武汉大学计算机学院

2010 年－2011 学年第二学期“数据结构”考试试题（A）

要求：所有的题目的解答均写在答题纸上，需写清楚题目的序号。每张答题纸都要写上姓名和学号。

一、单项选择题（每小题 1.5 分，共计 30 分）

1. 数据结构是指 D 。

A. 一种数据类型

B. 数据的存储结构

C. 一组性质相同的数据元素的集合

D. 相互之间存在一种或多种特定关系的数据元素的集合

2. 以下算法的时间复杂度为 A 。

```
void fun(int n)
{  int i=1;
   while (i<=n)
      i++;
}
```

A. O(n)　　B. O($\sqrt{n}$)

C. O(nlog$_2$n)　　D. O(log$_2$n)

3. 在一个长度为 n 的有序顺序表中删除元素值为 x 的元素时，在查找元素 x 时采用二分查找，此时的时间复杂度为 B 。

A. O(n)　　B. O(nlog$_2$n)

C. O(n^2)　　D. O($\sqrt{n}$)

4. 在一个带头结点的循环单链表 L 中，删除元素值为 x 的结点，算法的时间复杂度为 A 。

A. O(n)　　B. O($\sqrt{n}$)

C. O(nlog$_2$n)　　D. O(n^2)

5. 若一个栈采用数组 s[0..n-1]存放其元素，初始时栈顶指针为 n，则以下元素 x 进栈的正确操作是 C 。

A.top++;s[top]=x;　　B.s[top]=x;top++;

C.top--;s[top]=x;　　B.s[top]=x;top--;

6. 中缀表达式“2*(3+4)−1”的后缀表达式是 B ，其中#表示一个数值的结束。

A. 2#3#4#1#*+−　　B. 2#3#4#+*1#−

C. 2#3#4#*+1#−　　D. −+*2#3#4#1#

7. 设环形队列中数组的下标为 0～N−1，其队头、队尾指针分别为 front 和 rear（front 指向队列中队头元素的前一个位置，rear 指向队尾元素的位置），则其元素个数为 D 。

A. rear−front　　B. rear−front−1

C. (rear−front)％N+1　　D. (rear−front+N)％N

8. 若用一个大小为 6 的数组来实现环形队列，队头指针 front 指向队列中队头元素的前一个位置，队尾指针 rear 指向队尾元素的位置。若当前 rear 和 front 的值分别为 0 和 3，当从队列中删除一个元素，再加入两个元素后，rear 和 front 的值分别为 B 。

A. 1 和 5　　B. 2 和 4
C. 4 和 2　　D. 5 和 1

9. 一棵深度为 h（h≥1）的完全二叉树至少有 A 个结点。
A. 2^{h-1}　　B. 2^{h}
C. $2^{h}+1$　　D. $2^{h-1}+1$

10. 一棵含有 n 个结点的线索二叉树中，其线索个数为 C 。
A. 2n　　B. n−1
C. n+1　　D. n

11. 设一棵哈夫曼树中有 1999 个结点，该哈夫曼树用于对 C 个字符进行编码。
A. 999　　B. 998
C. 1000　　D. 1001

12. 一个含有 n 个顶点的无向连通图采用邻接矩阵存储，则该矩阵一定是 A 。
A. 对称矩阵　　B. 非对称矩阵
C. 稀疏矩阵　　D. 稠密矩阵

13. 设无向连通图有 n 个顶点 e 条边，若满足 A ，则图中一定有回路。
A. e≥n　　B. e<n
C. e=n−1　　D. 2e≥n

14. 对于 AOE 网的关键路径，以下叙述 D 是正确的。
A. 任何一个关键活动提前完成，则整个工程一定会提前完成
B. 完成整个工程的最短时间是从源点到汇点的最短路径长度
C. 一个 AOE 网的关键路径一定是唯一的
D. 任何一个活动持续时间的改变可能会影响关键路径的改变

15. 设有 100 个元素的有序表，用折半查找时，不成功时最大的比较次数是 D 。
A. 25　　B. 50
C. 10　　D. 7

16. 在一棵 m 阶 B-树中删除一个关键字会引起合并，则该结点原有 C 个关键字。
A. 1　　B.⌈m/2⌉
C.⌈m/2⌉−1　　D.⌈m/2⌉+1

17. 哈希查找方法一般适用于 D 情况下的查找。
A. 查找表为链表
B. 查找表为有序表
C. 关键字集合比地址集合大得多
D. 关键字集合与地址集合之间存在着某种对应关系。

18. 对含有 n 个元素的顺序表采用直接插入排序方法进行排序，在最好情况下算法的时间复杂度为 A 。
A. O(n)　　B. $O(n\log_2 n)$
C. $O(n^2)$　　D. $O(\sqrt{n})$

19. 用某种排序方法对数据序列{24,88,21,48,15,27,69,35,20}进行递增排序，元素序列的变化情况如下：

（1）{24,88,21,48,15,27,69,35,20}
（2）{20,15,21,24,48,27,69,35,88}
（3）{15,20,21,24,35,27,48,69,88}
（4）{15,20,21,24,27,35,48,69,88}
则所采用的排序方法是___A___。

A. 快速排序　　B. 简单选择排序
C. 直接插入排序　　D. 归并排序

20. 以下序列是堆的是___C___。

A. {75,65,30,15,25,45,20,10}　　B. {75,65,45,10,30,25,20,15}
C. {75,45,65,30,15,25,20,10}　　D. {75,45,65,10,25,30,20,15}

二、问答题（共 4 小题，每小题 10 分，共计 40 分）

1. 如果对含有 n（n>1）个元素的线性表的运算只有 4 种：删除第一个元素；删除最后一个元素；在第一个元素前面插入新元素；在最后一个元素的后面插入新元素，则最好使用以下哪种存储结构，并简要说明理由。

（1）只有尾结点指针没有头结点指针的循环单链表
（2）只有尾结点指针没有头结点指针的非循环双链表
（3）只有头结点指针没有尾结点指针的循环双链表
（4）既有头结点指针也有尾结点指针的循环单链表

2. 已知一棵度为 4 的树中，其度为 0、1、2、3 的结点数分别为 14、4、3、2，求该树的结点总数 *n* 和度为 4 的结点个数，并给出推导过程。

3. 有人提出这样的一种从图 G 中顶点 u 开始构造最小生成树的方法：

假设 G=(V，E)是一个具有 n 个顶点的带权连通无向图，T=(U，TE)是 G 的最小生成树，其中 U 是 T 的顶点集，TE 是 T 的边集，则由 G 构造从起始顶点 u 出发的最小生成树 T 的步骤如下：

（1）初始化 U={u}。以 u 到其他顶点的所有边为候选边。
（2）重复以下步骤 n−1 次，使得其他 n−1 个顶点被加入到 U 中。
从候选边中挑选权值最小的边加入到 TE，设该边在 V−U 中的顶点是 v，将 v 加入 U 中。
考查顶点 v，将 v 与 V−U 顶点集中的所有边作为新的候选边。
若此方法求得的 T 是最小生成树，请予以证明。若不能求得最小边，请举出反例。

4. 有一棵二叉排序树按先序遍历得到的序列为：(12,5,2,8,6,10,16,15,18,20)。回答以下问题：

（1）画出该二叉排序树。
（2）给出该二叉排序树的中序遍历序列。
（3）求在等概率下的查找成功和不成功情况下的平均查找长度。

三、算法设计题（每小题 10 分，共计 30 分）

1. 设 A 和 B 是两个结点个数分别为 m 和 n 的单链表（带头结点），其中元素递增有序。设计一个尽可能高效的算法求 A 和 B 的交集，要求不破坏 A、B 的结点，将交集存放在单链表 C 中。给出你所设计的算法的时间复杂度和空间复杂度。

2. 假设二叉树 b 采用二叉链存储结构，设计一个算法 void findparent(BTNode *b,ElemType x,BTNode *&p)求指定值为 x 的结点的双亲结点 p，提示，根结点的双亲为 NULL，若在 b 中未找到值为 x 的结点，p 亦为 NULL。

3. 假设一个连通图采用邻接表 G 存储结构表示。设计一个算法，求起点 u 到终点 v 的经过顶点 k 的所有路径。

四、附加题（10 分）

说明：附加题不计入期未考试总分，但计入本课程的总分。

假设某专业有若干个班，每个班有若干学生，每个学生包含姓名和分数，这样构成一棵树，如图 1 所示。假设树中每个结点的 name 域均不相同，该树采用孩子兄弟链存储结构，其结点类型定义如下：

```
typedef struct node
{ char name[50];           //专业、班号或姓名
  float score;             //分数
  struct node *child;        //指向最左边的孩子结点
  struct node *brother;      //指向下一个兄弟结点
} TNode;
```

完成以下算法：

（1）设计一个算法求所有的学生人数。

（2）求指定某班的平均分。

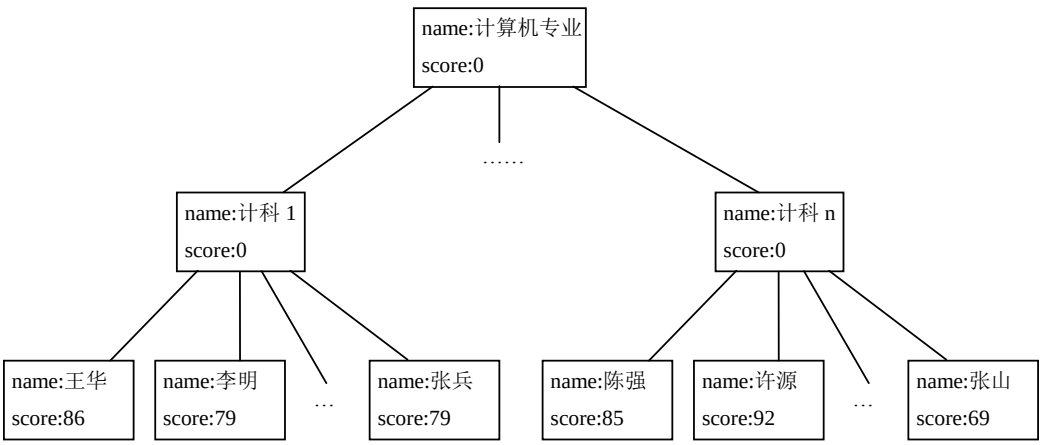

图 1 一棵学生成绩树

“数据结构”考试试题（A）参考答案

要求：所有的题目的解答均写在答题纸上，需写清楚题目的序号。每张答题纸都要写上姓名和学号。

一、单项选择题（每小题 1.5 分，共计 30 分）

1. D　　2. A　　3. B　　4. A　　5. C
6. B　　7. D　　8. B　　9. A　　10. C
11. C　12. A　13. A　14. D　15. D
16. C　17. D　18. A　19. A　20. C

二、问答题（共 4 小题，每小题 10 分，共计 40 分）

1. 答：本题答案为（3），因为实现上述 4 种运算的时间复杂度均为 O(1)。

【评分说明】选择结果占 4 分，理由占 6 分。若结果错误，但对各操作时间复杂度作了分析，可给 2～5 分。

2. 答：结点总数 $n=n_0+n_1+n_2+n_3+n_4$，即 $n=23+n_4$，又有：度之和 $=n-1=0\times n_0+1\times n_1+2\times n_2+3\times n_3+4\times n_4$，即 $n=17+4n_4$，综合两式得：$n_4=2$，$n=25$。所以，该树的结点总数为 25，度为 4 的结点个数为 2。

【评分说明】结果为 4 分，过程占 6 分。

3. 答：此方法不能求得最小生成树。例如，对于如图 5.1（a）所示的带权连通无向图，按照上述方法从顶点 0 开始求得的结果为 5.1（b）所示的树，显然它不是最小生成树，正确的最小生成树如图 5.1（c）所示。

在有些情况下，上述方法无法求得结果，例如对于如图 5.1（d）所示的带权连通无向图，从顶点 0 出发，找到顶点 1（边（0,1））,从顶点 1 出发，找到顶点 3（边（1,3））,再从顶点 3 出发，找到顶点 0（边（3,0））,这样构成回路，就不能求得最小生成树了。

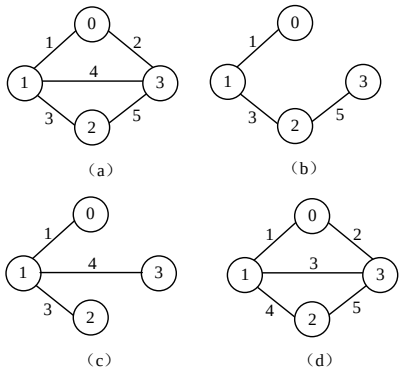

图 1　求最小生成树的反例

说明：只需给出一种情况即可。

【评分说明】回答不能求得最小生成树得 5 分，反例为 5 分。若指出可求得最小生成树，根据证明过程给 1～2 分。

4. 答：（1）先序遍历得到的序列为：(12,5,2,8,6,10,16,15,18,20)，中序序列是一个有序序列，所以为：(2,5,6,8,10,12,15,16,18,20)，由先序序列和中序序列可以构造出对应的二叉树，如图 2 所示。

（2）中序遍历序列为：2,5,6,8,10,12,15,16,18,20。

（3）$ASL_{成功}=(1\times1+2\times2+4\times3+3\times4)/10=29/10$。

$ASL_{不成功}=(5\times3+6\times4/11=39/11$。

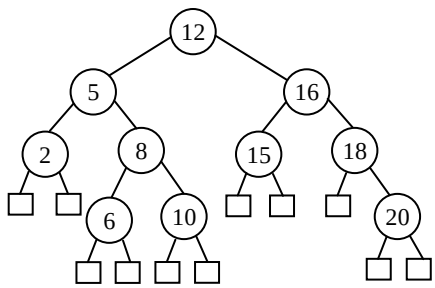

图 2

【评分说明】（1）小题占 6 分，（2）（3）小题各占 2 分。

三、算法设计题（每小题 10 分，共计 30 分）

1. 设 A 和 B 是两个结点个数分别为 m 和 n 的单链表（带头结点），其中元素递增有序。设计一个尽可能高效的算法求 A 和 B 的交集，要求不破坏 A、B 的结点，将交集存放在单链表 C 中。给出你所设计的算法的时间复杂度和空间复杂度。

解：算法如下：

```
void insertion(LinkList *A,LinkList *B,LinkList *&C)
{ LinkList *p=A->next,*q=B->next,*s,*t;
  C=(LinkList *)malloc(sizeof(LinkList));
  t=C;
  while (p!=NULL && q!=NULL)
  {  if (p->data==q->data)
     {  s=(LinkList *)malloc(sizeof(LinkList));
        s->data=p->data;
        t->next=s;
        t=s;
        p=p->next;
        q=q->next;
     }
     else if (p->data<q->data)
        p=p->next;
     else
        q=q->next;
  }
  t->next=NULL;
}
```

算法的时间复杂度为 O(m+n)，空间复杂度为 O(MIN(m,n))。

【评分说明】算法为 8 分，算法的时间复杂度和空间复杂度各占 1 分。

2. 假设二叉树 b 采用二叉链存储结构，设计一个算法 void findparent(BTNode *b,ElemType x,BTNode *&p)求指定值为 x 的结点的双亲结点 p，提示，根结点的双亲为 NULL，若未找到这样的结点，p 亦为 NULL。

解：算法如下：

```
void findparent(BTNode *b,ElemType x,BTNode *&p)
{ if (b!=NULL)
  {  if (b->data==x) p=NULL;
     else if (b->lchild!=NULL && b->lchild->data==x)
        p=b;
     else if (b->rchild!=NULL && b->rchild->data==x)
        p=b;
     else
     {  findparent(b->lchild,x,p);
        if (p==NULL)
           findparent(b->rchild,x,p);
     }
  }
  else p=NULL;
```

```
}
```

【评分说明】本题有多种解法，相应给分。

3. 假设一个连通图采用邻接表 G 存储结构表示。设计一个算法，求起点 u 到终点 v 的经过顶点 k 的所有路径。

解：算法如下：

```
int visited[MAXV]={0};      //全局变量
 void PathAll(ALGraph *G,int u,int v,int k,int path[],int d)
 //d 是到当前为止已走过的路径长度，调用时初值为-1
 { int m,i;
   ArcNode *p;
   visited[u]=1;
   d++;                        //路径长度增 1
   path[d]=u;                     //将当前顶点添加到路径中
   if (u==v && In(path,d,k)==1) //输出一条路径
   {  printf("  ");
      for (i=0;i<=d;i++)
         printf("%d ",path[i]);
      printf("\n");
   }
   p=G->adjlist[u].firstarc;  //p 指向顶点 u 的第一条弧的弧头节点
   while (p!=NULL)
   {  m=p->adjvex;            //m 为 u 的邻接点
      if (visited[m]==0)         //若该顶点未标记访问,则递归访问之
         PathAll(G,m,v,l,path,d);
      p=p->nextarc;           //找 u 的下一个邻接点
   }
   visited[u]=0;              //恢复环境：使该顶点可重新使用
 }
 int In(int path[],int d,int k) //判断顶点 k 是否包含在路径中
 { int i;
   for (i=0;i<=d;i++)
      if (path[i]==k)
         return 1;
   return 0;
 }
```

【评分说明】本题采用 DFS 算法给出一条路径时给 8 分，采用 BFS 算法给出一条路径时给 6 分。

四、附加题（10 分）

说明：附加题不计入期未考试总分，但计入本课程的总分。

假设某专业有若干个班，每个班有若干学生，每个学生包含姓名和分数，这样构成一棵树，如图 1 所示。假设树中每个结点的 name 域均不相同，该树采用孩子兄弟链存储结构，其结点类型定义如下：

```
typedef struct node
{ char name[50];           //专业、班号或姓名
  float score;             //分数
```

```
  struct node *child;          //指向最左边的孩子结点
  struct node *brother;        //指向下一个兄弟结点
} TNode;
```

完成以下算法：

（1）设计一个算法求所有的学生人数。

（2）求指定某班的平均分。

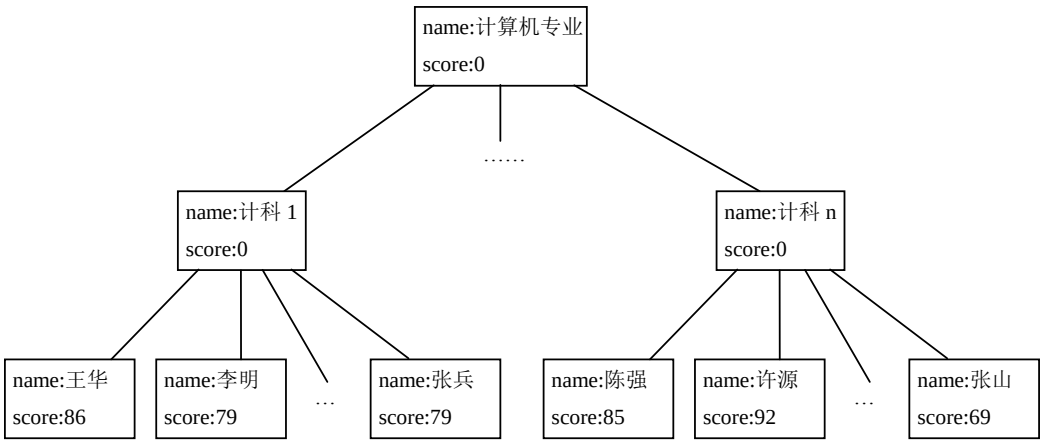

图 1 一棵学生成绩树

解：（1）算法如下：

```
int count(TNode *b)
{
  if (b==NULL) return 0;
  if (b->child==NULL) return 1;
  return count(b->child)+count(b->brother);
}
```

说明：本题可以从链表的角度求解。

（2）算法如下：

```
int average(TNode *b,char class[],float &avg)
{
  int n=0;
  float sum=0;
  TNode *p=b->child;       //p 指向班号结点
  while (p!=NULL && strcmp(p->name,class)!=0)
     p=p->brother;
  if (p==NULL) return 0;  //没找到该班号，返回 0
  p=p->child;             //p 指向该班的第一个学生
  while (p!=NULL)
  {
     n++;               //累计人数
     sum+=p->score;     //累计分数
     p=p->brother;
  }
  avg=sum/n;             //求平均分
```

```
  return 1;
}
```
【评分说明】两小题各占 5 分。