

计算机问题求解 – 论题2-1

- 算法问题与解题的算法

课程研讨

- TC第1、2、3章

问题1： 计算问题与算法

- 什么是a well-specified computational problem?
- 问题和问题的实例有什么区别?
- 什么是an algorithm?

- 你能举一组例子吗?

问题1：计算问题与算法 (续)

- a well-specified computational problem
 - input + output + their relationship
- an algorithm
 - well-defined computational procedure for achieving an input-output relationship

问题1：计算问题与算法 (续)

- 回忆一下，你设计过哪些糟糕的算法？糟糕在什么地方？
- 你觉得什么是一个“好”算法？

问题1：计算问题与算法 (续)

- 回忆一下，你设计过哪些糟糕的算法？糟糕在什么地方？
- 你觉得什么是一个“好”算法？
 - 正确的：总能停止且结果正确
 - 高效的：运行速度快、占用空间少
 - 易实现的：描述清晰、简单

问题1：计算问题与算法 (续)

- 你能不能总结一下，算法设计的一般步骤是什么？

问题1：计算问题与算法 (续)

- 你能不能总结一下，算法设计的一般步骤是什么？



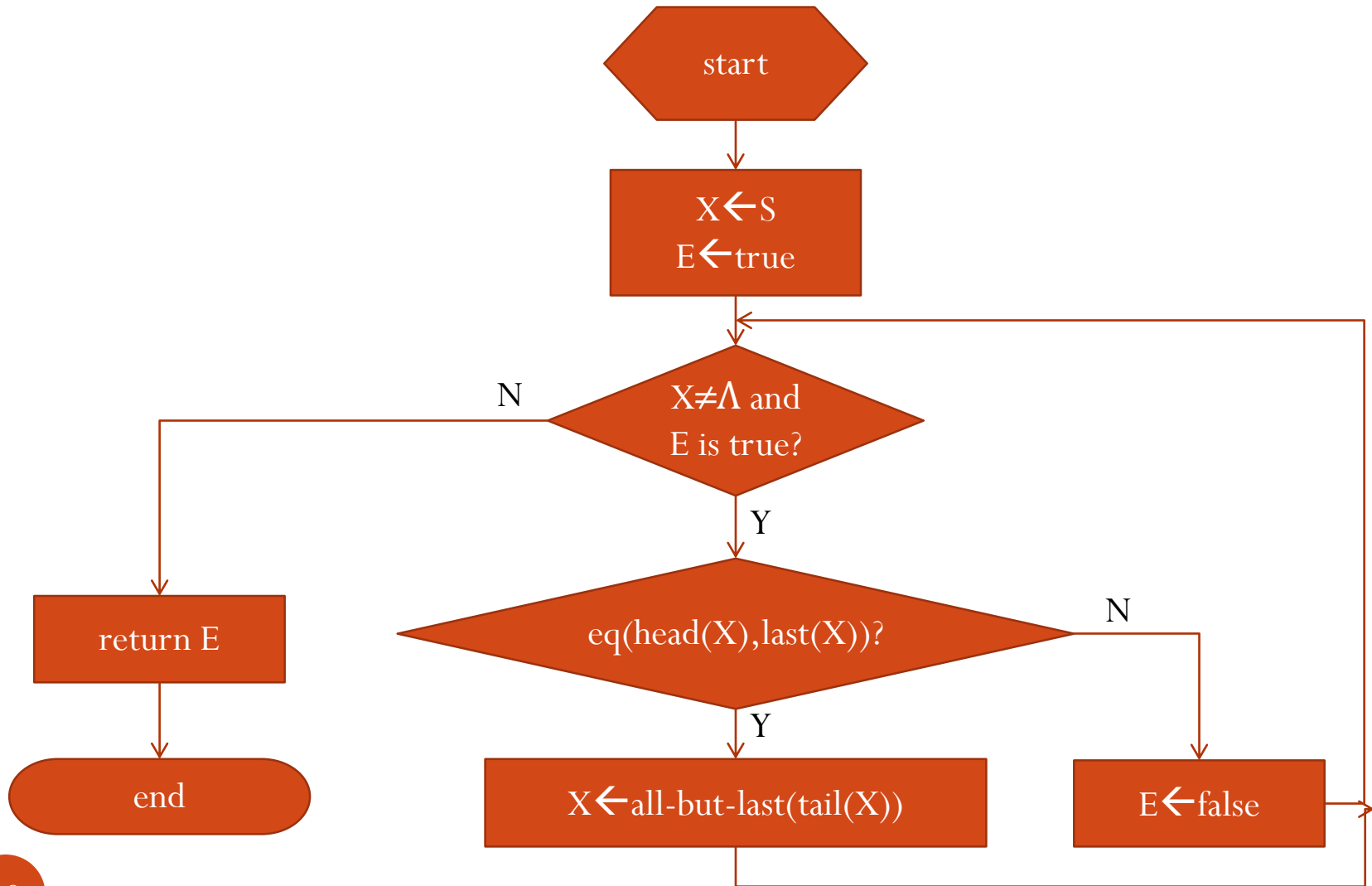
问题2：算法的正确性分析

- 你还记得如何证明算法的正确性吗？
 - 什么是partially correct? 如何证明?
 - 什么是totally correct? 如何证明?

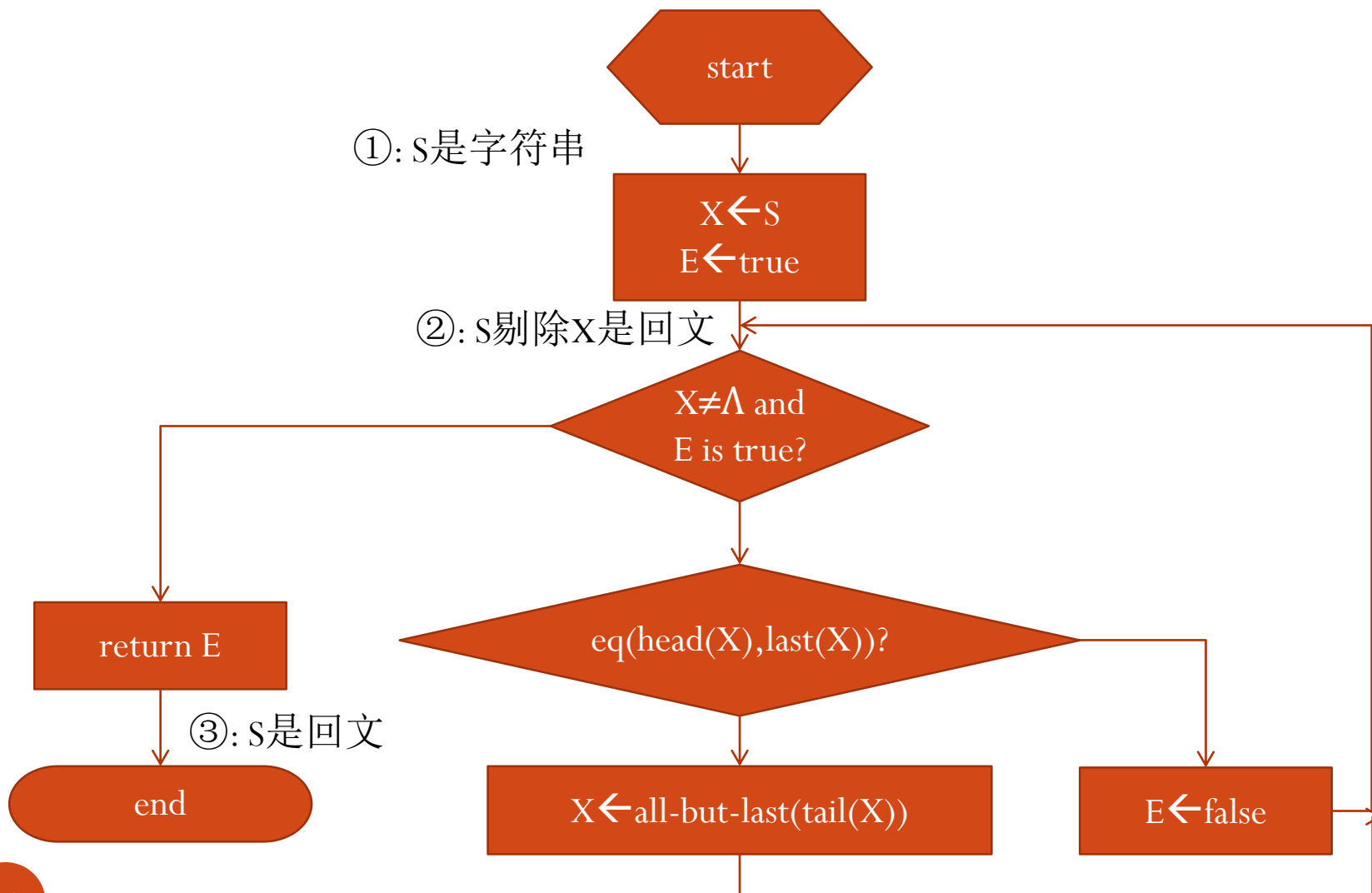
问题2：算法的正确性分析 (续)

- 如何证明算法是partially correct?
 1. 设置checkpoint
 - start后和end前各一个
 - 每个回路上至少一个（通常是第一次进入回路时）
 2. 为每个checkpoint设置invariant（最后一个invariant是算法期望的结果）
 3. 检查所有checkpoint之间的路径，说明为什么路径起点的invariant成立时，路径终点的invariant也成立
- 如何证明算法是totally correct?
 - partially correct + termination

举例：回文检测算法



举例：回文检测算法



问题3：算法的效率分析

- 分析算法的效率时，为什么要先定义计算模型？
- Random-Access Machine有哪些要素？

问题3：算法的效率分析

- 分析算法的效率时，为什么要先定义计算模型？
- Random-Access Machine有哪些要素？
 - 数据
 - 支持的类型
 - 存储的方式
 - 指令
 - 支持的类型
 - 执行的方式

举例：回文检测算法

PALINDROME-TEST (S)

1. $X=S$
2. $E=true$
3. while $X \neq \Lambda$ and $E=true$
4. if $eq(head(X),last(X))=true$
5. $X=all-but-last(tail(X))$
6. else
7. $E=false$
8. return E

- input size是什么？
- 如何计算它的running time？
- best/worst/average case分别是多少？

举例：回文检测算法

PALINDROME-TEST (S)

	cost	times	
1. X=S	c_1	1	
2. E=true	c_2	1	• input size是什么?
3. while X≠Λ and E=true	c_3	p	• 如何计算它的running time?
4. if eq(head(X),last(X))=true	c_4	p-1	• best/worst/average case分别是多少?
5. X=all-but-last(tail(X))	c_5	q	
6. else	c_6	p-1-q	
7. E=false	c_7	p-1-q	
8. return E	c_8	1	

$$\text{running time} = c_1 + c_2 + c_3 \cdot p + c_4 \cdot (p-1) + c_5 \cdot q + c_6 \cdot (p-1-q) + c_7 \cdot (p-1-q) + c_8$$

举例：回文检测算法

PALINDROME-TEST (S)	cost	times
1. X=S	c_1	1
2. E=true	c_2	1
3. while X \neq Λ and E=true	c_3	p
4. if eq(head(X),last(X))=true	c_4	p-1
5. X=all-but-last(tail(X))	c_5	q
6. else	c_6	p-1-q
7. E=false	c_7	p-1-q
8. return E	c_8	1

- input size是什么?
- 如何计算它的running time?
- best/worst/average case分别是多少?

$$\text{running time} = c_1 + c_2 + c_3 \cdot p + c_4 \cdot (p-1) + c_5 \cdot q + c_6 \cdot (p-1-q) + c_7 \cdot (p-1-q) + c_8$$

$$\text{best case: } c_1 + c_2 + c_3 \cdot 2 + c_4 \cdot 1 + c_5 \cdot 0 + c_6 \cdot 1 + c_7 \cdot 1 + c_8$$

$$\text{worst case: } c_1 + c_2 + c_3 \cdot \left(\left\lceil \frac{n}{2} \right\rceil + 1 \right) + c_4 \cdot \left\lceil \frac{n}{2} \right\rceil + c_5 \cdot \left\lceil \frac{n}{2} \right\rceil + c_6 \cdot 0 + c_7 \cdot 0 + c_8$$

average case?

问题3：算法的效率分析 (续)

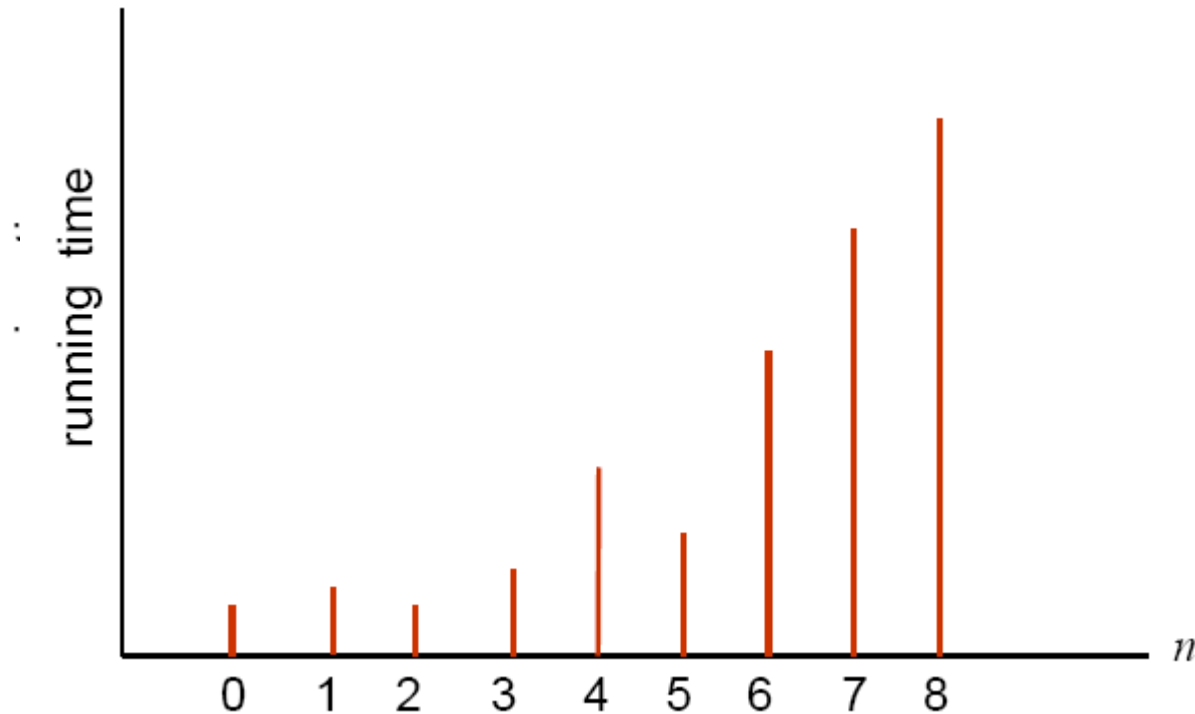
- 为什么我们通常最关注worst case?

问题3：算法的效率分析 (续)

- 为什么我们通常最关注worst case?
 - Gives us an upper bound on the running time for any input.
 - Occurs fairly often.
 - The “average case” is often roughly as bad as the worst case.

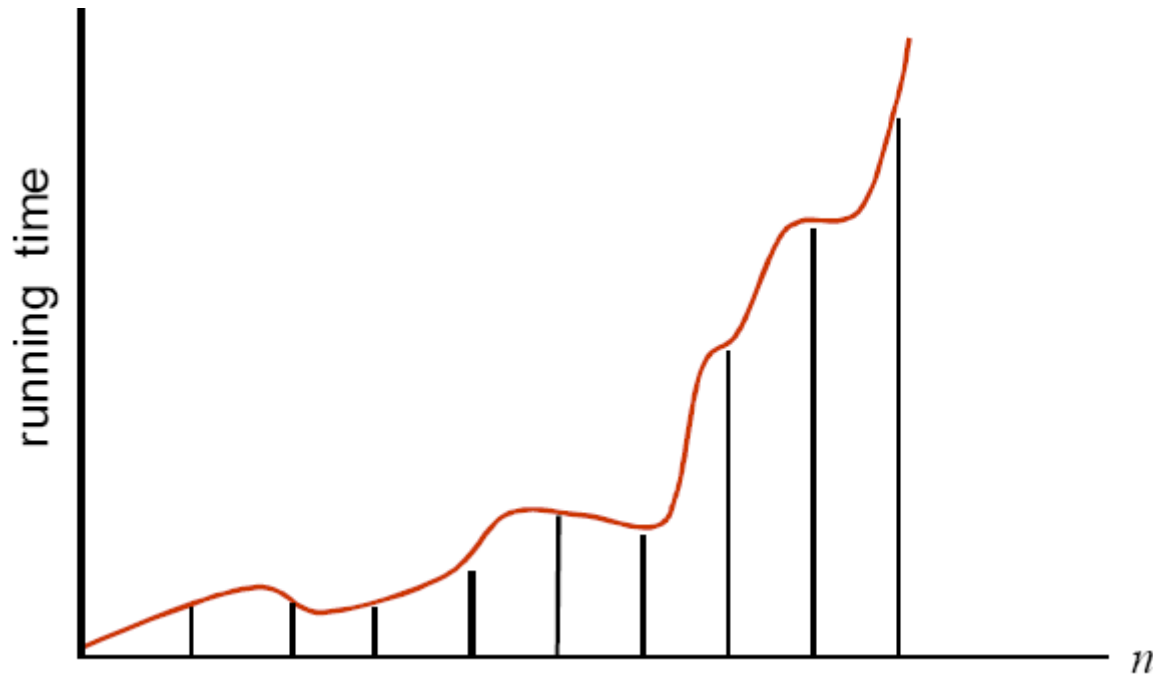
Worst Case Measure

- For each input size we select the worst case.



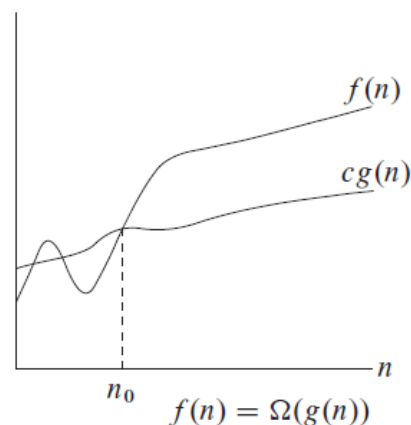
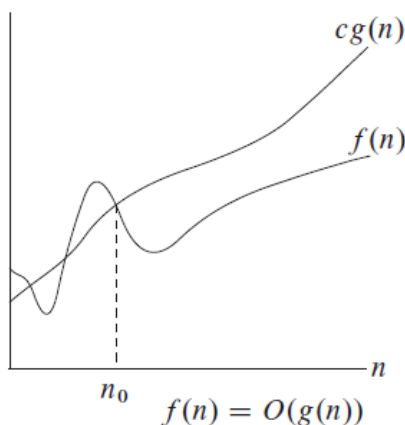
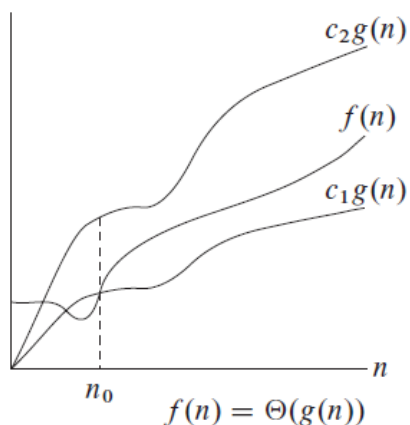
Running Time not Monotonic

- Actual running time may not be monotonic in n .



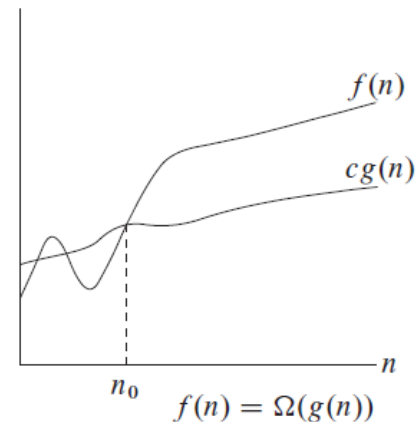
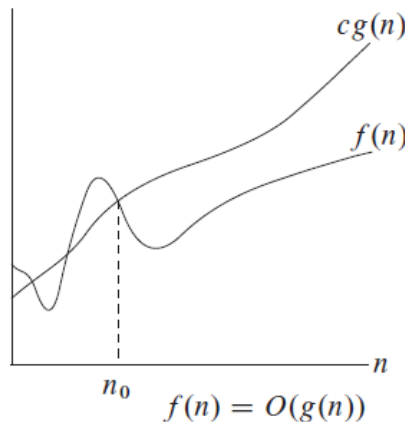
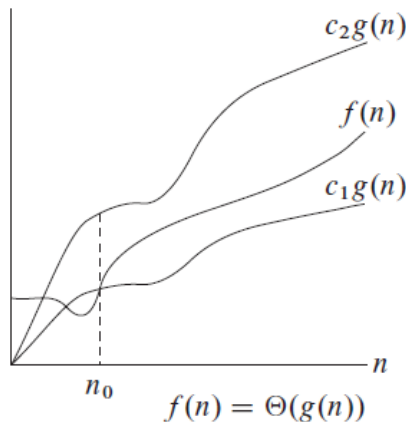
问题4：算法效率的渐进表示法

- Θ , O 和 Ω 的数学本质是什么？
 - 例如， $\Theta(n^2)$ 在数学上是一个什么？
- 你能严格说出 $\Theta(g(n))$, $O(g(n))$ 和 $\Omega(g(n))$ 的数学定义吗？



问题4：算法效率的渐进表示法

- Θ , O 和 Ω 的数学本质是什么？
 - 例如， $\Theta(n^2)$ 在数学上是一个什么？
- 你能严格说出 $\Theta(g(n))$, $O(g(n))$ 和 $\Omega(g(n))$ 的数学定义吗？



$\Theta(g(n)) = \{f(n) : \text{there exist positive constants } c_1, c_2, \text{ and } n_0 \text{ such that } 0 \leq c_1g(n) \leq f(n) \leq c_2g(n) \text{ for all } n \geq n_0\} .^1$

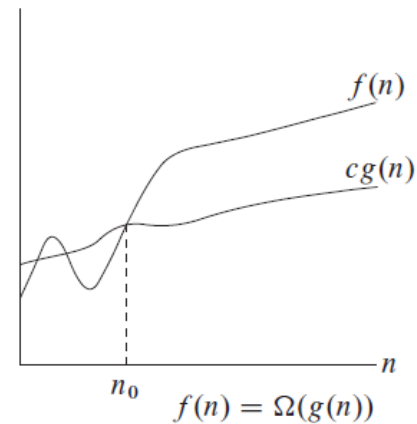
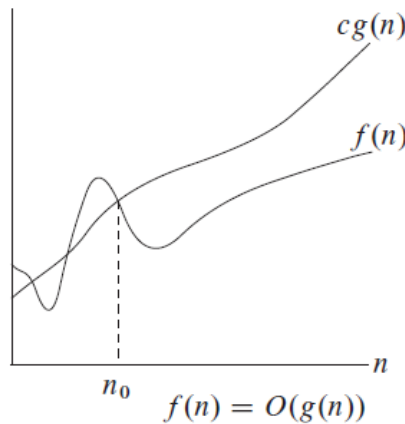
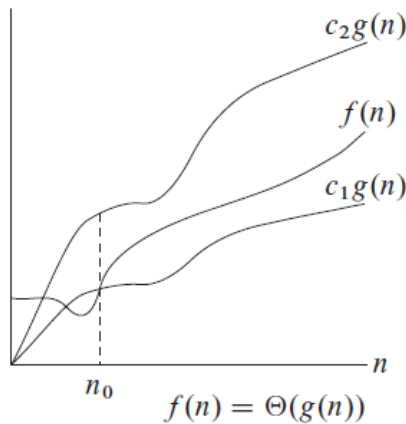
$O(g(n)) = \{f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0\} .$

$\Omega(g(n)) = \{f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \leq cg(n) \leq f(n) \text{ for all } n \geq n_0\} .$

有没有觉得似曾相识？
你能改写成更简洁的形式吗？

问题4：算法效率的渐进表示法

- Θ , O 和 Ω 的数学本质是什么？
 - 例如， $\Theta(n^2)$ 在数学上是一个什么？
- 你能严格说出 $\Theta(g(n))$, $O(g(n))$ 和 $\Omega(g(n))$ 的数学定义吗？



$\Theta(g(n)) = \{f(n) : \text{there exist positive constants } c_1, c_2, \text{ and } n_0 \text{ such that } 0 \leq c_1g(n) \leq f(n) \leq c_2g(n) \text{ for all } n \geq n_0\} .^1$

$O(g(n)) = \{f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0\} .$

$\Omega(g(n)) = \{f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \leq cg(n) \leq f(n) \text{ for all } n \geq n_0\} .$

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = c \in (0, \infty)$$



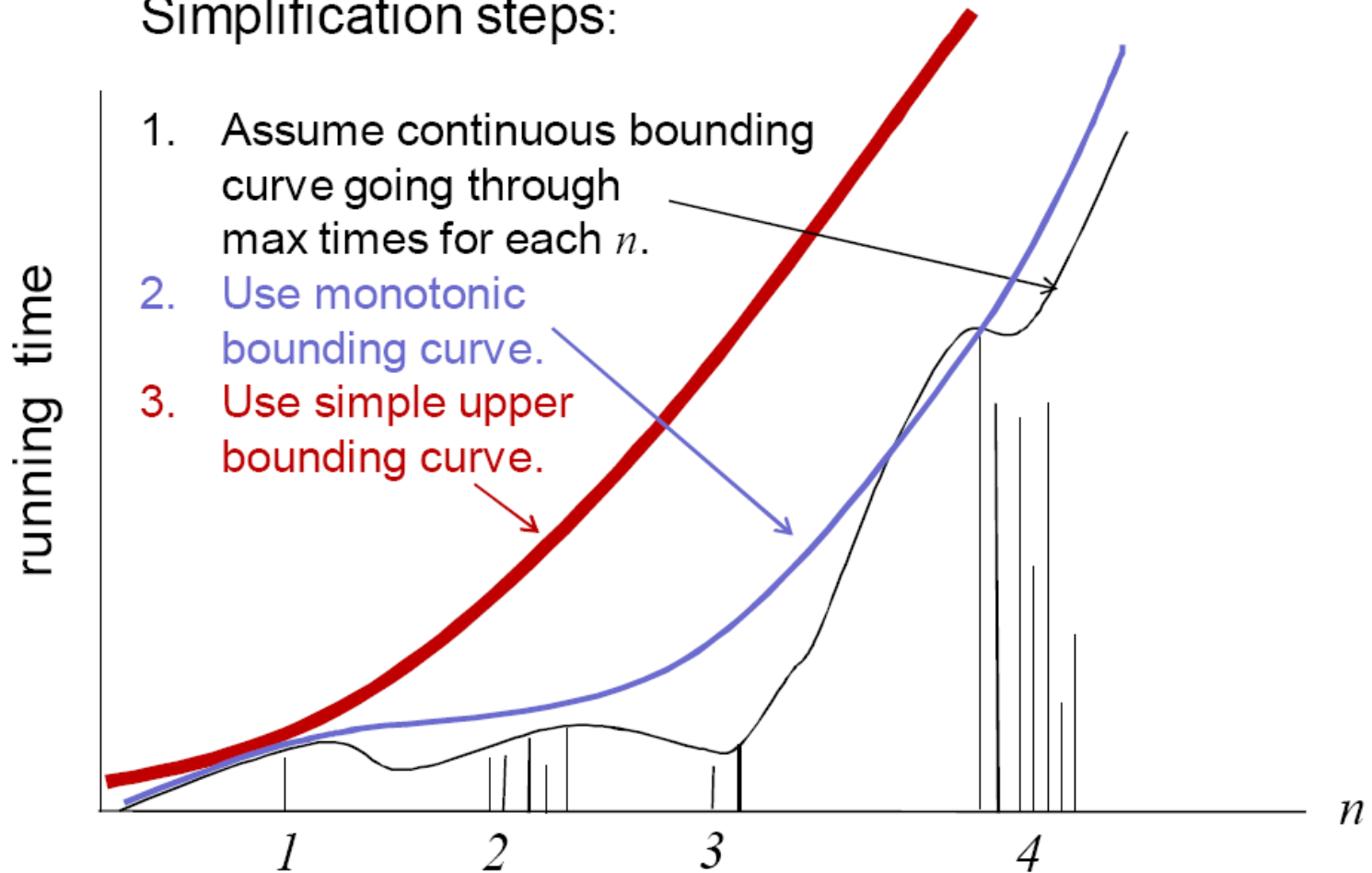
$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} < \infty$$

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} > 0$$

To use a “simple” Function

Simplification steps:

1. Assume continuous bounding curve going through max times for each n .
2. Use monotonic bounding curve.
3. Use simple upper bounding curve.



问题4：算法效率的渐进表示法 (续)

- 你能用 Θ 来表示它们的running time吗？

best case: $c_1 + c_2 + c_3 \cdot 2 + c_4 \cdot 1 + c_5 \cdot 0 + c_6 \cdot 1 + c_7 \cdot 1 + c_8$

worst case: $c_1 + c_2 + c_3 \cdot \left(\left\lceil \frac{n}{2} \right\rceil + 1 \right) + c_4 \cdot \left\lceil \frac{n}{2} \right\rceil + c_5 \cdot \left\lceil \frac{n}{2} \right\rceil + c_6 \cdot 0 + c_7 \cdot 0 + c_8$

- 你能不能证明 $\Theta(n^2) = \Theta(an^2 + bn + c)$ ？
- 尽管严格来说是 $bn + c \in \Theta(n)$ ，但有时候也写成 $bn + c = \Theta(n)$ ，因此 $an^2 + bn + c = an^2 + \Theta(n)$ ，这种写法有什么好处？
- 如果改用 O 而不是 Θ ，优缺点分别是什么？

问题4：算法效率的渐进表示法 (续)

- O 和 o 在数学上的区别是什么？

- 你是如何理解这个比喻的？

$f(n) = O(g(n))$ is like $a \leq b$,

$f(n) = \Omega(g(n))$ is like $a \geq b$,

$f(n) = \Theta(g(n))$ is like $a = b$,

$f(n) = o(g(n))$ is like $a < b$,

$f(n) = \omega(g(n))$ is like $a > b$.

问题4：算法效率的渐进表示法 (续)

- O 和 o 在数学上的区别是什么？

$O(g(n)) = \{f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0\}.$

$o(g(n)) = \{f(n) : \text{for any positive constant } c > 0, \text{ there exists a constant } n_0 > 0 \text{ such that } 0 \leq f(n) < cg(n) \text{ for all } n \geq n_0\}.$



$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} < \infty$$

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$$

- 你是如何理解这个比喻的？

$f(n) = O(g(n))$ is like $a \leq b$,

$f(n) = \Omega(g(n))$ is like $a \geq b$,

$f(n) = \Theta(g(n))$ is like $a = b$,

$f(n) = o(g(n))$ is like $a < b$,

$f(n) = \omega(g(n))$ is like $a > b$.