吉林大学 2019-2020 学年第 2 学期

《数据结构》课程设计

软件学院 D 题

（1）本题满分 100 分，以下每小题 50 分，即**任选 2 小题**完成即可获满分。各小题的难度与顺序无关。

（2）没带教材，没有课件怎么办？在超星慕课平台上，有完整的教学视频和课件。

（3）本题在 jlu.openjudge.cn 提交，在本题中，除了题目得分外，你在本班的排名将计入课程平时分。排名的规则是：得分多者排名靠前，得分相同的提交次数少者排名靠前，得分和提交次数都相同的完成时间靠前者排名靠前。所以如果 A 同学多次提交，完成了题目，然后把代码借给 B 同学抄，那么 B 同学一次提交就过了，这样 B 的名次会在 A 之前。

（4）C 题选择麻将的同学可不做 D 题。

（5）本题仍将进行软件批量查重，同学之间可以交流思路，但不允许交流源代码。被判定抄袭的同学，抄袭者与被抄袭者双方同论，不做区分。若有同学被判为抄袭与被抄袭，老师可以不联系这些同学，而是直接扣分。将网络上的代码改头换面提交者同样视为抄袭。之前题目，已有同学被判定为抄袭，并进行了严肃的处理，不但坑了自己，更坑了自己的朋友，非常遗憾。

（6）除在 jlu.openjudge.cn 提交程序外，与 A 题一样，每名同学在超星作业处提交自己全部通过题目的源文件，以“班号-姓名拼音-题号”命名，如 5 班许仙 D3 题，提交 5xuxian3.cpp。在老师登记成绩前仍未使用实名、未设置班级或未在本学院页面提交，导致成绩为 0 者。给予 1 次完善补充用户信息的机会，但该题分数减掉 15 分。

（7）本题无需提交课设报告，不允许使用 STL，使用 STL 通过的题目无效。

（8）本题不允许在 OJ 上注册小号提交，此类行为将被视为作弊，取消本题成绩。

D-1

Time Limit: 500MS　　Memory Limit: 256MB

我们称一个字符串 S 的后缀 T 为好后缀，如果 T 满足如下条件：
（1）它在字符串中至少出现 2 次；
（2）它是满足条件(1)的最长者。
我们称一个字符串 S 的子串 T 为好中缀，如果 T 是去除 S 中满足如下条件的两个子串 p 和 q 后剩余的字符串。
（4）p 是 S 的前缀，q 是 S 的后缀；
（5）p=q；
（6）p 和 q 是满足条件(4)(5)的所有子串中的第二长者。

注意一个字符串不能称为自己的前缀或后缀。好中缀至少为空串，其长度大于等于 0，不能为负数。请编写程序计算给定字符串 S 的好后缀和好中缀长度之和。

输入格式：

输入为一个字符串，包含不超过 100000 个字母。

输出格式：

输出为一个整数，表示输入字符串的好后缀长度与好中缀长度之和。

样例：

输入	输出	解释
abcabcxxxabcabc	15	好后缀为 abcabc，长度 6 好中缀为 abcxxxabc，长度 9
xacbacba	12	好后缀为 acba，长度 4 好中缀为 xacbacba，长度 8
abc	3	好后缀为空，长度 0 好中缀为 abc，长度 3
aaa	3	好后缀为 aa，长度 2 好中缀为 a，长度 1

D-2

Time Limit: 1000MS　　Memory Limit: 20MB

给定一个正权有向图，图中包含 n 个顶点，编号为 0 至 n-1。以顶点 0 作为源点，编程序求顶点 0 到各顶点的最短路径。若顶点 0 到某顶点存在多条最短路径，则输出经过顶点最少的那条路径，例如图 1(a)中 0 到 4 的经过顶点最少的最短路径为 $0 \rightarrow 3 \rightarrow 4$。**若存在多条最短路径且其经过顶点个数相等，则输出字典序最小者。例如图 1(b)中 0 到 5 的满足条件的最短路径为 $0 \rightarrow 2 \rightarrow 5$。**

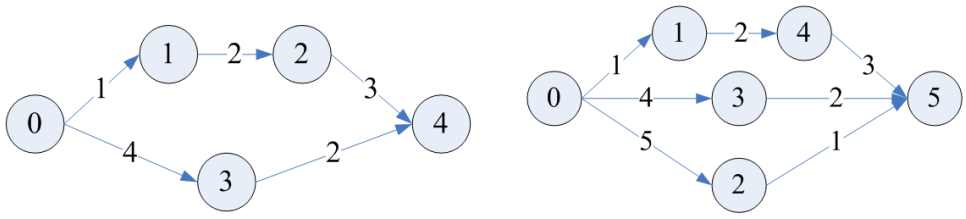


图 1　正权有向图示例

注：字典序，即对象在字典中的顺序。对于两个数字序列，从第一个数字开始比较，当某一个位置的数字不同时，该位置数字较小的序列，字典序较小，例如 1 2 3 9 比 1 2 4 5 小，1 2 8 9 比 1 2 10 3 小。

输入格式：

输入第一行为两个正整数 n 和 e，分别表示图的顶点数和边数，其中 n 不超过 15000，e 不超过 1000。接下来 e 行表示每条边的信息，每行为 3 个非负整数 a、b、c，其中 a 和 b 表示该边的端点编号，c 表示权值。各边并非按端点编号顺序排列。

输出格式：

输出为若干行由“->”间隔的数字序列，每行为源点 0 到某顶点的满足条件的最短路径，如源点到某顶点无最短路径，则不输出该条路径。各条路径按终点的递增顺序输出，源点到源点的最短路径无需输出。

样例：

输入	输出
6 7 0 1 1 1 4 2 4 5 3 0 3 4 3 5 2 0 2 5 2 5 1	0->1 0->2 0->3 0->1->4 0->2->5

D-3

Time Limit: 500MS　　Memory Limit: 20MB

为鼓励公交出行，Tabu 市出台一项优惠政策，若出行乘坐公交车，则可以免费坐一站地铁，下地铁后，可使用刚才的公交票继续坐公交。小明想以最短的时间出行，请编写程序为小明找到一条从起点到终点最快的路线及换乘地铁的方案。假设换乘时间忽略不计，公交车与地铁站点相同，但线路和速度不一定相同，所有线路都是双向的。

输入格式：

输入第一行为 3 个整数 n、s 和 t，分别表示车站数（编号为 1 至 n），小明所在的起点站和想去的终点站。下一行为一个整数 m，表示公交车的线路信息，接下来 m 行，每行为 3 个正整数 a、b、c，其中 a 和 b 为车站编号，c 表示公交车从 a 站到 b 站所需的时间（即从 a 站到 b 站需要 c 分钟）。下一行为一个整数 k，表示地铁的线路信息，接下来 k 行，每行为 3 个正整数 a、b、c，表示地铁从 a 站到 b 站需要 c 分钟。k 不超过 200，其余整数均不超过 20000。

输出格式：

输出为 2 行，第 1 行为 1 个整数，表示从起点站到终点站的最短时间；第 2 行为一个整数，表示换乘地铁的站点编号，若有多个可能的换乘站点，则输出编号最小者，如果无需换乘地铁，则第二行输出“only bus”。

样例：

输入	输出
4 1 4 4 1 2 2 1 3 3 2 4 4 3 4 5 1 2 4 3	5 2

D-4

Time Limit: 1000MS　　Memory Limit: 512MB

已知树结点为不等于 0 的整数。编写程序找出非空树中乘积最大的路径。本题的“路径”定义为树中的结点序列 $v_i, \ldots, v_j$，序列中前一个结点是后一个结点的父结点，但路径不一定是以根结点为起点，也不一定是以叶结点为终点。路径的乘积定义为该路径所包含的所有结点的数据值之积。例如对于图 2(a)所示的树 t，乘积最大的路径为 8-5-6。

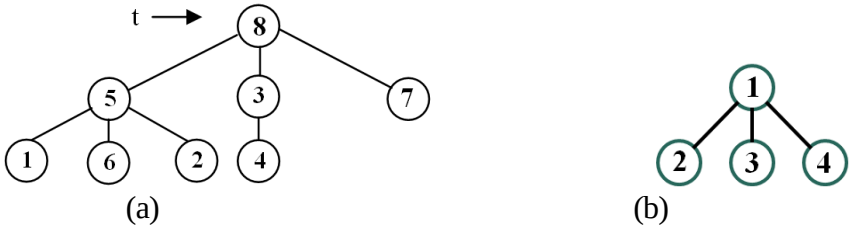


图 2 树示例

输入格式：

输入为一组用空格间隔的整数，个数不超过 100 个，表示加入空指针信息的二叉树先根序列。其中空指针信息用 0 表示。

注：我们已知二叉树与其自然对应的树相比，二叉树中结点的左孩子对应树中结点的左孩子，二叉树中结点的右孩子对应树中结点的右兄弟。进而我们可以利用“基于引入空指针信息的先根序列构建二叉树”的方法来构建其对应的树的左孩子-右兄弟存储结构。如 8 5 1 0 6 0 2 0 0 3 4 0 0 7 0 0 0 对应图 2(a)所示的树，1 2 0 3 0 4 0 0 0 对应如图 2(b)所示的树。

输出格式：

输出为两行，第一行为该树的路径乘积的最大值，第二行为一组空格间隔的整数，即该最大乘积路径包含的结点值（按所在层数递增顺序输出）。**如果存在多条满足条件的路径，则输出最短（包含结点个数最少）者，如果存在多条最短的路径，则输出最靠左上者**。

样例：

输入	输出
8 5 1 0 6 0 2 0 0 3 4 0 0 7 0 0 0	240 8 5 6
1 2 0 3 0 4 0 0 0	4 4

D-5

Time Limit: 1000MS　　Memory Limit: 512MB

有 n 个房间，编号为 0…n-1，每个房间都需要网络连接。房间 i 有网络，当且仅当满足如下 2 个条件之一：
（1）房间 i 安装了路由器（成本为 $r_i > 0$）
（2）房间 i 和房间 j 有网线连接且房间 j 有网络（在房间 i 和房间 j 之间布置网线的成本为 $f_{ij} > 0$）

请编写程序给出一个网络布线方案（哪些房间安装路由器，哪些房间之间布置网线），使得所有房间都有网络，且总成本最小。房间编号从 0 开始。

例如图 3 包含 7 个房间和 10 个可能的连接，安装路由器的成本为括号内数字，房间之间布置网线的成本为边的权值。其解决方案为右下图，即在房间 1 和 4 安装路由器，并进行图中的网线布置。总成本为 120。

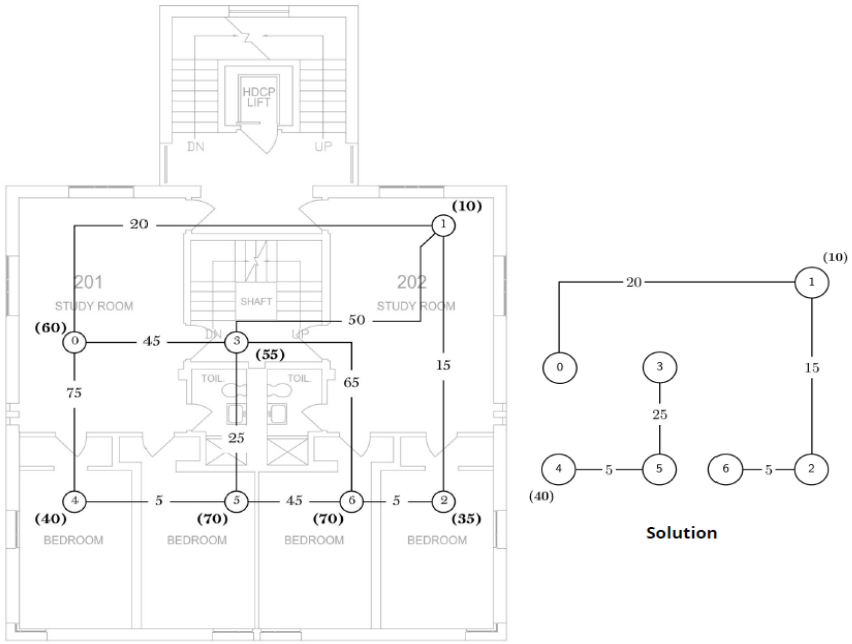


图 3

输入格式：

输入第 1 行为两个正整数 n 和 e；n 为房间数，不超过 100；e 为可能的连接数，不超过 1000。接下来 1 行为 n 个空格间隔的正整数，第 i 个整数($i \geq 0$)表示在房间 i 安装路由器的成本。接下来 e 行，每行为 3 个非负整数 i、j、f，表示在房间 i 和房间 j 之间布置网线的成本为 f。

输出格式：

输出为为一个整数，表示最优网络布线方案的成本。

样例：

输入	输出
7 10 60 10 35 55 40 70 70 0 1 20 0 4 75 0 3 45 1 3 50 1 2 15 2 6 5 5 6 45 4 5 5 3 5 25 3 6 65	120

D-6

Time Limit: 5000MS　　Memory Limit: 512MB

百度、谷歌等搜索引擎往往包含这样一个功能：当用户输入了错误的单词时，系统能够自动识别并向用户建议正确的单词。Word 等文字处理软件也有类似的功能。这个功能是如何实现的呢？一种典型的实现方法是：系统后台维护一个“字典”，当用户输入的单词不在字典中时，则认为是错误的单词，并在字典中查找与用户所输单词相似度高且用户使用频率高的单词，向用户建议。

图 4

采用“距离”如何衡量两个单词的相似度。两个字符串的距离定义为一个字符串转化成另一个字符串，所需的最少“操作”次数。有三种操作：增加一个字符、删除一个字符、替换一个字符。显然，距离越大，说明两个字符串的相似程度越小；距离越小，说明两个字符串的相似程度越大。对于两个完全相同的字符串，距离为 0。

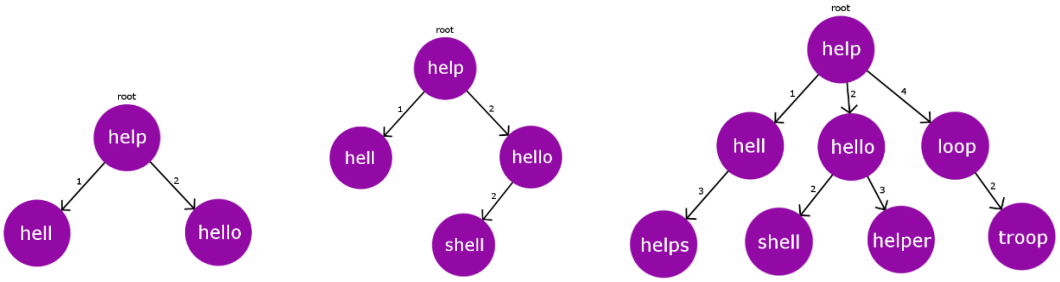
例如将 FOOD 转换成 MONEY，最少通过如下 4 步操作：第 1 位 F 替换为 M，第 3 位 O 替换为 N，在第 4 位插入 E，第 5 位 D 替换为 Y。故 FOOD 和 MONDY 的距离为 4。

F	O	O		D
M	O	N	E	Y

这样，当用户输入错误单词后，则系统可以在字典中查找与该错误单词距离不超过 n（n 一般不超过 3）的单词建议给用户。字典通常包含几万个单词，如果将用户输入单词与字典里的单词逐一比对，显然非常耗时。一种高效的处理方法是：结合各单词间的距离，将字典组织成一棵多叉树。在该树中，每个结点表示一个单词，对于每个结点 p，其第 i 个子树包含与结点 p 的距离为 i 的所有单词对应的结点。

树的创建过程如下：取字典中任意单词作为根结点，比如第一个单词。然后将剩余单词逐个插入到树中，插入一个新单词 w 时，首先计算该单词与根结点的距离 d。若根结点的第 d 个子树为空，则将单词 w 对应的结点作为根结点的第 d 个子结点。若根结点的第 d 个子树非空，即根结点已有第 d 个子结点 p_d。则按上述规则将单词 w 递归插入以 p_d 为根的子树，即计算 w 与 p_d 的距离.....。

例如字典为{help, hell, hello, shell, helper, loop, helps, troop}，将 help 作为根结点，然后将 hell 插入，hell 与 help 的距离为 1，故 hell 作为 help 的第一个子结点。hello 与 help 的距离为 2，故 hello 作为 help 的第 2 个子结点。如图 5(a)所示。



(a) (b) (c)

图 5

然后插入 shell，shell 与 help 的距离为 2，故应在 help 的第 2 个子树里，但 help 已经有第 2 个子结点 hello 了，此时将 shell 递归插入以 hello 为根结点的子树：计算 shell 与 hello 的距离为 2，将 shell 作为 hello 的第 3 个子结点，如图 5(b)所示。插入 helper 时，helper 与 help 的距离为 2，将 helper 递归插入以 hello 为根结点的子树：计算 helper 与 hello 的距离为 3，将 helper 作为 hello 的第 3 个子结点。以此类推，最终上述字典对应的树结构如图 5(c)所示。该树结构保证与任意结点距离为 d 的单词都在该结点的第 d 棵子树里。

假定我们需要向用户返回与错误单词距离不超过 n 的单词，当用户输入一个单词 w 时，在树中查询 w，计算 w 与根结点 T 的距离 d，接下来我们不必考察 T 的所有子树中是否包含与 w 距离不超过 n 的结点/单词，而只需要递归考察根结点 T 的第 d-n 到第 d+n 棵子树即可。例如 n=1，d=5，我们只需要递归考察根 T 的第 4、第 5、第 6 棵子树是否包含与 w 距离不超过 1 的结点/单词。其他子树无需考察，为什么呢？举个例子，我们考虑根 T 的第 3 棵子树的任意结点 P。w 与 T 的距离为 d=5，即 w 最少经过 5 步操作才能转换为 T，T 与 P 的距离为 3，T 经过最少 3 步操作才能变为 P，这意味着 w 至少需要 2 步操作才能变为 P。不可能通过 1 步操作变为 P。故第 3 棵子树的所有结点都不满足条件。

$$w \xrightarrow{5} T \xrightarrow{3} P$$

由于 n 通常很小，因此该方法在查询时往往可以排除很多子树，进而节省时间。当考察一个结点时，计算 w 与该结点的距离 d；若 d=0，意味着用户输入的单词 w 在字典中，是正确的单词；若 $d > n$ 则该结点不是候选单词，继续递归考察该结点第 d-n 到 d+n 的子树。若 $d \le n$ 则该结点就是候选单词之一，此时可有两种策略，一是将该单词直接返回给用户，二是继续向下考察子树，找出所有候选单词并选择用户历史使用频率最高的单词返回给用户；

在本题中，请你编写程序实现上述功能。

输入格式：

输入第 1 行为 3 个正整数 n、m、d。n 为字典中单词个数。m 为用户查询数，即用户输入的单词个数。对于用户输入的每个错误单词，程序需要返回与错误单词距离不超过 d 的单词。接下来 n 行，表示字典信息，字典包含 n 个单词及其历史使用频率。每行为 1 个整数和一个由字母组成的字符串，整数表示单词的历史使用频率，字符串表示单词。接下来 m 行，表示用户的查询，每行一个字母组成的字符串，表示用户输入的单词。 ($n \le 10000, m \le 1000, d \le 2$)

输出格式：

对于用户输入的每个单词，若该单词正确（其在字典中），则直接输出该单词；若该单词错误（不在字典中），则输出字典中与该单词距离不超过 d 的所有单词中历史使用频率最高的单词，若多个满足条件的单词使用频率相等，则返回字典序最靠前的单词；若没有满足条件的单词，则输出 No similar word in dictionary。

样例：

输入	输出
9 10 2 327769900 my 322417800 are 302713400 me 283256900 one 282026500 their 280248100 so 264141700 an 263713600 said 250991700 them mb ake me wne therr xxxxx they ax sy sds	my are me are their No similar word in dictionary their my my so