一、

1．根；2：38，46，56，79，40，80；3：O(logn)　O（nlogn）

4.出度、入度　5：55；6：5；7：链式；8：层；9：散列函数、冲突处理

二、1C；2C；3B；4B；5B；6D；7B；8C；9A；10B

三、1 对；2 对；3 错；4 错；5 队；6 错；7 队；8，9，10 错；

四、2.

0-1：16　0，1　0-2：10　0，2　1 分

0-3：14 0，3

0-4：25 0，2，4　1 分

0-5：21 0，1，5　1 分

0-6：31 0，2，4，5　1 分

五．参考答案

存储结构：3 分

算法思想：2 分

程序：逻辑正确 5 分　其它：酌情扣分。

1．　算法思想：可以通过层序遍历的方法来解决。不管当前结点是否有左右孩子，都入队列。这样当树为完全二叉树时，遍历时得到是一个连续的不包含空指针的序列。反之，则序列中会含有空指针。

```
typedef    struct node {
                char data;
                node *lchild;
                node *rchild;
          {*btree;

 int    Juage(btree T)//判断二叉树是否完全二叉树，是则返回 1，否则返回 0
{    node *p;*q;
     makenull(Q); //清空队列
     flag=0; p=T;
     EnQueue(Q,p); //建立工作队列
   while(!Empty(Q))
     {
        q=DeQueue(Q);
        if(!q)
            flag=1;
        else if(flag)
```

```
            return 0;
        else
        {
          EnQueue(Q,q->lchild);
          EnQueue(Q,q->rchild); //不管孩子是否为空，都入队列
        }
    }//while
    return 1;
}//juage
```

2．算法思想：1）把边的权值按从大到小排序；

2）取最大权边，判断是否在环路中，在则删除。

3）重复 2）直到边数小于顶点数。

```
int C[n][n];
typedef struct
{ int beg, end; /*  边的起点与终点  */
   int weight; /*  边权  */
}Edge;
typedef Edge E[MAXEDGE];

int path( int C[n][n], int v, int w)
      { /*  从  w  顶点出发判断 w 到 vj 是否存在路*/
int i, visited[n], yes; STACK S;
MAKENULL(S); yes=0;
for (i=0; i<n; i++) visited[i]=0;
PUSH (w, S);
visited[w]=1; /*  访问  w  顶点  */
while (!EMPTY(S) && yes==0)
  {k=TOP(S);
   j=0;
   while((C[k][j]= =1&& visited[j]))
   j=j+1;/*找到没被访问的顶点*/
   if (j==n) POP(S);
   else    if (j==v) yes=1;
           else { visited[j]=1; PUSH (j, S);
            /*  对  v  的尚未访问的邻接顶点  w */
   }
   return yes;
```

```
   }

void spanning (int C[n][n], int e ,int n)
{ int    i ;
       Sort ( E, n );/*对权值按降幂排序*/
       i=1
       while(e>=n)
       {      C[E[i].beg][E[i].end]=0;
              C[E[i].end][E[i].beg]=0
              p=path();
              if (p==1) e=e-1;
              else    { C[E[i].beg][E[i].end]=1; C[E[i].end][E[i].beg]=1;}
              i=i+1;
       }
}
```