# 算法导论

## 第二章 算法基础

### 2.2-1



### 2.2-2

选择排序中外层循环的范围为,因为若当i的上界为len时，最后一层循环的i可以取len-1, 内层循环的j则为len.因j不小于len，故不会进入到内层循环。所以i的范围为[0,len-1)时就已经足够遍历完整个数组中的元素。

for (int i = 0; i < len - 1; ++i) { c1 n-1

int min = i; c2 n-1

for (int j = i + 1; j < len; ++j) { c3 (n-2)\*(2/n)

if (arr[j] < arr[min]) c4 (n-2)\*(2/n-1)

min = j;

}

if (min != i) { c5 n-1

auto tmp = arr[i];

arr[i] = arr[min];

arr[min] = tmp;

}

}

最坏(好)复杂度为

Tn=c1\*(n-1)+c2\*(n-1)+c3\*（n-2）\*(2/n)+c4\*(n-2)\*(2/n-1)



### 2-2-3

平均情况以及最坏情况：

### 2-3-3

当n=2时，T(2)=2=2lg2=2

设n=时，T()=2T()+=lg=k成立

当n=时，T()=2T()+=2 k+=(k+1)

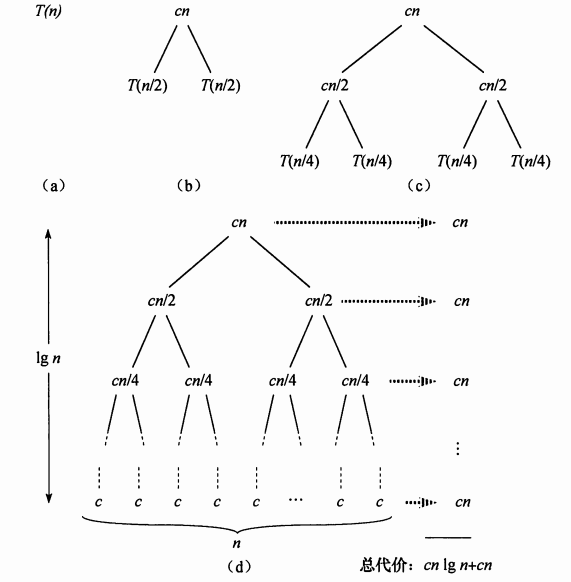
所以，当n刚好是2的幂时，

的解是T(n)=nlgn

注：算法导论中lgn表示的为

### 2-3-5

二分查找的最坏复杂度的计算：根据书上对归并排序的复杂度分析，如果用递归算法给出，可以清楚的看到其构成一个完全二叉树。完全二叉树的高度为或者，如下图。



### 2-3-6

不会。因为在插入排序中内循环(也就是2.1中INSERT-SORT的5~7行)仅仅是查找到了需要插入到的位置，即比较元素的次数。若改为二分查找，也仅仅是将查找次数的复杂度降低了，并不会影响到整个插入排序的复杂度。

### 思考题：

### 2-3

a.

b.

|  |
| --- |
| for i=0 to n-1{  yi = x;  for j = 1 to k  yi \*= x;  sum += a[i] \* yi; |
| } |