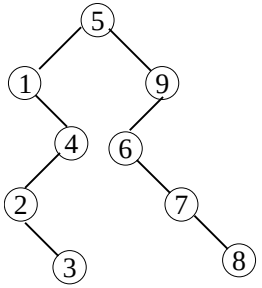
# 13-14 学年第 1 学期期末《数据结构》试题 A 参考答案

## 一、单项选择（20 × 2 = 40分）

CADCB　　DBBDB　　CCCAB　　DADCA

## 二、应用题（3 × 10 = 30分）

1、在二叉排序树中标出结点的值




2、如果顶点数组为{$V_1$, $V_2$, $V_3$, $V_4$, $V_5$}，则邻接矩阵为：

| | | | | |
|---|---|---|---|---|
| ∞ | 4 | 2 | ¥ | 8 |
| 4 | ¥ | ¥ | 4 | 5 |
| 2 | ¥ | ¥ | 1 | ¥ |
| ¥ | 4 | 1 | ¥ | 3 |
| 8 | 5 | ¥ | 3 | ¥ |

从 $V_1$ 点到各终点的距离值和最短路径的求解过程如下表：

| 顶点 | *i*=1 | *i*=2 | *i*=3 | *i*=4 | *i*=5 |
|---|---|---|---|---|---|
| $V_1$ | ∞ | ∞ | ∞ | ∞ | ∞<br>无 |
| $V_2$ | 4<br>{$V_1$, $V_2$} | 4<br>{$V_1$, $V_2$} | 4<br>{$V_1$, $V_2$} | | |
| $V_3$ | 2<br>{$V_1$, $V_3$} | | | | |
| $V_4$ | ∞ | 3<br>{$V_1$, $V_3$, $V_4$} | | | |
| $V_5$ | 8<br>{$V_1$, $V_5$} | 8<br>{$V_1$, $V_5$} | 6<br>{$V_1$, $V_3$, $V_4$, $V_5$} | 6<br>{$V_1$, $V_3$, $V_4$, $V_5$} | |
| $V_j$ | $V_3$ | $V_4$ | $V_2$ | $V_5$ | |
| S | {$V_1$, $V_3$} | {$V_1$, $V_3$, $V_4$} | {$V_1$, $V_2$, $V_3$, $V_4$} | {$V_1$, $V_2$, $V_3$, $V_4$, $V_5$} | |

$V_1$ 到 $V_2$ 最短路径长度为 4，路径为{$V_1$, $V_2$}；
$V_1$ 到 $V_3$ 最短路径长度为 2，路径为{$V_1$, $V_3$}；
$V_1$ 到 $V_4$ 最短路径长度为 3，路径为{$V_1$, $V_3$, $V_4$}；
$V_1$ 到 $V_5$ 最短路径长度为 6，路径为{$V_1$, $V_3$, $V_4$, $V_5$}。

3、• 哈希表的示意图：

| 地址下标 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 关键字 | | 8 | 16 | 15 | 32 | 24 | 30 | | | |
| 查找成功比较次数 | | 1 | 1 | 3 | 1 | 3 | 5 | | | |

，如果查找关键字 24，需要依次与 15, 32, 24 等 3 个元素比较

ƒ 查找成功时的平均查找长度 ASL = $\frac{1}{6}(1 + 1 + 3 + 1 + 3 + 5) = \frac{7}{3}$

## 三、算法设计（2 × 15 = 30分）

1、(1) 算法的基本设计思想：

顺次比较 *a*, *b* 两个数组的元素，将较小的一个存放到 *c* 数组中去，并取下一个元素；当一个数组搜索到尽头的时候，将另一个数组的剩余元素依次存放到 *c* 数组中

(2) 算法的详细实现步骤：

用 *i*, *j* 分别表示 *a*, *b* 数组的当前比较元素的下标，初值分别为 0；用 *k* 表示结果数组 *c* 的待用单元的下标，初值为 0

- 当 *i* < *m* 且 *j* < *n* 时，比较 *a*[*i*] 与 *b*[*j*]，比较小的元素存放的 *c* 数组中，且下标自加 1，*c* 数组下标自加 1，重复 •；

，当 *i* < *m* 时，将 *a* 数组的剩余元素依次存放到 *c* 数组中；

ƒ 当 *j* < *n* 时，将 *b* 数组的剩余元素依次存放到 *c* 数组中

(3) 算法的 C 语言描述：

```
void Merge(int a[ ], int b[ ], int c[ ], int m, int n)   //   函数首部
{
    int i, j, k;
    for (i = 0, j = 0, k = 0; i < m && j < n; ++ k)
    {   //   当a, b 中均有元素的时
        if (a[i] < b[j])     //   a[i]存放到c[k]中，取a 的下一个元素
            c[k] = a[i ++];
        else          //   b[j] 存放到c[k] 中，取b 的下一个元素
```

```
            c[k] = b[j ++];
        }
        while (i < m)           //  当b 中没有元素时
            c[k ++] = a[i ++];
        while (j < n)           //  当a 中没有元素时
            c[k ++] = b[j ++];
    }
```

2、 (1) 算法的基本设计思想：

用尾指针标识的带头结点的循环链表表示队列，初始化即生成一个头结点，头结点的指针域指向自身，形成循环链表。入队列，给新元素分配结点空间，并将新结点插到链表尾部，改写尾指针。出队列，判断队列是否为空，若不为空，则删除循环链表中第一个数据元素，若原队列只有一个元素，删除后要改写尾指针。

(2) 算法的详细实现步骤：

数据结构为单链表，一个数据域 *data*，一个指针域 *link*。

*a*. 初始化：生成一个新结点 *rear*，若分配成功，*rear* 的指针域指向自身。

*b*. 入队列：给新元素 *x* 分配结点空间 *s*，若分配成功，*s* 数据域赋值为 *x*，*s* 的指针域指向头结点，*s* 结点插入到链表尾部，*s* 为新的队尾。

*c*. 出队列：判断队列是否为空；若不为空，*h* 指向链表的头结点，删除 *h* 之后的结点；若 *h* 的指针域指向自身，即原队列只有一个元素，删除后要改写尾指针为 *h*。

(3) 算法的 C 语言描述：

```
#define OK 1
#define ERROR 0
typedef int Status;
typedef struct Node
{
    int data;
    struct Node *link;
} QNode,*QLink;
Status InitQueue(QLink &rear)
{   //  函数首部，初始化成功返回OK，否则返回ERROR
    rear = (QLink)malloc(sizeof(QNode));      //  分配头结点空间
    if (! rear)     //  动态存储分配失败，返回ERROR
        return ERROR;
    rear->link = rear;          //  循环链表
    return OK;
}
Status EnQueue(QLink &rear, int x)
{   //  函数首部，入队列成功返回OK，否则返回ERROR
    QLink s;
    s = (QLink)malloc(sizeof(QNode));     //  给x 分空间
    if (! s)                    //  动态存储分配失败，返回ERROR
        return ERROR;
    s->data = x;                //  新结点的数据域为x
    s->link = rear->link;       //  新结点的指针域指向头结点
    rear->link = s;             //  将新结点连到链表尾部
    rear = s;                   //  s 为新的链尾
    return OK;
}
Status DeQueue(QLink &rear)
{   //  函数首部，出队列成功返回OK，否则返回ERROR
    QLink h, p;
    if (rear->link == rear)         //  循环链表为空，删除失败
        return(ERROR);
    h = rear->link;
    p = h->link;
    h->link = p->link;          //  删除头结点后的结点
    free(p);
    if (h->link == h)
        rear = h;       //  原队列只有一个数据，删除后改写尾指针
    return OK;
}
```