

《数据结构 (A 类)》(A 卷) 参考答案

一、单项选择题 (每格 1.5 分, 共 24 分)

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

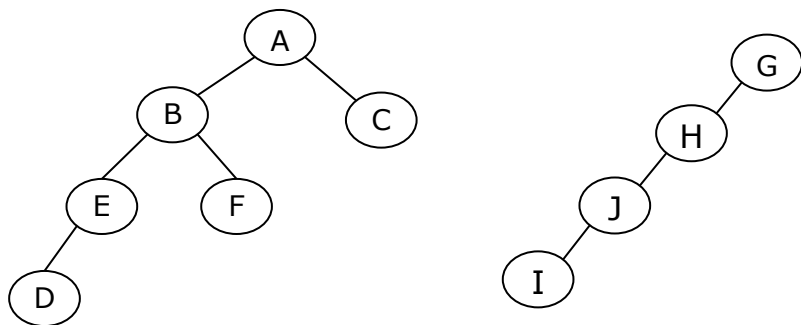
二、程序填充题 (每格 1.5 分, 共 24 分)

1.
 - 1) `p->data > x`
 - 2) `f->left = p->right`
 - 3) `delete p`
 - 4) `s->right`
 - 5) `s == p->left`
 - 6) `delete s`
2.
 - 1) `(low + high) / 2`
 - 2) `high = mid - 1`
 - 3) `low = mid + 1`
3.
 - 1) -1 (与第二个空格对应, 如-1 对应 99、0 对应 100)
 - 2) `stack.top == 99`
 - 3) `++stack.top`
 - 4) `stack.s[stack.top] = x`
4.
 - 1) `p == NULL`
 - 2) `p` 或 `p != NULL`
 - 3) `q`

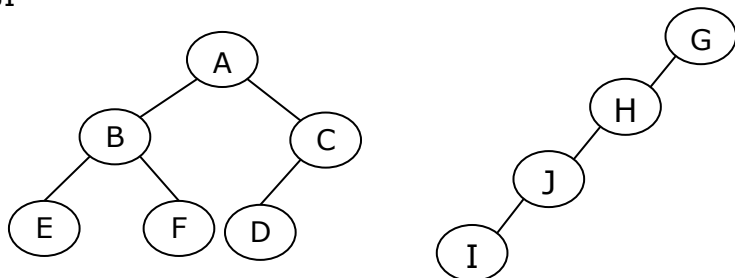
三、简答题 (每题 8 分, 共 24 分)

1. 每个序列各 2 分, 每个图各 1 分, 共 8 分。

DFS 序列: ABEDFCGHJI

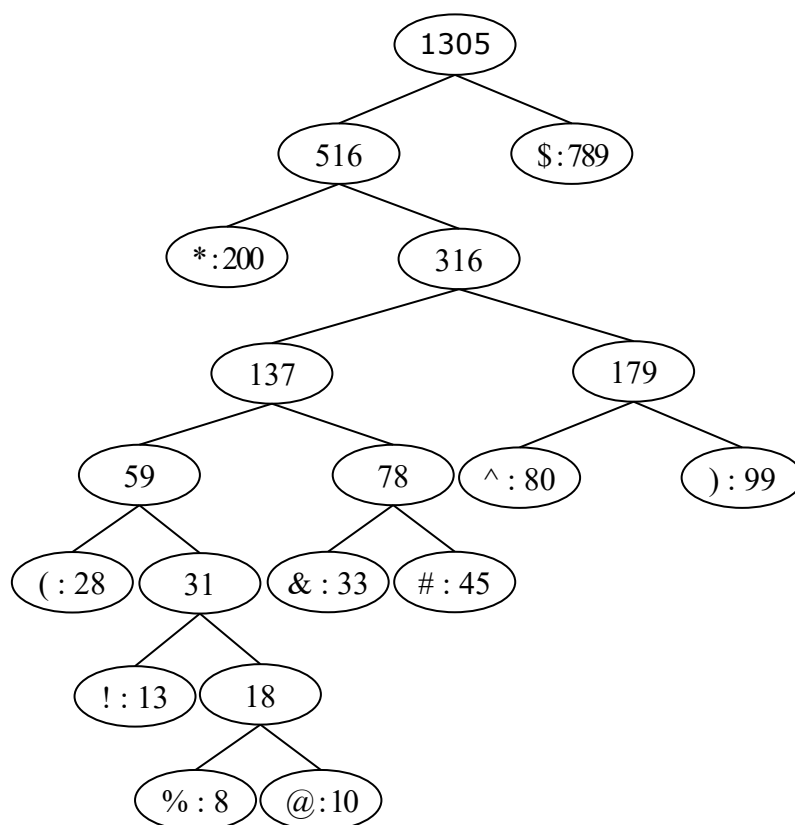


BFS 序列: ABCEFDGHJI



2. 哈夫曼树得 4 分，10 个字符的哈夫曼编码得 4 分（每个 0.4 分）。
构造哈夫曼树的要求：对于新构造的结点，其左儿子的权值较小，右儿子的权值较大。

! : 010010 @ : 0100111 # : 01011 \$: 1
% : 0100110 ^ : 0110 & : 01010 * : 00
(: 01000) : 0111



3. 两小题各 4 分。
1) 35: 探测序列 9、1、2，共 3 次
20: 探测序列 7、4，共 2 次
33: 探测序列 7，共 1 次
48: 探测序列 9，共 1 次
2) 平均比较长度: $(3+2+1+1)/4=7/4$ 或 $(3+2+1+1+2)/5=9/5$

四、分析题（每题 8 分，共 16 分）

- 1) 2 分: 排序
2) 3 分: ++x, --y
3) 3 分: 最坏情况 $O(n^2)$, 最好情况 $O(n \log_2 n)$
- 1) 2 分: p 指向最后一个结点
2) 3 分: 将第一个结点作为链表最后一个结点
3) 3 分: $(a_2, a_3, \dots, a_n, a_1)$

五、设计题（12 分）

1) 设计思路（8 分）

- a) 2 分：首先定义一个大小为 k 的最小化堆，用数组的前 k 个元素组成一个最小化堆
- b) 2 分：对剩余的数组中的每个元素进行如下处理：如果当前元素比堆顶元素大，则删除堆顶元素，添加当前元素进堆
- c) 2 分：最后堆顶元素即为第 k 大值。创建一个大小为 k 堆时间复杂度为 $O(k)$
- d) 2 分：在一个大小为 k 的堆中添加或删除元素时间复杂度都是 $O(\log k)$ ，所有总的时间复杂度为 $O(k) + (N-k)O(\log k) = O(N \log k)$

2) 伪代码（4 分）

```
template <typename T>
T Getkth(T a[], int N, int K) {
    //利用优先队列取数组 a 前 k 个元素建立一个最小化堆 H;
    for (int i = k; i < N; i++) {           //for 循环（3 分）
        if (H 的堆顶元素 < a[i]) {
            删除堆顶元素； a[i]进堆；
        }
    }
    return 堆顶元素；                       //返回值（1 分）
}
```

六、附加题（10 分）

//判断顶点 v_i 和 v_j 之间是否有路径，有则返回 true，否则返回 false。

```
bool path_i_j(TypeOfVer vi, TypeOfVer vj) {
    int i, j, k, m;
    for (i=1; i<=Vers; i++) visited[i]=0;    //访问标记数组初始化
    for (i=0; i<Vers; ++i) if (verList[i].ver==vi) break;    //查找 vi 的编号
    if (i==Vers) {cout<<"顶点 vi 不存在\n"; return false;}
    for (j=0; j<Vers; ++j) if (verList[j].ver==vj) break;    //查找 vj 的编号
    if (j==Vers) {cout<<"顶点 vj 不存在\n"; return false;}
    int *s=new int[Vers], top=0;
    s[++top]=i;                                           //编号 i 入栈
    edgeNode *p;
    while (top>0) {k=s[top--];                           //出栈（2 分）
        p=verList[k].head;
        while (p!=NULL && visited[p->end]==1)    //（2 分）
            p=p->next;                            //查第 k 个链表中第一个未访问的边结点
        if (p==NULL) top--;
        else {m=p->end;
            if (m==j) return true;                //vi 和 vj 间有路径（1 分）
            else {visited[m]=1; s[++top]=m;}}
    }
    return false;                                         //顶点 vi 和 vj 间无通路（1·分）
}
```

初始化 4 分

主体 6 分