PS 3

1. (a) $T(n) \geq cn\lg n$ 审题：下界

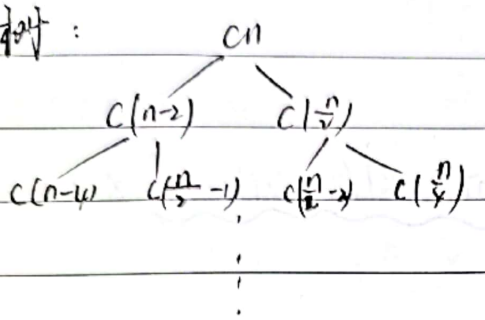
$T(n) = 2T(\frac{n}{2}) + n \geq 2c\cdot\lfloor\frac{n}{2}\rfloor \lg\lfloor\frac{n}{2}\rfloor + n$

$= cn\lg n + n(1-c)$

只需 $0<c<1$

边界条件 $T(2)$, $T(3)$ 成立

(b) 递推树：



cn

$\frac{3}{2}n-2$

$\frac{9}{4}n-7$

⋮

树高 $\frac{n}{2}$

上界：$(1+\frac{3}{2}+\cdots+(\frac{3}{2})^{\frac{n}{2}}) \in O((\frac{3}{2})^{\frac{n}{2}})$ → 无需再乘 n

代入法验证：$T(n) = c\cdot(\frac{3}{2})^{\frac{n}{2}}$

$T(n) = T(n-2) + T(\frac{n}{2}) + n = c\cdot(\frac{3}{2})^{\frac{n}{2}-1} + c\cdot(\frac{3}{2})^{\frac{n}{4}} + n$

$\leq c\cdot(\frac{3}{2})^{\frac{n}{2}}$

$T(1) = 1$ $T(2) = 3$

$T(1) \leq c\cdot\frac{3}{2}$, $T(2) = \frac{3}{2}c$

只需 $c \geq 2$ 即可

2. (a) 若 $T(a) \in O(1)$, 则 $T(n) \in \Omega(n)$

否则：$T(n) \in \Omega(n\cdot T(a))$

(b) $T(n) \in \Omega(n\cdot\lg n)$

3. 分治法.

将每个 n 位 2 进制数分成 2 个 $\lfloor \frac{n}{2} \rfloor$ 位，记为 x_L, x_R

则 $x^2 = x_L^2 \times 2^n + 2^{\frac{n}{2}+1} x_L x_R + x_R^2$

$= (x_L + x_R)^2 \times 2^{\frac{n}{2}} + x_L^2 \cdot (2^n - 2^{\frac{n}{2}}) + x_R^2 \cdot (1 - 2^{\frac{n}{2}})$

```
Square (x):
if (x is of 1 bit)
    return x * x
xL, xr = most, least significant |x|/2 bits of x
y1 = Square (xL)
y2 = Square (xR)
y3 = Square (xL+xR)
return y1*(2^n) + (y3 - y1 - y2) * (2^(n/2)) + y2
```

$T(n) = 3T(\frac{n}{2}) + \Theta(n)$

树高 $\lg n$. $n \cdot (\frac{3}{2})^{\lg n} = n \cdot n^{\lg \frac{3}{2}} = n^{\lg 3}$ ⇒ $\Theta(n^{\lg 3})$

4. 思路：不好认为 $x < n$

先比较 A[x] 与 x 大小关系

① 如果等于，则在 A[1~x] 二分查找

② 如小于，则比较 A[x] A[x+1] ... 与 x 依次

直至大于，再二分查找

```
BinarySearch (A, x, a, b)
left = a    right = b
while (left <= right)
    middle = (left + right)/2
    if (A[middle] = x)
        return middle
    else if (A[middle] < x)
        left = middle + 1
    else
        right = middle - 1
return
```

```
Mainsearch (A, x)
if A[x] = x
    return x.
else if (A[x] > x)
    return BinarySearch (A. x. 1, x)
else
    m = 2x.
    while [A[m] < x]
        m = 2m
    return Binary Search (A, x. m/2, m)
```

找出一名 majority 成员后，O(n) 内即完成

5. 二分法思想

循环如下：

若无人落单

将剩余的人两两配对，一定至少有一附组 majority 配对成功 ①

且由于 majority 人数过半，每有一组非 majority 配对，必有额外一组 majority 配对成功，则 majority 的成功组别必多于总成功组别的一半 ②

于是在所有成功组别中各选一人，进入下一次循环

特别说明：若有人落单时，可能会导致不成立，但此时那个落单的人必是 majority 成员，随后可以让此人与所有人配对，则 O(n) 时间即可完成筛选

循环结束说明：

① 剩1人，则必为 majority 成员

② 剩2人，则上一轮必有人落单，且为 majority 成员

Correctness：① 每次传入下一次循环的人中 majority 必超一半，否则上一轮必有落单且为 majority 中

② 循环可以结束，最后必剩1或2人

Time：

$\frac{n}{2} + \frac{n}{4} + \frac{n}{8} + \cdots \in O(n)$

6.

Divide：每次分成2个子数组 A[low, mid]、A[mid+1, high]

Conquer：每次改为求解子数组中4个最大片段并返回

① A[left, m] 为以包含左端侧尽头的

② A[n, right] 为以包含右侧尽头的

③ A[a, b] (left ≤ a ≤ b ≤ right)

④ A[left, right]

⇒ 保证连续性，易合

Combine：每次结合 A[left1, right1] 与 A[left2, right2] 时

(这里 right1 + 1 = left2)

① 比较 A[left1, m1]、A[left1, m2]

② 比较 A[n1, right1]，A[n1, right2]

③ 比较 A[a1, b1] A[n1, m2]，A[a2, b2]。

④ 比较 A[left1, right1]，A[left2, right2]，A[left1, right2]

每组取一个最大的

⇒ 返回新的4个片段 且O(1)即可

且分别对应 Conquer 中4个片段

满足 $T(n) = 2T(\frac{n}{2}) + O(1)$

7.

(a) 第二大的一定在第二行的二个中

FindSecond():

return max(data[2], data[3])

(b) 前 $\lfloor \lg k \rfloor$ 层共 $2^{\lfloor \lg k \rfloor} - 1 < k$ 个，所以 k^{th} 至少在 $\lfloor \lg k \rfloor + 1$ 层

第 k 层的元素一定有 k 至少小于 $k-1$ 个堆中元素，即 k^{th} 至多在第 k 层

*思想：只考虑堆的前 k+1 层，依次把前 k 层的 max 换到第 k+1 层，而控制原来 k+1 层的元素在后续操作中不仍在前 k 层

① HeapMax(i):

max = data[i]

data[i] = data[2^k - 1 + i]

Heapify(i)

return max

② Heapify(i)

a = 2×i , b = 2×i+1

imax = (data[a] > data[b] ? a : b)

imax = (data[i] > data[imax] ? i : imax)

if (imax ≠ i, and i ≤ $2^k - 1$)

data[imax] ↔ data[i]

Heapify(imax)

③ Main(k):

for i = 1 to k

kthmax = HeapMax(i)

return kthmax

Correctness: ① 每次 Main 都取了前 k 层最大的，也是所有层中未取进的中最大的，第 k 次 Main 取到了 k^{th}

② 被取进的 max 留在第 k+1 层，不会再被交换到前 k 层

running time:

Heapify : $O(\lg k)$

HeapMax : $O(1) + O(\lg k) = O(\lg k)$

Main() : $k\,O(\lg k) = O(k \lg k)$