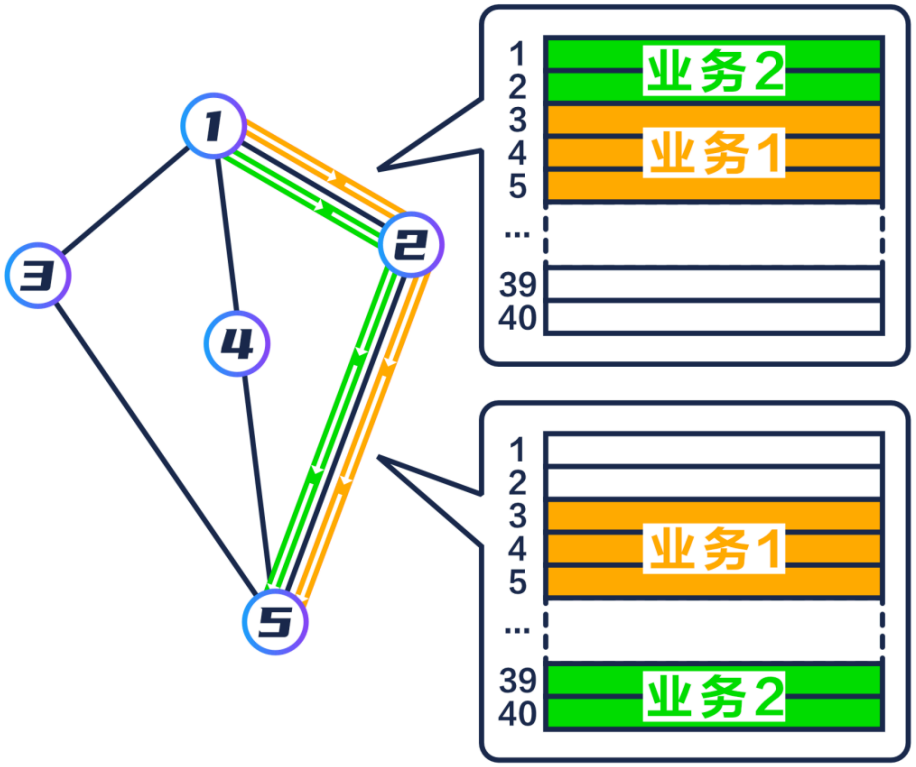

图 2

如图 2 所示，若初始时节点 2 的变通道次数为 1，可以注意到业务 2 在（1，2）边上使用了通道 [1，2]，在（2，5）边上使用了通道 [39，40]，节点 2 的变通道次数则变为 0，那么此时业务 1 则无法经过节点 2 变通道；

在实时的网络运行中，可能存在自然灾害、硬件故障或者维护活动引起的多种光纤故障情况；当光纤发生故障后，**只有**业务路径经过该故障光纤的业务需要被重新规划，但并非所有的业务都能被成功规划路径：

重新规划的业务路径必须满足，起点、终点和宽度与业务的原始属性一样，并且满足上述约束；另外，重新规划的某条业务虽然不能使用其它业务的资源，但是可以使用该业务当前的资源(即通道和节点变通道能力)，例如，如果光纤故障影响了业务 1 和业务 2，记业务 1 的新老路径为 new 1，old 1，记业务 2 的新老路径为 new 2，old 2，则 new 1 不能使用 new 2 和 old 2 的资源，但 new 1 可以使用 old 1 的资源。若规划成功则业务将释放它们原来占用的边上的通道以及对应节点的变通道次数；如果 new 1 使用了 old 1 的部分资源，则该部分资源不会在规划成功后被释放；

重新规划的业务是同时进行的规划，不存在顺序关系，意味着只有此批的业务重新规划完，才会释放自己的老路径资源；

若你没有为某些业务规划路径，则视为这些业务死亡，**它们所占用的边上的通道以及对应节点的变通道次数不会被释放，且在该测试场景下后续这些死亡的业务不能再重新规划；**

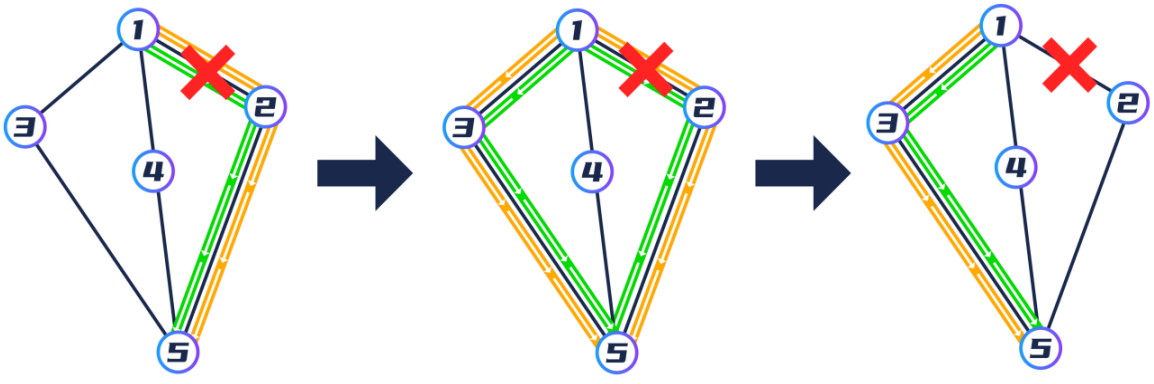

图 3

如图 3 所示，当网络发生（1，2）边的中断时，业务 1 和 2 将重新规划路径，且在规划成功之后它们原始路径上占用的资源将会被恢复（路径通道，变通道次数）；

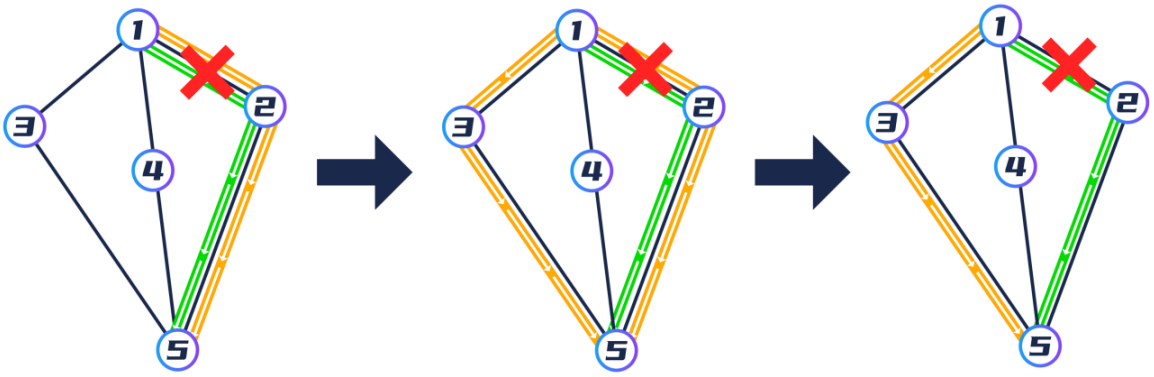

图 4

如图 4 所示，当网络发生（1,2）边的中断时，只有业务 1 重新规划了路径，虽然业务 2 死亡，但后续它依然将一直占用原始路径上的资源；

在这种情况下，你需要设计出一种算法，确保在多次光纤故障后，网络上存在总价值尽可能大的未死亡业务；

同样的，在初赛中我们也实现了对应的 baseline 解法，该解法基于最短路算法，将同批的规划路径请求按照权重从大到小排序，随后依次为每条业务寻找全局最短路，只会使用不变通道的路径；

在复赛中，你的算法不仅需要处理多次光纤故障后的业务重新规划，还应该尽量寻找到 baseline 解法的瓶颈用例；

具体来说，你的算法需要在读入初始环境后，输出 T_1 组断纤序列，你的算法输出的 T_1 组断纤序列将会和我们内置的 $T_2(1 \leq T_2 \leq 70)$ 组断纤序列综合作为测试数据；

同时，为了防止你给出的 T_1 组序列相似度过高，令 S_i 为第 i 组序列构成的集合，定义两个集合的 $Jaccard$ 相似度 为 $J_{i,j} = \frac{|S_i \cap S_j|}{|S_i \cup S_j|}$，你给出的序列必须保证 $Max_{i,j}J_{i,j} \leq 0.5, i \in [1, T_1], j \in [1, T_1], i \neq j$;

节点变通道主要在物理机器上插入光传输单元(Optical Transport Unit)板卡来实现，通过将光信号转化为电信号，电信号再转化为光信号，来实现光通道的改变；然而 OTU 板卡的数量是有限的，这种情况下，如何合理的规划 OTU 板卡的分配成为了一个很关键的问题，一个好的分配方式，能够极大程度上提高网络中业务的生存性；

在决赛中，你可以重新规划网络中各个节点变通道能力的大小，主要的限制是重新规划后的节点变通道能力之和不能超过原始网络中的节点变通道能力之和，并且单个节点变通道能力不能超过节点的变通道能力上限 20；

题目交互

你的程序应该以系统的标准输入流和标准输出流作为系统的标准输入输出；

初始环境输入

第一行两个整数 N 和 M，N 表示图的节点数，M 表示图的边数；
$(2 \leq N \leq 200, 1 \leq M \leq 1000)$;

第二行 N 个整数，第 i 个整数 P_i 表示节点 i 允许的最大变通道数；$(0 \leq P_i \leq 20)$

接着 M 行每行两个整数 u_i, v_i 表示图上的第 i 条边 (u_i, v_i)；$(1 \leq u_i, v_i \leq N, u_i \neq v_i)$;

输入的图保证连通，无自环，可能有重边；

接着一行一个整数 J，表示图上初始运行的业务数；$(1 \leq J \leq 5000)$；

接着 $2J$ 行每两行表示一条业务；

每条业务的第一行六个整数 Src, Snk, S, L, R, V 表示该业务的起点为 Src，终点为 Snk，经过的边数目为 S，在路径上所有边的占用通道范围为 $[L, R]$，该业务的业务价值为 V；第二行 S 个整数依次表示业务依次经过的路径上的每条边的编号，$e_1, e_2 \ldots e_S$，其中 e_k 表示路径的第 k 条边；
$(1 \leq Src, Snk \leq N, Src \neq Snk, 1 \leq L \leq R \leq K,\ 0 \leq V \leq 100000,$
$1 \leq e_k \leq M)$

注意初始业务并不会变通道；

题目中所有编号都是从 1 开始，业务的编号和边的编号按照输入顺序递增；

注意交互部分的每个测试场景，都是从该初始环境开始；

交互部分

输出重新分配后的各个节点的变通道能力：

你需要输出一行 N 个整数，第 i 个整数 P_i' 表示节点 i 重新规划后的变通道能力；

$0 \leq P_i' \leq 20, \sum_{i=1}^{N} P_i' \leq \sum_{i=1}^{N} P_i$

输出瓶颈断边场景的交互部分：

你需要输出一行一个整数 T_1 表示你提供的断边测试场景的个数；需要保证 $0 \leq T_1 \leq 30$；

接下来你需要输出 $2 \cdot T_1$ 行每两行表示一个测试场景，其中第一行第一个整数 c_i 表示该断边测试场景的断边个数；第二行 c_i 个整数表示顺序断边的边编号，需要保证 $0 \leq c_i \leq 60$ 以及这 c_i 个整数都是合法的边编号且互不相同；

读入测试场景并规划路径的交互部分：

首先一行一个整数 T 表示$T(1 \leq T \leq 100, T = T_1 + T_2)$ 个独立的测试场景，前 T_1 个测试场景是你的程序提供的测试场景，在处理每一个场景时，都是保证**从上面的初始环境开始**的(此时还没有任何中断边)；

对于每一个场景，你的程序需要接收一系列的整数 e_{failed} ，e_{failed} 表示此时网络中发生中断的边的编号，$(1 \leq e_{failed} \leq M)$，在每接收到一个 e_{failed} 时，你的程序需要重新规划因为此次中断所影响的业务的路径，以及为路径上的每条边规划通道；具体的，对于当前的 e_{failed}，假设当前受影响的业务集合为 $Serv = \{S_1, S_2 \ldots S_n\}$，那么你的程序应该输出一行一个整数 R，其中 $0 \leq R \leq n$，表示规划成功的业务数，接下来输出 $2R$ 行表示每个规划成功的业务，其中第一行两个整数 $Serv_{id}$ 和 S ，分别表示业务的编号和业务的新路径的边数，第二行 $3S$ 个整数 e_k, e_l, e_r 表示业务依次经过的边的编号以及在该边上占用的通道区间为 $[e_l, e_r]$；直到你的程序接收到一个整数 -1 时，则表示该测试场景结束，然后你需要将程序恢复到初始环境状态，进入下个测试场景；

每个测试场景的 e_{failed} 个数不超过 60；

注意：在你每输出一行时，为了确保交互程序能正确读取你的输出，你需要刷新输出，其中：

C 和 C++ 可以使用 `fflush(stdout);`

Java 可以使用 `System.out.flush();`

评分

对于每一个用例 i，令 T_i 为此用例测试场景的个数，令 $BeginV_i^j$ 为第 j 个测试场景开始时的业务总价值，令 $EndV_i^j$ 为第 j 个测试场景结束后存活的业务总价值；则该测试场景的得分为 $Score_i^j = \frac{EndV_i^j \cdot 10000}{BeginV_i^j}$；该用例的分数为所有测试场景的得分之和 $Score_i = \sum_{j=1}^{T_i} Score_i^j$；

令 $Score_i^1$ 是你的算法的用例分数，令 $Score_i^2$ 是我们的 baseline 算法的用例分数，则题目的得分为所有用例的差距得分之和 $Score = max(\sum(Score_i^1 - Score_i^2), 0)$；

分数越高排名越前；分数相同时，先提交的排名靠前；

样例

初始环境输入

```
5 6
1 1 1 1 1
1 2
2 5
1 4
4 5
1 3
3 5
2
1 5 2 1 20 1
1 2
1 5 2 21 40 1
1 2
```

交互部分

```
程序输出：1 1 1 1 1
程序输出：1
程序输出：2
程序输出：1 6
环境输入：1
环境输入：1
程序输出：2
程序输出：1 2
```

> 程序输出: 5 1 20 6 1 20
>
> 程序输出: 2 2
>
> 程序输出: 5 21 40 6 21 40
>
> 环境输入: 6
>
> 程序输出: 2
>
> 程序输出: 1 2
>
> 程序输出: 3 1 20 4 1 20
>
> 程序输出: 2 2
>
> 程序输出: 3 21 40 4 21 40
>
> 环境输入: -1

错误类型

基础错误类型

1. 代码编译错误
2. 程序异常退出（可能原因：运行错误，使用异常权限，输出参数比实际多，输出参数格式不对，etc...）
3. 超出时间限制（可能原因：交互时未使用清空流缓存命令，程序运行超时，输出参数比实际少，etc...）
4. 超出内存限制

逻辑错误类型

1. "Unknown Error"（出现时请联系大赛方）
2. "RE"（程序内存访问越界或异常退出）
3. "Incorrect Number of Services"（输出的业务数量不对）
4. "Incorrect Service ID"（输出的业务ID不对）
5. "Duplicate Service ID"（单次输出的业务ID重复）
6. "Unaffected Service ID"（输出未受到此次光纤中断影响的业务ID）
7. "Incorrect Number of Edges"（路径的边数不对）
8. "Incorrect Edge ID"（路径的边ID不对）
9. "Duplicate Edge ID"（路径上的边ID重复）
10. "Pass Break Edge"（路径经过了已经中断的光纤）
11. "Inconsistent Service width"（路径上的业务宽度不一致）
12. "Incorrect Channel ID"（业务所使用的通道ID不对）
13. "Cyclic Path"（路径成环）
14. "Channel Occupied Kind 1"（业务所使用的通道被占用，可能与其他业务的老路径有关）
15. "Channel Occupied Kind 2"（业务所使用的通道被占用，可能与其他业务的新路径有关）
16. "Disconnected Path"（路径不连通）
17. "Insufficient Channel Quantity"（节点变通道次数不够）
18. "Mismatched start and end"（业务新路径的起点或终点与老路径不匹配）

19. "Incorrect Number of TestCase"（输出的测试场景个数或者中断边数超过范围）
20. "Incorrect Edge ID of TestCase"（输出的测试序列中断边ID不对）
21. "Duplicate Edge ID of TestCase"（输出的测试序列中断边重复）
22. "Large Jaccard Similarity Between TestCase"（输出的测试序列两两间 Jaccard 相似度超出阈值）
23. "Incorrect Channel Quantity"（重新分配的节点变通道次数不满足限制条件）