云南省普通高校“专升本”招生考试

数据结构冲刺模拟试题（四）参考答案

一、单项选择题

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

11．C　12．A　13．B　14．D　15．B

二、填空题

16．O(n)　17．s->next=p->next；p->next=s　18．(1，3，2，4，5)

19．n-1　20．129　21．F= =R　22．p->lchild==0&&p->rchild==0

23．$O(n^2)$　24．$O(n\log_2 n)$；O(n)　25．开放定址法；链地址法

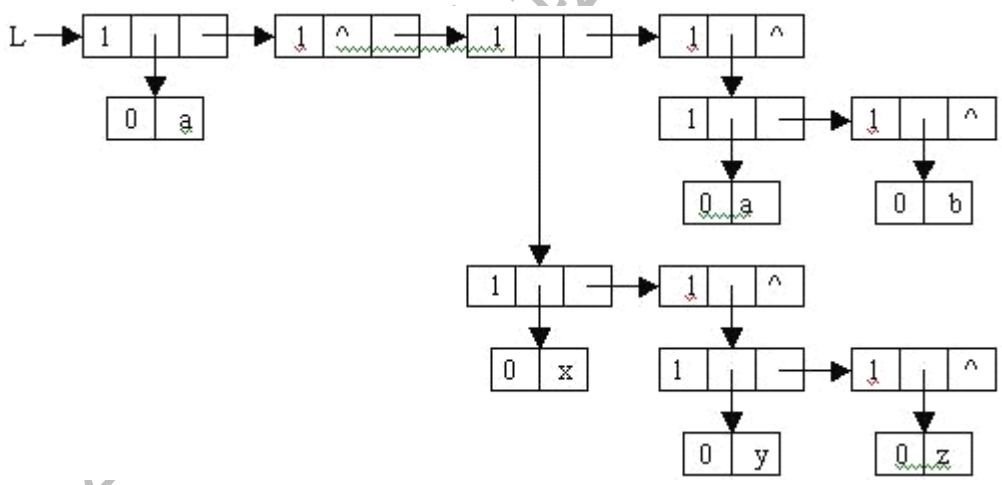
三、简答题

26．Huffman 算法：求 Huffman 树（带权路径长度最短的二叉树）；

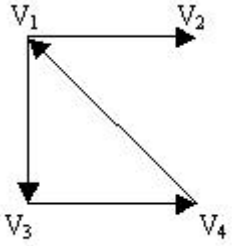
Dijkstra 算法：求图中从某个源点到其余各顶点的最短路径；

Prim 算法：求最小生成树；Kruskal 算法：求最小生成树

27．



28．举例：

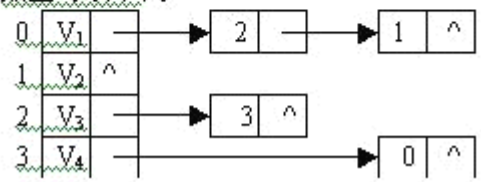


(1) 邻接矩阵（数组表示法）

如图可表示为 $\begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$

(2) 邻接表

如图可表示为



29．Hash 表如下

0	1	2	3	4	5	6	7	8	9	10
53	64	76	99	15	48	82	7		20	31
3	4	4	4	1	2	2	1		1	2

查找成功情况下的平均查找长度为(3+4+4+4+1+2+2+1+1+2)/10=2.4

四、算法阅读题

30．功能为：从初始点 v_i 出发广度优先搜索由邻接表 GL 所表示的图。

31．（1）查询链表的尾结点；（2）将第一个结点链接到链表的尾部，作为新的尾结点；

（3）返回的线性表为（$a_2,a_3,\cdots,a_n,a_1$）

五、算法计算题

32．typedef struct BiTNode

```
{
    TelemType data;
    struct BiTNode     *lchild, *rchild;
}BiTNode,*BiTree;
Status SearchBST(BiTree T,KeyType key,BiTree f,BiTree &p)
{
        if(!T){p=f;return FALSE;}
        else if EQ(key,T->data.key){p=T;return TRUE;}
        else if LT(key,T-> data.key) SearchBST(T->lchild,key,T,p);
        else if SearchBST(T->rchild,key,T,p);
}
Staus InsertBST(BiTree &T,ElemType e)
```

```
{
    if(!SearchBST(T,e.key,NULL,p)
    {
                s=(BiTree)malloc(sizeof(BiTNode));
                s->data=e;s->lchild=s->rchild=NULL;
                if(!p)    T=s;
                else if LT(e.key,p->data.key)p->lchild=s;
                else p->rchild=s;
                return TRUE;
    }
    else return FALSE;
}
```

从空树出发,经过一系列的查找插入操作之后,可生成一棵二叉树.

```
main()
{
        BiTree BiT=NULL;
        int elememt;
        scanf("%d",&element);
        while(element!=9999)
        {
                InsertBST(BiT,element);
                scanf("%d",&element);
        }
}
```

33．

```
typedef struct BiTNode
{
      TelemType data;
      struct BiTNode *left, *right;
      int bal;
}BiTNode,*BiTree;
int computbal(BiTree bt)
{
      int hl, hr, max;
      if(bt==NULL)    return(0);
      else {
            hl=computbal(bt->LChild);/*  求左子树的深度  */
            hr=computbal(bt->RChild);/*  求右子树的深度  */
            bt->bal=hl-hr;/*求出平衡因子*/
```

```
            max=hl>hr?hl:hr;/* 得到左、右子树深度较大者*/
            return(max+1);/* 返回树的深度 */
        }
}
```