哈尔滨工业大学

2009 年春季学期数据结构与算法 试卷 A 参考答案

一、填空题（每空 2 分，共 20 分）

1.等概率　2. 15　3. 冒泡　4. EFH　5. (m+1)%N　6．51
7. n(n-1)/2　8. m-1　9. 2(n-1)　10. 平衡归并分类　多阶段归并分类

二、选择题（每小题 1 分，共 10 分）

1.C　2. A　3.B　4.C　5.C
6. C　7. A　8. D　9. B　10. D

三、简答题（10 分）

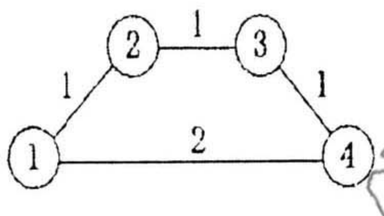

图 A-5

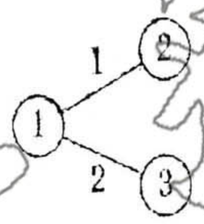

图 A-6

该方法不一定能（或不能）求得最短路径。（4 分）
举例说明：（6 分）图 A-5 中，设初始顶点为 1，目标顶点为 4，欲求从顶点 1 到顶点 4 之间的最短路径，显然这两点之间的最短路径长度为 2。利用给定方法求得的路径长度为 3，但这条路径并不是这两点之间的最短路径。图 A-6 中，设初始顶点为 1，目标顶点为 3，欲求从顶点 1 到顶点 3 之间的最短路径。利用给定的方法，无法求出顶点 1 到顶点 3 的路径。
【评分说明】(注：2009 年计算机统考题目)
①若考生回答“能求得最短路径”，无论给出何种证明，均不给分。
②考生只要举出类似上述的一个反例说明“不能求得最短路径”或答案中体现了“局部最优不等于全局最优”的思想，均可给 6 分；若举例说明不完全正确，可酌情给分。

四、算法设计（每题 10 分，共 30 分）

1.

```
int IsSearchTree(const BTNode *t){
    if (!t)        //空二叉树情况
        return 1;
    else if (!(t->lchild) && !(t->rchild))//左右子树都无情况
        return 1;
    else if ((t->lchild) && !(t->rchild)){//只有左子树情况
        if (t->lchild->data>t->data)
            return 0;
        else
            return IsSearchTree(t->lchild);
    }
    else if ((t->rchild) && !(t->lchild)){//只有右子树情况
        if (t->rchild->data<t->data)
            return 0;
```

```
        else
            return IsSearchTree(t->rchild);
    }
    else{                                   //左右子树全有情况
        if ((t->lchild->data>t->data) || (t->rchild->data<t->data))
            return 0;
        else
            return IsSearchTree(t->lchild) && IsSearchTree(t->rchild);
    }
}
```

2.

```
bool LocateNode(DLinkList *h, ElemType x)
{
    DLinkList *p = h->next, *q;
    while (p != NULL && p->data != x)
        p = p->next;     //找data域值为x的节点*p
    if (p == NULL)       //未找到的情况
        return false;
    else             //找到的情况
    {
        p->freq++;       //频度增1
        q = p->prior;        //*q为p前驱节点
        while (q != h && q->freq<p->freq)
        {
            p->prior = q->prior; p->prior->next = p;//交换*p和*q的位置
            q->next = p->next;
            if (q->next != NULL)//*p不为最后一个节点时
                q->next->prior = q;
            p->next = q; q->prior = p;
            q = p->prior;        //q重指向*p的前趋节点
        }
        return true;
    }
}
```

3. 采用广度优先遍历，其邻接点均已遍历的结点是叶子结点，记下结点的半径(以分枝个数记)

```
int MiniRadius(AdjList g, int v)  //图g以邻接表形式存储，求半径最小的生成树。设顶点
//信息就是编号，从顶点v开始遍历
{
    typedef struct
    {
        int v, level;
    }node; //队列元素
    int MAX = 100; //设最大层次数
```

```
int visited[MAX] = 0; //访问数组
node R, Q[]; //Q为队列，容量足够大
R.v = v;
R.level = 1;
Makenullt(Q);
EnQueue(Q, R);
while (!Empty(Q)
{
    R = DeQueue(Q);     //出队
    v = R.v;
    l = R.level;
    p = g[v].firsteage;
    flag = 0;   //flag是顶点是否是叶子的标记
    while (p)
    {
        w = p->adjvex;
        if (visited[w] == 0)
        {
            flag = 1;
            R.level = l + 1;
            EnQueue(Q, R);
        }
        p = p->next;
    } if (flag == 0)  //其邻接点均已遍历的顶点是叶子结点
    {
        if (l<MAX) MAX = l;
    }
}
return MAX;
}
```