**2013 年全国硕士研究生入学统一考试**

**计算机专业基础综合试题(数据结构部分)**

一、单项选择题：每小题 2 分。

1．求整数 n(n≥0)阶乘的算法如下，其时间复杂度是（ ）

int fact(int n)

{ if (n<=1) return 1;

return n\*fact(n-1);

}

A. O(log2n) B. O(n) C. (nlog2n) D. O(n2)

2．已知操作符包括“+”，“-”，“\*”,“、”，“（”和“）”。将中缀表达式 a+b-a\*((c+ d)/e-f)+g 转换为等价的后缀表达式 ab+acd+e/f-\*-g+ 时，用栈来存放暂时还不能确定运算次序的操作符，若栈初始时为空，则转换过程中同时保存在栈中的操作符的最大个数是（ ）

A. 5 B. 7 C. 8 D. 11

3.若一棵二叉树的前序遍历序列为 a, e, b, d, c，后序遍历序列为 b, c, d, e, a，则根结点的孩子结点（ ）

A. 只有 e B. 有 e、b C. 有 e、c D. 无法确定

4．若平衡二叉树的高度为 6，且所有非叶结点的平衡因子均为 1，则该平衡二叉树的结点总数为（ ）

A. 10 B. 20 C. 32 D. 33

5．对有 n 个结点、e 条边且使用邻接表存储的有向图进行广度优先遍历，其算法时间复杂度是( )

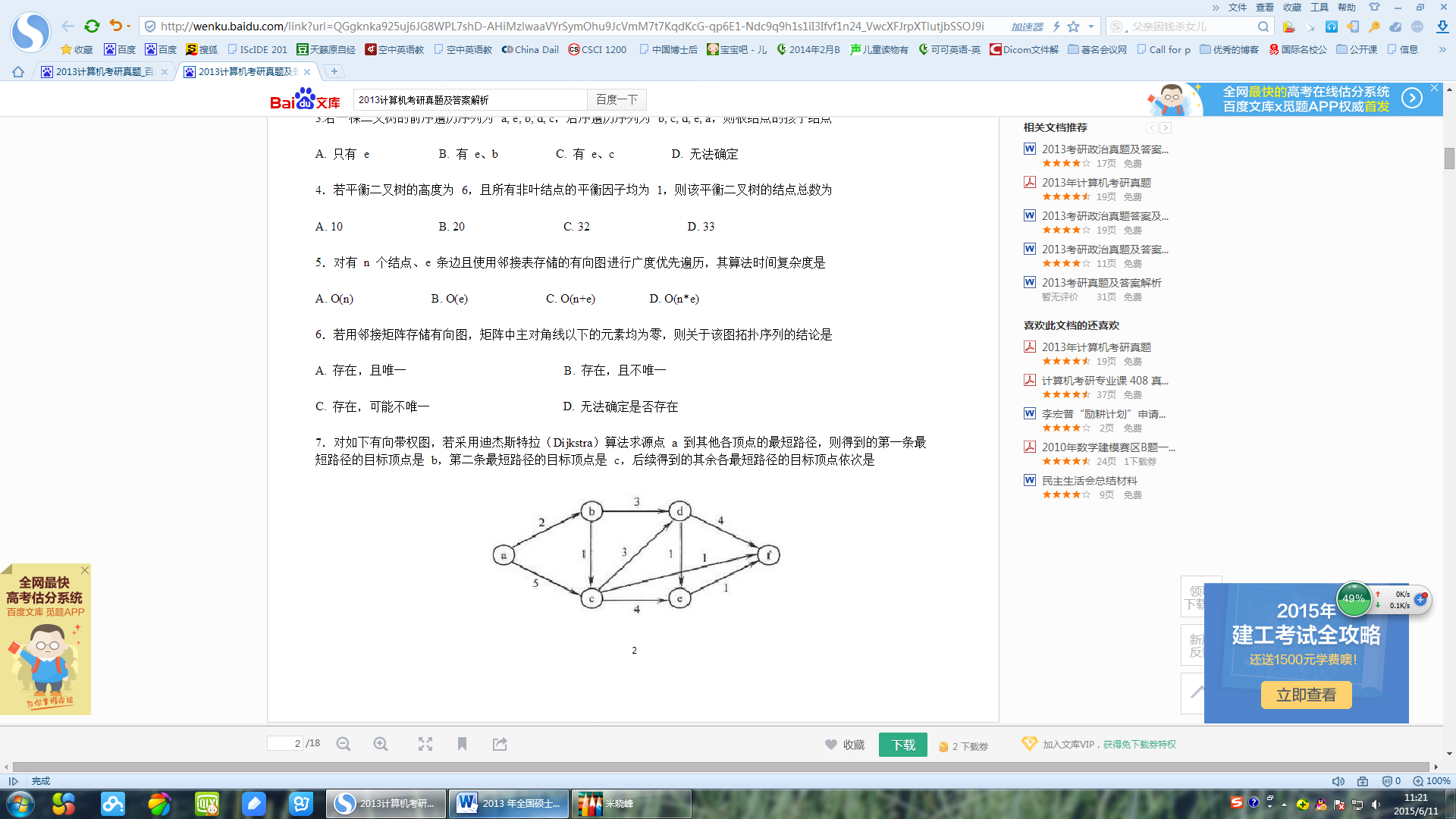
A. O(n) B. O(e) C. O(n+e) D. O(n\*e)

6．若用邻接矩阵存储有向图，矩阵中主对角线以下的元素均为零，则关于该图拓扑序列的结论是( )

A. 存在，且唯一 C. 存在，可能不唯一

B. 存在，且不唯一 D. 无法确定是否存在

7．对如下有向带权图，若采用迪杰斯特拉（Dijkstra）算法求源点 a 到其他各顶点的最短路径，则得到的第一条最短路径的目标顶点是 b，第二条最短路径的目标顶点是 c，后续得到的其余各最短路径的目标顶点依次是( )



A.d,e,f B.e,d,f C. f,d,e D.f,e,d

8．下列关于最小生成树的说法中，正确的是 ( )

I. 最小生成树的代价唯一

II. 权值最小的边一定会出现在所有的最小生成树中

III. 用普里姆（Prim）算法从不同顶点开始得到的最小生成树一定相同

IV. 普里姆算法和克鲁斯卡尔（Kruskal）算法得到的最小生成树总不相同

A. 仅 I B. 仅 II C. 仅 I、III D. 仅 II、IV

9．设有一棵 3 阶 B 树，如下图所示。删除关键字 78 得到一棵新 B 树，其最右叶结点所含的关键字是( )



A. 60 B. 60, 62 C. 62, 65 D. 65

10．在内部排序过程中，对尚未确定最终位置的所有元素进行一遍处理称为一趟排序。下列排序方法中，每一趟排序结束都至少能够确定一个元素最终位置的方法是( )

I. 简单选择排序 II. 希尔排序 III. 快速排序

IV 堆排序 V. 二路归并排序

A. 仅 I、III、IV

B. 仅 I、III、V

C. 仅 II、III、IV

D. 仅 III、IV、V

11．对一待排序序列分别进行折半插入排序和直接插入排序，两者之间可能的不同之处是( )

A. 排序的总趟数 C. 使用辅助空间的数量

B. 元素的移动次数 D. 元素之间的比较次数

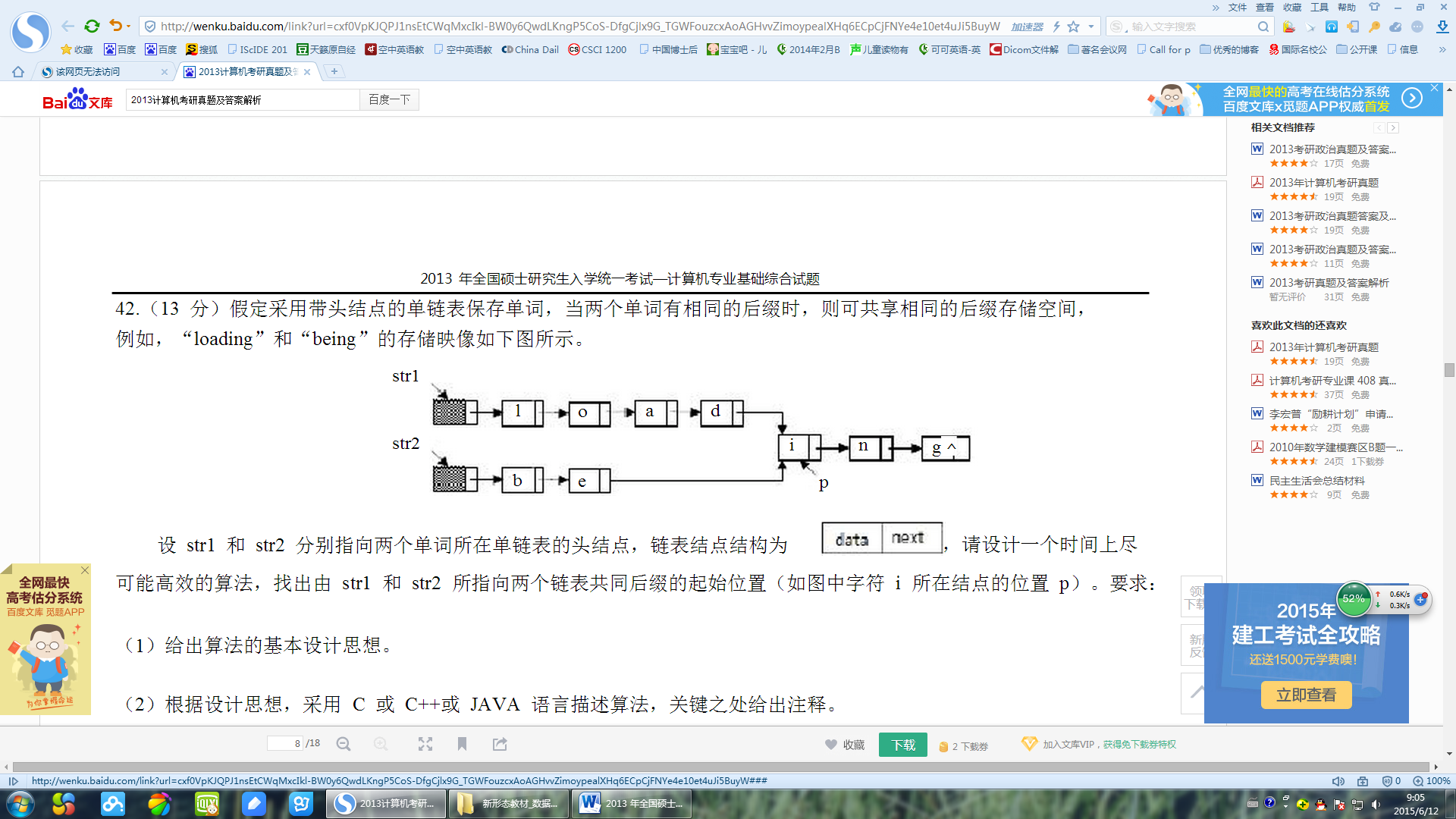
二. 综合题

1.（10 分）设有 6 个有序表 A、B、C、D、E、F，分别含有 10、35、40、50、60 和 200 个数据元素，各表中元素按升序排列。要求通过 5 次两两合并，将 6 个表最终合并成 1 个升序表，并在最坏情况下比较的总次数达到最小。 请问答下列问题。

（1）给出完整的合并过程，并求出最坏情况下比较的总次数。

（2）根据你的合并过程，描述 n（n≥2）个不等长升序表的合并策略，并说明理由。

2.（13 分）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”的存储映像如下图所示。



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 C 或 C++或 JAVA 语言描述算法，关键之处给出注释。

（3）说明你所设计算法的时间复杂度。

一、单项选择题：每小题 2 分。

1~5：BAABC

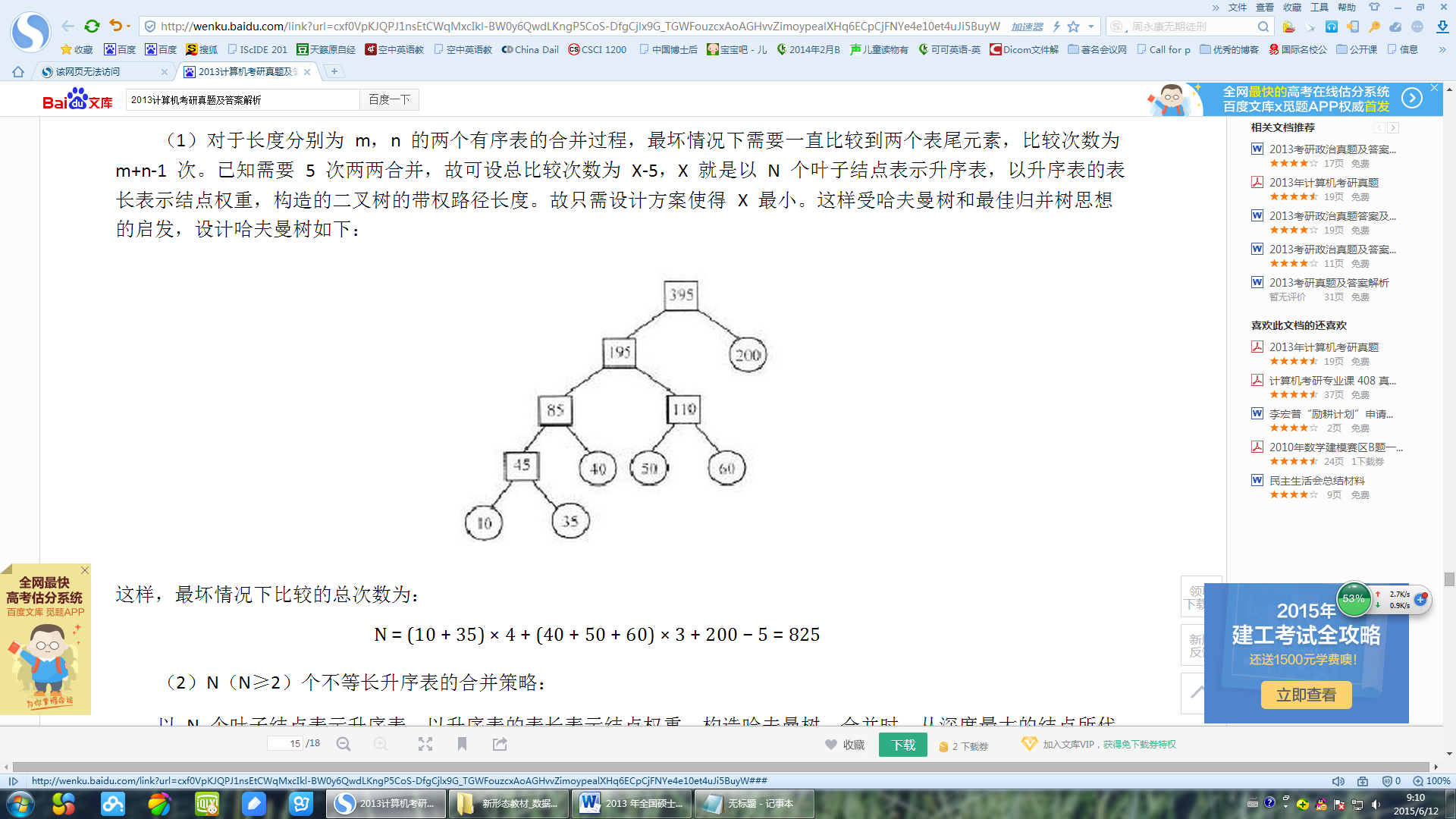
6~10：ACADA

11：D

二、综合题。

1. （10分）

（1）对于长度分别为 m，n 的两个有序表的合并过程，最坏情况下需要一直比较到两个表尾元素，比较次数为 m+n-1 次。已知需要 5 次两两合并，故可设总比较次数为 X-5，X 就是以 N 个叶子结点表示升序表，以升序表的表长表示结点权重，构造的二叉树的带权路径长度。故只需设计方案使得 X 最小。这样受哈夫曼树和最佳归并树思想的启发，设计哈夫曼树如下：



这样，最坏情况下比较的总次数为：

N = (10 + 35) × 4 + (40 + 50 + 60) × 3 + 200 − 5 = 825

（2）N（N≥2）个不等长升序表的合并策略： 以 N 个叶子结点表示升序表，以升序表的表长表示结点权重，构造哈夫曼树。合并时，从深度最大的结点所代 表的升序表开始合并，依深度次序一直进行到根结点。

理由：N 个有序表合并需要进行 N-1 次两两合并，可设最坏情况下的比较总次数为 X-N+1，X 就是以 N 个叶子 结点表示升序表，以升序表的表长表示结点权重，构造的二叉树的带权路径长度。根据哈夫曼树的特点，上述设计的比较次数是最小的。

2.（13分）

（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，我们先在长链表上遍历 k 个结点，之后同步遍历两个链表。这 样我们就能够保证它们同时到达最后一个结点了。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 L=|L1-L2|；

③ 先遍历长链表的 前L 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 C 语言代码描述：

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; L=len1-len2;//表长之差

}

else{

longList=L2->next; shortlist=L1->next; L=len2-len1;//表长之差

}

while(L--)

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; }

}

return NULL;

}

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