# $Homework2$

罗华坤 软件02 2019011799

## 1.

由于对于长度为 $n$ 的数组，插入位置共有 $n+1$ 种，插入会导致无序向量移动。

而每次插入导致移动的时间复杂度为 $O(n-r)$

即 $\exists c>0$，使得每次移动消耗 $c(n-r)$

因此时间复杂度应为

$$T(n)=\sum_{i=0}^{n}p_i\times move=\sum_{i=0}^{n}\frac{1}{n+1}\cdot c(n-i)=\frac{1}{n+1}\cdot\frac{c(n+1)(n+0)}{2}=\frac{cn}{2}=O(n)$$

## 2. D

## 3. A

## 4. D

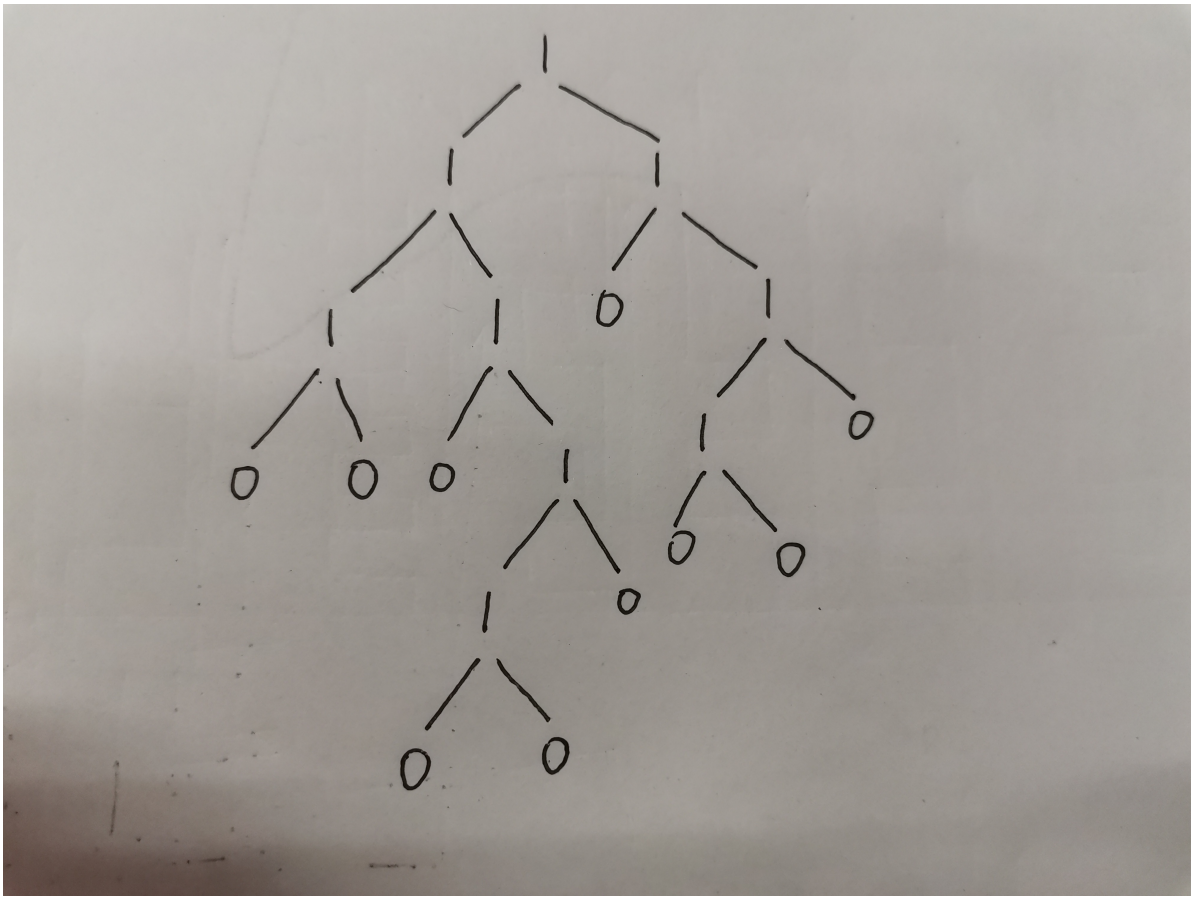
## 5.

$$parent[]=[-1,5,5,7,0,4,5,0,0,7]$$

## 6. 平均高度改变的节点个数为$O(x)$

## 7.

具体树图如下：



## 8.

### (1)

```
//construct the circular list
t -> next = new node(i,t->next);//创建节点
t = t -> next;

//game start
for(int i = 1; i < m; i++){//执行m-1次
    tempNode = x;
    x = x -> next;
}
tempNode -> next = x -> next;
delete x;
x = tempNode -> next;
```

### (2)

总体来说，$phi$ 能更快的抢到座位。但是，由于 $n, m$ 未给定，对于不同的情况我们仍需加以一定讨论。

分析两种做法，链表主要耗时在查找，而 $array$ 耗时在删除后的移位。

当 $n$ 比 $m$ 大得多 时，由于采用 $array$ 的方法需要将数组移位，而此时移位的长度接近 $array$ 长度，例如 $m = 1, n = 1000$ 时；而链表则能很快找到并删除，此时 $array$ 的表现更为不佳。

当 $m$ 较 $n$ 更大时，链表查找更为耗时，而 $array$ 移位的数量较少，因此该情况下 $array$ 更好。

当 $m$ 大小适中时，需要分析链表与 $array$ 的单次操作成本。由于链表的储存位置不连续，因此查找时缓存命中率更低，访问内存次数更多，因此耗时更长；而数组储存位置连续，缓存命中率更高，因此耗时更短。

且 $array$ 也可实现静态链表，由于上述性质执行同样较链表更快。

综上，使用 $array$ 往往能更快抢到座位。