**一. 单选题**

**1.** 以下排序算法（ ）是稳定的。

A. 起泡排序 B. 快速排序 C. 堆排序 D. 直接选择排序

**二．**一个最小最大堆（Min-Max Heap）是一种特定的堆，其最小层和最大层交替出现，根节点总是处于最小。它具有以下性质：对于堆中处于最大（小）层的任一节点，其关键字的值总是在以之为根的子树中的最大（小）关键字；下图是一个最小最大堆的例子，其存储结构为{8, 75, 42, 30, 19, 11, 18, 45,50,30,22, 25}。

45

50

30

22

25

30

19

11

18

75

8

42

最小层

最大层

最小层

最大层

（1）请画出在上图堆后面依次插入关键字为5和80的节点后的最小最大堆。

（2）请说明插入操作的基本过程和插入操作的时间复杂度。

**（1）略**

**（2） 基本思想：从插入的位置开始，从下向上依次进行调整。**

**1、根据元素的位置求出其所在的层数，若为奇数层则在最小层，若为偶数则在最大层。**

**2、若插入元素在最大层，则先比较其关键字是否比其双亲小，如果是则交换。然后将已经形成的小堆与其祖先进行同样的调整，直到满足小堆定义或者到达根节点；若插入元素不小于其双亲，则调整大堆，直到满足大堆定义。**

**若插入元素在最小层，则反之。**

**复杂度 O(logn)**

**三.** 假定采用带头结点的单链表保存单词，当两个单词有相同的后缀时，则可共享相同的后缀存储空间，例如，“loading”和“being”，如下图所示。



设str1和str2分别指向两个单词所在单链表的头结点，链表结点结构为（data, next），请设计一个时间上尽可能高效的算法，找出由str1和str2所指向两个链表共同后缀的起始位置（如图中字符i所在结点的位置p）。要求：

（1）给出算法的基本设计思想。

（2）根据设计思想，采用类C语言的伪代码描述算法，关键之处给出注释。

（3）说明你所设计算法的时复杂度。

**（1）算法思想：顺序遍历两个链表到尾结点时，并不能保证两个链表同时到达尾结点。这是因为两个链表的长度不同。假设一个链表比另一个链表长 k 个结点，先在长链表上遍历k个结点，之后同步遍历两个链表。这样就能够保证它们同时到达最后一个结点。由于两个链表从第一个公共结点到链表的尾结点都是重合的。所以它们肯定同时到达第一个公共结点。于是得到算法思路：**

**① 遍历两个链表求的它们的长度 L1，L2；**

**② 比较 L1，L2，找出较长的链表，并求 k=|L1-L2|；**

**③ 先遍历长链表的k个结点；**

**④ 同步遍历两个链表，直至找到相同结点或链表结束。**

**（2）算法的 类C 语言伪代码描述**

**LinkList Search\_First\_Common(LinkList L1,LinkList L2){**

**//本算法实现线性时间内找到两个单链表的第一个公共结点**

**int len1=Length(L1);**

**len2=Length(L2);**

**LinkList longList,shortlist;//分别指向较长和较短的链表**

**if(len1>len2){**

**longList=L1->next;**

**shortlist=L2->next;**

**k=len1-len2;//表长之差**

**}**

**else{**

**longList=L2->next;**

**shortlist=L1->next;**

**k=len2-len1;//表长之差**

**}**

**while(k--)**

**longList=longList->next;**

**while(longList!=NULL){**

**if(longList==shortList)//同步寻找共同结点**

**return longList;**

**else{**

**longList=longList->next;**

**shortlist=shortlist->next;**

**}**

**}//while**

**return NULL;**

**}**

**（3）算法的时间复杂度为 O(len1+len2)，空间复杂度为 O(1)。**

**四.** 给定一个有n行数字组成的数字三角形, 设计一个算法, 计算出从三角形的顶至底的一条路径, 使该路径经过的数字和最大。要求以下图为例，给出算法的基本思想以及该测试用例的详细求解过程。

7

3 8

8 1 0

2 7 4 4

4 5 2 6 5

**使用动态规划法**

**满足最优子结构性质**

**dp[i][j]表示为从底到第i行第j列能得到的最大分数。**

**dp[i][j] = a[i,j] + max { dp[i+1,j], dp[i+1,j+1]}, 当j<=i**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** |
| **1** | **30** |  |  |  |  |
| **2** | **23** | **21** |  |  |  |
| **3** | **20** | **13** | **10** |  |  |
| **4** | **7** | **12** | **10** | **10** |  |
| **5** | **4** | **5** | **2** | **6** | **5** |

**答案：30**

五．试题库问题：一个试题库中有n道试题。每道试题都标明了所属类别。同一道题可能有多个类别属性。现要从题库中抽取m道题组成试卷.并要求试卷包含指定类型的试题。

数据输入：第1行有2个正整数k和n(2<k<20, k<n<1000), k表示题库中试题类型总数， n表示题库中试题总数。第2行有k个正整数，第i个正整数表示要选出的类型i的题数. 这k个数相加就是要选出的总题数。接下来n行给出了题库中每个试题的类型信息。 每行的第1个正整数p标明该题可以属于p类，接着的p个数是该题所属的类型号。

结果输出:第i行输出“i:”后接类型i的题号。如果有多个满足要求的方案, 只要输出1个方案。 如果问题无解, 则输出“No Solution!”。 给出算法的基本思想.

**解: 构造流网络.**

**顶点构造:**

**构造题型号1:k对应的顶点x[1:k]**

**构造试题号1:n对应的顶点y[1:n]**

**添加源点s, 和汇点t.**

**边的构造:**

**从s各连1条边到k个顶点x[1:k], 容量为题型k需要的题数**

**若试题i属于题型j, 则添边(x[j],y[i]), 容量1**

**对每个试题i, 添加1条边(y[i],t), 容量1**

**使用最大流算法, 得到相应解.**

**六.** 无向图哈密顿圈问题是

UHC={<G>|G是有哈密顿回路的无向图}。

旅行售货员问题是

TSP={<G,s,w,b>| G有从s出发回到s总费用≤b的回路}。

已知UHC是NP完全问题。证明TSP也是NP完全问题。

**证明:**

**(1) TSP∈NP**

**构造如下非确定图灵机**

**N=“对于输入<G,s,w,b>, G是无向图,s是G的一个顶点, w是非负权函数, b是一个整数,G有n个顶点**

**(a) 非确定地产生一个n个顶点的排列**

**(b) 若这个排列对应回路的边都在G中，且回路上的权和小于等于b, 则接受; 否则, 拒绝”.**

**因为N的语言是TSP，且N是在多项式时间运行，所以TSP∈NP。**

**(2) 证明UHC可以多项式时间映射归约到TSP**

**对任意无向图G=(V,E)，设G的顶点数为n。按如下方式构造f(<G>)=<G’,s,w,n>，**

**其中G’=(V,V×V), s属于V, 对于任意vi,vj,**



**这一映射在多项式时间内能完成。**

**下面证<G>∈UHC ⇔ f(<G>)∈TSP**

**若<G>∈UHC，则G中有Hamilton圈c。对于G’, c正好是一个从s出发回到s，经历所有节点，且总费用=n的路径，从而f(<G>)∈TSP。**

**若f(<G>)∈TSP，则G’中有一条路径c从s出发回到s，经历所有节点，且总费用=n的路径。这条路径的边都在G中，是一个Hamilton圈。所以<G>∈UHC。**

**所以f是从UHC到TSP的多项式时间映射归约。**

**由(1)和(2) 及UHC是NP完全问题，TSP是NP完全问题。**