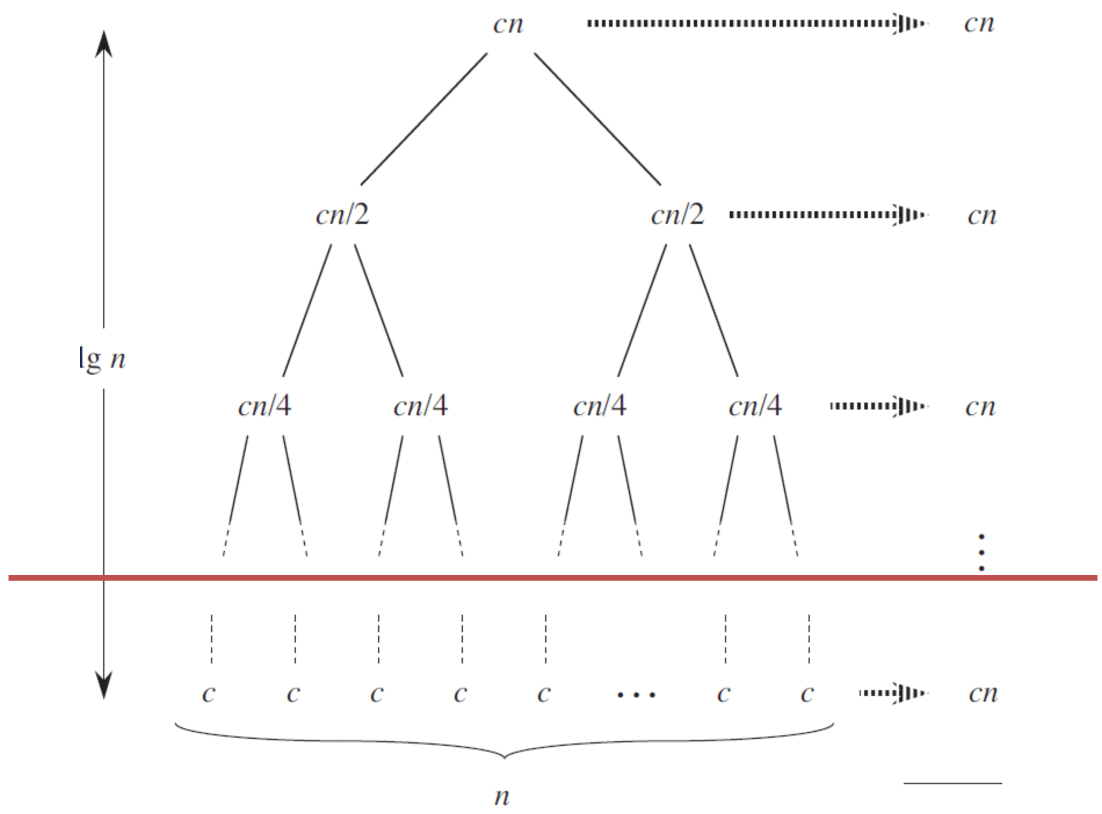
- 作业讲解
  - TC第2章问题1、2、3、4
  - TC第3章问题2、3、4

TC第2章问题1

- 这个算法的基本思路是什么？



TC第2章问题2

- a：对排序算法而言，partially correct的含义是什么？
  - A’[1]≤A’[2]≤…≤A’[n]
  - A’是A的一个permutation
- b：内层循环的loop invariant是什么？
  - $A[j]=\min_{j\le x\le n} A[x]$
  - A[j]…A[n]是原A[j]…A[n]的一个permutation
  - 不改变A[1]…A[i-1]
- c：外层循环的loop invariant是什么？
  - A[1]…A[i-1]是输入A[1]…A[n]的最小元素
  - A[1]≤A[2]≤…≤A[i-1]
  - A[1]…A[n]是输入A[1]…A[n]的一个permutation

TC第2章问题3

1. y=0
2. for i=n downto 0
3. $y=a_i+xy$

- c
  - 开始时，i=n
  - 结束时，i=-1

$$y = \sum_{k=0}^{n-(i+1)} a_{k+i+1} x^k$$

- d
  - Totally correct = partially correct + termination

TC第2章问题4c

```
INSERTION-SORT(A)
for j = 2 to A.length
    key = A[j]
    // Insert A[j] into the sorted sequence A[1..j-1].
    i = j - 1
    while i > 0 and A[i] > key
        A[i + 1] = A[i]
        i = i - 1
    A[i + 1] = key
```

- 算法运行时间
  - Ω(n)
  - O(n+逆序数)

TC第2章问题4d

- CNT(A, p, r) = CNT(A, p, q) + CNT(A, q+1, r) + CNT’(A, p, q, r)
  - CNT(A, p, r)：A[p..r]内的逆序对数
  - CNT’(A, p, q, r)：跨越A[p..q]和A[q+1..r]的逆序对数

-
```
i = 1
j = 1
for k = p to r
    if L[i] ≤ R[j]
        A[k] = L[i]
        i = i + 1
    else A[k] = R[j]
        j = j + 1
```
  - else时，发现n_1-(i-1)个逆序对

TC第3章问题2

	A	B	O	o	Ω	ω	Θ
a.	$\lg^k n$	n^ϵ	Yes	Yes			
b.	n^k	c^n	Yes	Yes			
c.	$\sqrt{n}$	$n^{\sin n}$					
d.	2^n	$2^{n/2}$			Yes	Yes	
e.	$n^{\lg c}$	$c^{\lg n}$	Yes		Yes		Yes
f.	$\lg(n!)$	$\lg(n^n)$	Yes		Yes		Yes

TC第3章问题3a

$2^{2^{n+1}}$

2^{2^n}

$(n+1)!$

$n!$

e^n

$n \cdot 2^n$

2^n

$(\frac{3}{2})^n$

$(\lg n)^{\lg n}$ $n^{\lg \lg n}$

$(\lg n)!$

n^3

n^2 $4^{\lg n}$

$n \lg n$ $\lg(n!)$

n $2^{\lg n}$

$(\sqrt{2})^{\lg n}$

$2^{\sqrt{2 \lg n}}$

$\lg^2 n$

$\ln n$

$\sqrt{\lg n}$

$\ln \ln n$

$2^{\lg^* n}$

$\lg^* n$ $\lg^*(\lg n)$

$\lg(\lg^* n)$

$n^{1/\lg n}$ 1

TC第3章问题3b

- $n^{\sin n}$是否满足要求？
- $2^{2^{n+2}\sin n}$是否满足要求？

TC第3章问题4

a. $f(n) = O(g(n))$ implies $g(n) = O(f(n))$. f(n)=n, g(n)=n^2

b. $f(n) + g(n) = \Theta(\min(f(n), g(n)))$. f(n)=n, g(n)=n^2

c. $f(n) = O(g(n))$ implies $\lg(f(n)) = O(\lg(g(n)))$, where $\lg(g(n)) \geq 1$ and $f(n) \geq 1$ for all sufficiently large n.

d. $f(n) = O(g(n))$ implies $2^{f(n)} = O\left(2^{g(n)}\right)$. f(n)=2^{n+1}, g(n)=2^n

e. $f(n) = O\left((f(n))^2\right)$. f(n)=n^{-1}

f. $f(n) = O(g(n))$ implies $g(n) = \Omega(f(n))$.

g. $f(n) = \Theta(f(n/2))$. f(n)=2^n

h. $f(n) + o(f(n)) = \Theta(f(n))$.

- 教材讨论
  - TC第4章

问题1：maximum-subarray problem

```
FIND-MAXIMUM-SUBARRAY(A, low, high)
if high == low
    return (low, high, A[low])          // base case: only one element
else mid = ⌊(low + high)/2⌋
    (left-low, left-high, left-sum) =
        FIND-MAXIMUM-SUBARRAY(A, low, mid)
    (right-low, right-high, right-sum) =
        FIND-MAXIMUM-SUBARRAY(A, mid + 1, high)
    (cross-low, cross-high, cross-sum) =
        FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)
    if left-sum ≥ right-sum and left-sum ≥ cross-sum
        return (left-low, left-high, left-sum)
    elseif right-sum ≥ left-sum and right-sum ≥ cross-sum
        return (right-low, right-high, right-sum)
    else return (cross-low, cross-high, cross-sum)
```

```
FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)
left-sum = −∞
sum = 0
for i = mid downto low
    sum = sum + A[i]
    if sum > left-sum
        left-sum = sum
        max-left = i
right-sum = −∞
sum = 0
for j = mid + 1 to high
    sum = sum + A[j]
    if sum > right-sum
        right-sum = sum
        max-right = j
return (max-left, max-right, left-sum + right-sum)
```

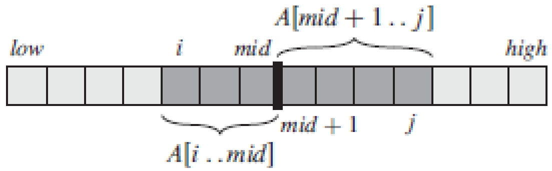
- divide、conquer、combine在这个算法中分别如何体现？
- 为什么这个divide-and-conquer比brute-force快？
  节约了哪些计算？
- 运行时间的递归式是什么？

问题1：maximum-subarray problem

```
FIND-MAXIMUM-SUBARRAY(A, low, high)
if high == low
    return (low, high, A[low])            // base case: only one element
else mid = ⌊(low + high)/2⌋
    (left-low, left-high, left-sum) =
        FIND-MAXIMUM-SUBARRAY(A, low, mid)
    (right-low, right-high, right-sum) =
        FIND-MAXIMUM-SUBARRAY(A, mid + 1, high)
    (cross-low, cross-high, cross-sum) =
        FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)
    if left-sum ≥ right-sum and left-sum ≥ cross-sum
        return (left-low, left-high, left-sum)
    elseif right-sum ≥ left-sum and right-sum ≥ cross-sum
        return (right-low, right-high, right-sum)
    else return (cross-low, cross-high, cross-sum)
```

```
FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)
left-sum = -∞
sum = 0
for i = mid downto low
    sum = sum + A[i]
    if sum > left-sum
        left-sum = sum
        max-left = i
right-sum = -∞
sum = 0
for j = mid + 1 to high
    sum = sum + A[j]
    if sum > right-sum
        right-sum = sum
        max-right = j
return (max-left, max-right, left-sum + right-sum)
```

- divide、conquer、combine在这个算法中分别如何体现？
- 为什么这个divide-and-conquer比brute-force快？
  节约了哪些计算？
- 运行时间的递归式是什么？



问题1： maximum-subarray problem (续)

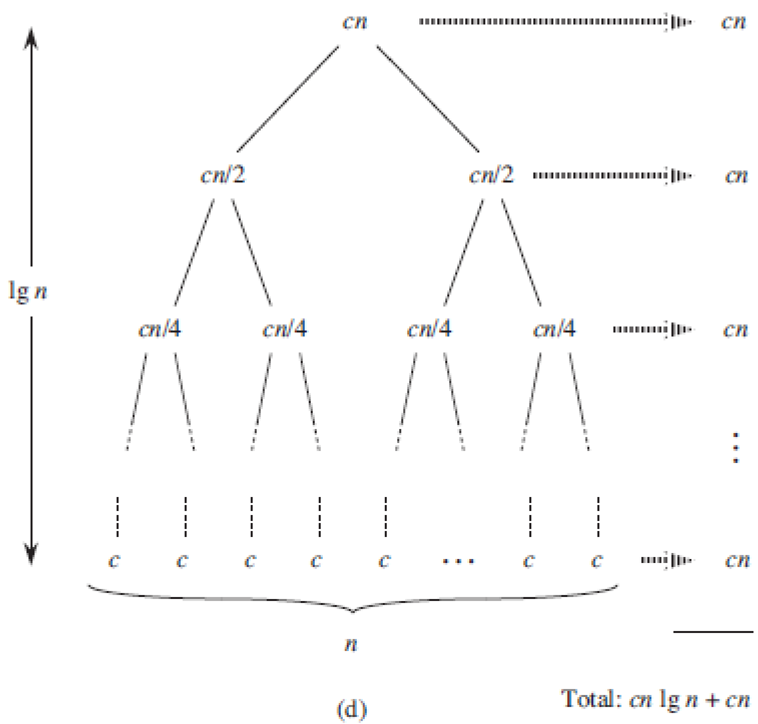
$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1\,, \\ 2T(n/2) + \Theta(n) & \text{if } n > 1\,. \end{cases}$$

- 你能画出递归树，并利用递归树来猜测递归式的解吗？

问题1：maximum-subarray problem (续)

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 , \\ 2T(n/2) + \Theta(n) & \text{if } n > 1 . \end{cases}$$

- 你能画出递归树，并利用递归树来猜测递归式的解吗？



问题1：maximum-subarray problem (续)

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1\,, \\ 2T(n/2) + \Theta(n) & \text{if } n > 1\,. \end{cases}$$

- 这段基于数学归纳法的证明，你能解释其中的红色标注吗？
  - 目标：$\exists c > 0, T(n) \le cn\lg n$
  - 初始：
    - $T(1) = \Theta(1) \le c1\lg 1$ Oops!
    - $T(2) = 2\Theta(1) + \Theta(2) \le c2\lg 2$
    - $T(3) = 2\Theta(1) + \Theta(3) \le c3\lg 3$
  - 递推：
    - 假设：$T\left(\frac{n}{2}\right) \le c\frac{n}{2}\lg\frac{n}{2}$
    - 推导：$T(n) \le 2c\frac{n}{2}\lg\frac{n}{2} + \Theta(n) = cn\lg\frac{n}{2} + \Theta(n) = cn\lg n - cn\lg 2 + \Theta(n)$

      $\le cn\lg n - cn + dn = cn\lg n - (c-d)n \le cn\lg n$

      d是什么？　　最后一步的理由？

问题1：maximum-subarray problem (续)

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \, , \\ 2T(n/2) + \Theta(n) & \text{if } n > 1 \, . \end{cases}$$

- 主定理的3种case能覆盖所有情形吗？
- 你能利用主定理来解这个递归式吗？

Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

$$T(n) = aT(n/b) + f(n) \, ,$$

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. ■

问题2：substitution method

- $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$

- 尝试 $T(n) \le cn$

$$\begin{aligned} T(n) &\le c\lfloor n/2 \rfloor + c\lceil n/2 \rceil + 1 \\ &= cn + 1 , \end{aligned}$$

- 尝试 $T(n) \le cn - d$

$$\begin{aligned} T(n) &\le (c\lfloor n/2 \rfloor - d) + (c\lceil n/2 \rceil - d) + 1 \\ &= cn - 2d + 1 \\ &\le cn - d , \end{aligned}$$

- 教材希望通过这个例子教我们什么？你理解这段证明了吗？

问题2：substitution method (续)

- $T(n) = 2T\left(\lfloor\sqrt{n}\rfloor\right) + \lg n$

- $m = \lg n \quad \Rightarrow \quad T(2^m) = 2T(2^{m/2}) + m$
- $S(m) = T(2^m) \quad \Rightarrow \quad S(m) = 2S(m/2) + m$

  $\Rightarrow \quad S(m) = O(m \lg m)$

  $\Rightarrow \quad T(n) = T(2^m) = S(m) = O(m \lg m) = O(\lg n \lg \lg n)$

- 教材希望通过这个例子教我们什么？你理解这段证明了吗？

问题3：recursion-tree method

Argue that the solution to the recurrence $T(n) = T(n/3) + T(2n/3) + cn$, where c is a constant, is $\Omega(n \lg n)$ by appealing to a recursion tree.

问题4： master method

- 你能用主定理解这些递归式吗？

a. $T(n) = 2T(n/4) + 1.$

b. $T(n) = 2T(n/4) + \sqrt{n}.$

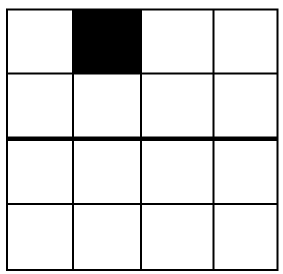
c. $T(n) = 2T(n/4) + n.$

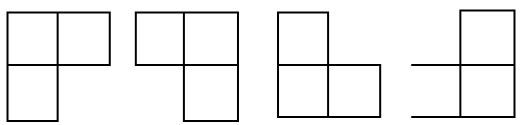
d. $T(n) = 2T(n/4) + n^2.$

e. $T(n) = 2T(n/4) + \sqrt{n}\lg n$

问题5：divide-and-conquer

- 在一个2^k*2^k的棋盘中，有某个格子已被覆盖了，你能否设计一个分治算法，使用一些L型骨牌恰覆盖棋盘上的其它所有格子？
- 你能分析你给出的这个算法的运行时间吗？





问题5：divide-and-conquer

- 在一个2^k*2^k的棋盘中，有某个格子已被覆盖了，你能否设计一个分治算法，使用一些L型骨牌恰覆盖棋盘上的其它所有格子？
- 你能分析你给出的这个算法的运行时间吗？