1. 求解递归方程 $T(n) = T(\frac{5n}{6}) + n$

由master定理:

$$a = 1,\ b = \frac{6}{5},\ f(n) = n,\ n^{log_b a} = \theta(n^2)$$

$$\because a = 1,\ log_b a = 0,\ f(n) = \Omega(n^{log_b a+1})$$

$$又\ \forall n > 0,\ f(\frac{5n}{6}) < f(n)$$

$$\therefore T(n) = \theta(f(n))$$

2. 证明或否证明 $f(n) + o(f(n)) = \theta(f(n))$

证明:

$$\because \forall a > 0,\ o(f(n)) < af(n) \Rightarrow f(n) + af(n) < (a+1)f(n)$$

$$又 \because o(f(n)) \geq 0$$

$$\therefore\ f(n) < f(n) + o(f(n)) < (a+1)f(n)$$

3. 证明 $\forall k \in \mathbb{Z}^+,\ log^k n = O(n)$

令命题 $P(k)$ 为 $log^k n = O(n)$

设 $f_k(x) = \frac{log^k x}{x},\ x \in \mathbb{R}^+$, 则 $P(k) \Leftrightarrow \lim_{x\to+\infty} f_k(x) = 0$

显然当 $k = 1$ 时 $P(k)$ 为真，下假设$P(k_0)$成立，即$\lim_{x\to+\infty} f_{k0}(x) = 0$，由洛必达法则:

$$\lim_{x\to+\infty} f_{k0+1}(x) = \lim_{x\to+\infty} \frac{(k_0+1)log^{k0}x}{x} = (k_0+1)\lim_{x\to+\infty}\frac{log^{k0}x}{x} = (k_0+1)\lim_{x\to+\infty} f_{k0}(x) = 0$$

故 $P(k_0 + 1)$ 也为真，综上可知原命题成立

4. 证明 $O(f(x)) + O(g(x)) = O(max(f(x), g(x)))$

题目等价于"$\forall r \in O(f(x)), s \in O(g(x)),\ \exists t \in O(max(f(x), g(x)))\ s.t.\ r(n) + s(n) = t(n)$"

$$r(x) \leq c_1 f(x),\ s(x) \leq c_2 g(x) \Rightarrow r(x) + s(x) \leq c_1 f(x) + c_2 g(x) \leq (c_1 + c_2)(max(f(x), g(x))) = O(max(f(x), g(x)))$$

因此原命题成立

5. 求解递归方程$T(n) = T(\lceil \frac{n}{2} \rceil) + 1$

$$T(n) = T(\lceil \frac{n}{2} \rceil) + 1 \leq T(\frac{n+1}{2}) + 1$$

假设$T(n) = O(logn)$

$$T(n) \leq clog(\frac{n+1}{2} - b) + 1 = clog(n+1-2b) - clog2 + 1 \leq clog(n+d) \quad (d = 1 - 2b)$$

6. 对于平面上的两个点 $p1 = (x1, y1)$ 和 $p2 = (x2, y2)$ ，如果 $x1 \leq x2$ 且 $y1 \leq y2$ ，则p2支配p1，给定平面上的n个点，请设计算法求其中没有被任何其他点支配的点。

用$a >> b$表示a支配b

由支配的定义知支配关系具有传递性，即$a >> b, b >> c \Rightarrow a >> c$，因此如果一个点被支配了，它可以不再参与接下来的任何比较

```
Input: 点集合 A
Output: 未被支配的点集合 B

B = []
for i in A
    for j in B
        if A[i] >> B[j]
            delete(B,j)
        else if B[j] >> A[i]
            continue i
    B += A[i]
return B
```

7.

如果一个数组A[1...n]中某个元素的数量超过其元素数量的一半，称其包含主元素，假设比较两个元素大小的时间不是常数但判定两个元素是否相等的时间是常数，要求对于给定数组A，设计算法判定其是否有主元素，如果有，找到该元素。 (1) 设计时间复杂性为O(nlogn)的算法完成该任务。 (2) 设计时间复杂性为O(n)的算法完成该任务。

分析: O(nlogn)算法考虑二分，由于不能比较大小，因此不能用排序的方法。但是根据主元素的定义，整个集合的主元素一定是它两个等分集合之一的主元素。因此可以采取二分的递归方法。 O(n)算法考虑桶排序或者哈希表，只需要记录元素的数量即可。

解:

```
Input: 集合 A
Output: 集合A中的主元素及其个数或null

O(nlogn) findMaster(A):
if A.length == 1
    return (A[0], 1)
(X,Y) = partition(A) // partition等分一个集合
(x,nx) = findMaster(X)
(y,ny) = findMaster(Y)
if x != null && nx + scan(Y,x) > A.length/2 //x为null意味着X没有主元素，此时A也一定没有主元素
    return (x, nx + scan(Y,x)) // scan在O(n)内查找一个集合中某元素的个数
if y != null && ny + scan(X,y) > A.length/2
    return (y, ny + scan(X,y))
return null // 程序运行到这里说明A没有主元素

O(n) findMaster(A):
B = new HashMap
for i in A
    if A[i] in B // O(1) for hashmap
        B[A[i]] += 1
    else
        B[A[i]] = 1
for i in B
    if B[i] > A.length / 2
        return (B, B.keys.length)
return null
```

8. 证明：在有 n 个数的序列中找出最大的数至少需要 $n-1$ 次比较

设 n 次比较最多能找出 $P(n)$ 个数中的最大值。下证 $P(n)=n+1$

显然，当 $n=1$ 时， $P(1)=2$ 满足上式。

假设当 $n=k$ 时 $P(k)=k+1$ ，那么对于 $k+1$ 次比较，前k次比较可以找出 $P(k)$ 个数中的最大值，而第 $k+1$ 次要增加新的信息，其操作数之一必须是 $P(k)$ 中的数，因此只能引入最多一个新的数，所以 $P(k+1)=P(k)+1=k+2$

由数学归纳法知原命题成立

9. 设计一个对 7 个元素进行排序的方法，保证其平均比较次数最少，要求证明这个结论

7个元素排序决策树高 $log_2(7!)=12.3$

1. 排序两个元素(1,2)，需要1次比较
2. 排序另两个元素(3,4)，需要1次比较
3. 排序剩下两个元素(5,6)，需要1次比较
4. 排序(2,4,6)，得到(1,2,4,6)，需要2到3次比较
5. 将5插入(1,2,4,6)，由于已知(5,6)，因此只需将5插入(1,2,4)，仅需2次比较
6. 将3插入(1,2,4,5,6)，由于已知(3,4)和(4,6)，因此只需将3插入(1,2,5)，仅需2次比较
7. 将7插入(1,2,3,4,5,6)，需要3次比较

至此排序完毕，最佳情况（排序2,4,6三个数只用2次比较）需要12次比较，最差情况（排序2,4,6三个数用了3次比较）需要13次比较，因此决策树是平衡二叉树，平均比较次数最少。

10.

a1, a2, …, an 是{1, 2, …, n}的一个随机排列，等可能第位 n!中可能排列中的任意一个，当对列表 a1, a2, ..., an 排序时，元素 ai 从它当前位置到达排序位置必须一定|ai-i|的距离，求元素必须移动的期望总距离 $E[\sum_{i=1}^{n}|a_i - i|]$

解: 设d(n)为需要移动的总距离

如果n在x的位置上，那么这(n-1)!个排列里y和n发生交换的次数有(n-2)!次，每次这样的交换带来(|n-x|+|n-y|-|x-y|)=(2n-x-y-|x-y|)次移动

$$\sum_{x=1}^{n-1}\sum_{y=1}^{n-1} 2n - x - y - |x-y| = 2n(n-1)^2 - \frac{n(n-1)^2}{2} * 2 - \frac{n(n-1)(n-2)}{3} = \frac{n(n-1)(2n-1)}{3}$$

$$d(n) = n * d(n-1) + \frac{n(n-1)(2n-1)}{3}$$

$$E(n) = (n-1)E(n-1) + \frac{(n-1)(2n-1)}{3}$$

E(1) = 0
E(2) = 1
E(3) = 16/3
E(4) = 23
E(5) = 104
通项公式$E(n) = -\frac{2n}{3} + \frac{5e\Gamma(n,1)}{3} - 1$