《数据结构A类》（A卷）笔试试卷参考答案

一、单项选择题（每格1.5分，共24分）

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

10. A 11. A 12. D 13. B 14. C 15. D 16. B

二、程序填充题（每格1.5分，共24分）

1. p!=NULL !found f = p; t != p t = NULL;

2. ptr != NULL !stack.empty() ptr != NULL ptr = ptr->left; ptr = ptr->right;

3. flag = false; i+=2 t = a[i+1]; i+=2 t = a[i+1]; !flag

三、简答题（每题8分，共24分）

1. 应用置换选择，一共生成3个初始排序片段，具体如下：（2分）

1) 2, 5, 10, 23, 34, 54（2分）

2) 3, 4, 7, 12, 26, 33, 40（2分）

3) 1, 11, 15, 18, 27, 35（2分）

2. 该图的邻接表为：（4分）

3

5

0

2

3

2

|  |  |
| --- | --- |
| 0 |  |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |

该图共有4个强连通分量。（4分）

3. 该棵二叉树为：

****

四、分析题（20分）

1. 在邻接矩阵类中实现Kruskal函数由两个步骤组成：

1) 将所有的边加入优先级队列，这是通过扫描整个邻接矩阵完成的，所需的时间为 O(|V|2)。（3分）

2) 执行|V|-1次归并，这是通过处理|E|条边实现的。每次处理一条边时，先检查边的两个端点是否连通，即对两个端点分别调用find操作。在不相交集中，有|V|个结点，find操作的时间复杂度是O(log|V|)。如果两个端点不连通，则调用Union归并这两个子集。Union函数的时间复杂度是O(1)的，所以执行归并所需的时间是O(|E|\*log|V|)。（3分）

3) 总的时间复杂度就是 O(|V|2 + |E|\*log|V|)。（2分）

2. //寻找最短路径（1分）

//更新min.node的邻接点距离（1分）

//保留经过结点最少的那条路径（1分）

该算法的运行时间主要由两部分组成：

1. 查找路径最短的尚未在S中的结点（设为u），以及扫描u的邻接点，更新它们的距离。利用优先级队列存放源点到所有结点的距离，每次找出距离最短的结点所需的时间为O(log|V|)，在整个算法的执行过程中，寻找距离最短的结点将花去O(Vlog|V|)的时间。（2分）
2. 更新V-S中的结点的距离所需的时间是O(|E|)，所以总的时间复杂度是O(|E|+Vlog|V|) = O(Vlog|V|)。（2分）

该算法运行结果如下：（5分）

b to a:a-d-c-b Length:13

b to b:b Length:0

b to c:b-d-c Length:9

b to d:b-d Length:7

b to e:b-e Length:2

b to f:b-d-f Length:10

b to g:b-e-g Length:6

五、程序题（8分）

对于最小化堆来讲，叶子上的结点之值大于等于内部结点之值。因此，挑选最大结点只需在叶子上挑选一个最大的即可。即：从下标为⎣n/2」+1到下标为n的结点，进行一次比较。（2分）

int max(int a[], int size) {

int m = size/2; //（2分）

for (int k = size/2; k < size; ++k) //（1分）

if (a[k] > a[m]) m = k; //（2分）

return m; //（1分）

}

六、附加题（10分）

1) 算法思想（2分）

定义一个大小为n的布尔数组，初始化为false。如果被访问结点的元素值a，flag[|a|]为false，则保留此结点，并设flag[|a|]为true，否则删除该结点。

2) 结点的数据结构定义如下（6分）

typedef struct Node { //（1分）

int data;

struct Node \* next;

} Node;

bool flag[n]; //全局数组，标志结点的绝对值的值是否出现过（1分）

void DeleteABSEqualNode (Node \* head) {

memset (flag, false, n); //初始化为false

if (head == NULL) return NULL; //（1分）

Node \* p = head; Node \* r = head;

while (p != NULL) { //（3分）

if (flag[abs(p->data)]) { //如此绝对值已出现过

r->next = p->next;

delete p;

p = r->next;

} //则删除当前结点

else {

flag[abs(p->data)] = true;

r = p;

p = p->next;

} //否则，将数组中对应的元素置true，并将指针指向下一个元素

}

return head;

}

3) 只遍历一次链表，所以时间复杂度为O(m)；因为申请大小为n（n为节点绝对值的最大值）的数组，所以空间复杂度为O(n)。（2分）