Chapter 4

Algorithm Analysis and Asymptotic Notation

Bahar Aameri
Department of Computer Science
University of Toronto

Mar 20, 2015
Announcements

Final Exam:
- Chapter 1: Only Section 1.5 (Problem-solving techniques in Section 1.3 are useful in doing proofs. See Tutorial 5 and its solutions)
- Chapter 2
- Chapter 3
- Chapter 4: Excluding Sections 4.2 and 4.3
Today’s Topics

- Analyzing Running Time of Algorithms
- Running Time Analysis for Linear Search
- Running Time Analysis for Insertion Sort
Worst-case analysis of algorithm

Given a program $P$, calculate a coarse estimation of its worst-case running time based on the size of the input.

Symbolic Translation

- $P$: A program.
- $t_P(x)$: running time of program $P$ with input $x$.
- $I$: the set of all inputs for $P$.
- $T_P(n)$: worst-case running time of $P$ for inputs with size $n$.

$$T_P(n) = \max\{t_P(x) \mid x \in I \land \text{size}(x) = n\}$$
Review (from Week 8)

Worst-case analysis of algorithm

- It is infeasible to calculate the exact value of $T_P(n)$.
- We only need a coarse estimation
  1. We assume that the running time of all lines of $P$ are equal.
  2. $t_P(x)$ denotes the number of lines that are executed for $x$.
  3. We use Big-$\Theta$ to give an estimation for $T_P(n)$.

Running Time Analysis for Linear Search (LS)

**Claim:** $T_{LS} \in \Theta(n)$

def LS(A, x):
    
    """ Return an index $i$ such that $x == L[i]$. Otherwise, return -1. """
    i = 0 # (line 1)
    while i < len(A): # (line 2)
        if A[i] == x: # (line 3)
            return i # (line 4)
        i = i + 1 # (line 5)
    return -1 # (line 6)
Finding a Tight Bound

\( T_P \in \Theta(W) \) iff \( T_P \in \mathcal{O}(W) \) and \( T_P \in \Omega(W) \).

Finding an Upper Bound

\( T_P \in \mathcal{O}(W) \) iff

\[ \exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq B \Rightarrow T_P(n) \leq cW(n) \]  iff

\[ \exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq B \Rightarrow \max\{t_P(x) \mid x \in I \wedge \text{size}(x) = n\} \leq cW(n) \]

To prove \( T_P \in \mathcal{O}(W) \), must show that:

\[ \exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall x \in I, \text{size}(x) \geq B \Rightarrow t_P(x) \leq cW(\text{size}(x)) \]

Finding a Lower Bound

\( T_P \in \Omega(W) \) iff

\[ \exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq B \Rightarrow \max\{t_P(x) \mid x \in I \wedge \text{size}(x) = n\} \geq cW(n) \]

To prove \( T_P \in \mathcal{O}(W) \), must show that:

\[ \exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq B \Rightarrow \exists x \in I, \text{size}(x) = n \wedge t_P(x) \geq cW(n) \]
Example #1

Running Time Analysis for Linear Search (LS)

- **Claim:** $T_{LS} \in \mathcal{O}(n)$

```
    def LS(A,x):
        """ Return an index i such that x == L[i]. """
        n = len(A) # (line 1)
        i = 0

        while i < len(A):  # (line 2)
            if A[i] == x:
                return i
            i = i + 1

        return -1 # (line 6)
```

- Final loop check

- At most $n+1$
- At most $n$
- At most $n$
- At most $1$

Mathematical Expression and Reasoning
Example #1

Running Time Analysis for Linear Search (LS)

- **Claim**: $T_{LS} \in \mathcal{O}(n)$
  
  Assume $n \in \mathbb{N}$, $A$ is an array of length $n$.
  Then lines 2–5 execute not more than $3n$ steps, plus 1 step for the last loop test. (add the explanations from the previous slide for justification)
  So lines 1–6 take no more than $3n + 3$ steps.
  Then $\forall n \in \mathbb{N}$ and all possible inputs $A, x$, $t_{LS}(A)$ is less than $3n + 3$ steps.

- $\exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq B \Rightarrow 3n + 3 \leq cn$
  
  Let $c = 6$. Let $B = 1$. Then $c \in \mathbb{R}^+$ and $B \in \mathbb{N}$.
  Assume $n \in \mathbb{N}$, $A$ is an array of length $n$, and $n \geq B$.
  Then $3n \leq 3n$. # both sides are equal
  Then $3n + 3 \leq 3n + 3n = 6n$. # $n \geq 1$
  Then $\exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq B \Rightarrow T_{LS}(n) \leq cn$. # definition of $T_{LS}$
  Then $T_{LS} \in \mathcal{O}(n)$. # by definition of $\mathcal{O}(n)$
Example #1

Running Time Analysis for Linear Search (LS)

- **Claim:** $T_{LS} \in \Omega(n)$

  Assume $x$ is equal to the middle entry of $A$

  ```python
def LS(A, x):
    """ Return an index $i$ such that $x == L[i]$. 
    Otherwise, return -1. """
    i = 0
    # (line 1)
    while i < len(A):
      at most $n/2 + 1$ # (line 2)
      if A[i] == x:
        at most $n/2$ # (line 3)
        return i
        # (line 4)
        i = i + 1
        at most $n/2 - 1$ # (line 5)
    return -1 # (line 6)
  ```
Example #1

Running Time Analysis for Linear Search (LS)

- **Claim:** $T_{LS} \in \Omega(n)$

  Assume $n \in \mathbb{N}$, Let $A$ is an array of length $n$ and $x = A[[n/2]]$. Then lines 2–5 execute not more than $3\lfloor n/2 \rfloor$ steps. (add the explanations from the previous slide for justification) So lines 1–6 take no less than $3\lfloor n/2 \rfloor + 1 \geq 3n/2$ steps.
  Then $\forall n \in \mathbb{N}$ and all possible inputs $A$, $t_{LS}(A)$ is greater than $3n/2$ steps.

- $\exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq B \Rightarrow 3n/2 \geq cn$

  Let $c = 1/2$. Let $B = 1$. Then $c \in \mathbb{R}^+$ and $B \in \mathbb{N}$.
  Assume $n \in \mathbb{N}$, $A$ is an array of length $n$ and $x = A[[n/2]]$, and $n \geq B$. Then $3n/2 \geq n/2$. # $3 > 1$
  Then $\exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq B \implies T_{LS}(n) \geq cn$. # definition of $T_{LS}$
  Then $T_{LS} \in \Omega(n)$. # by definition of $\Omega(n)$
Example #2

Running Time Analysis for Insertion Sort (IS)

- **Claim:** \( T_{IS} \in \mathcal{O}(n^2) \)

```
def IS(A):
    """ Sort the elements of A in non-decreasing order. """
    i = 1  
    # (line 1)
    while i < len(A):  
        at most \( n+1 \)  
        # (line 2)
        t = A[i]  
        at most \( n \)  
        # (line 3)
        j = i  
        at most \( n \)  
        # (line 4)
        while j > 0 and A[j-1] > t:
            # (line 5)
            each iteration \( n \)
            at most \( n \)
            \[
            \begin{align*}
            j &= j-1
            \end{align*}
            \]  
            # (line 6)
            # (line 7)
        A[j] = t  
        at most \( n \)  
        # (line 8)
        i = i+1  
        at most \( n \)  
        # (line 9)
```
Example #2

Running Time Analysis for Insertion Sort (IS)

- **Claim:** $T_{IS} \in \mathcal{O}(n^2)$
  
  Assume $n \in \mathbb{N}$, $A$ is an array of length $n$.
  Then lines 5–7 execute not more than $3n$ steps, plus 1 step for the last loop test. (add the explanations from the previous slide for justification)
  Then lines 2–9 take no more than $n(5 + 3n) + 1 = 5n + 3n^2 + 1$ steps. (add the explanations from the previous slide for justification)
  So lines 1–9 take no more than $5n + 3n^2 + 2$ steps.
  Then $\forall n \in \mathbb{N}$ and all possible inputs $A$, $t_p(A)$ is less than $5n + 3n^2 + 2$ steps.

- $\exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq B \Rightarrow 3n^2 + 5n + 2 \leq cn^2$
  
  Let $c = 9$. Let $B = 1$. Then $c \in \mathbb{R}^+$ and $B \in \mathbb{N}$.
  Assume $n \in \mathbb{N}$; $A$ is an array of length $n$, and $n \geq B$.
  Then $3n^2 \leq 3n^2$. # both sides are equal
  Then $3n^2 + 5n + 2 \leq 3n^2 + 5n^2 + 2n^2 = 9n^2$. # $n \geq 1$
  Then $\exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq B \implies T_{IS}(n) \leq cn^2$. # definition of $T_{IS}$
  Then $T_{IS} \in \mathcal{O}(n^2)$. # by definition of $\mathcal{O}(n^2)$
Example #2

Running Time Analysis for Insertion Sort (IS)

- **Claims:** $T_{IS} \in \Omega(n^2)$  
  assume $A=[n-1,n-2,\ldots,0]$

```python
def IS(A):
    """ Sort the elements of A in non-decreasing order. """
    i = 1
    # (line 1)
    while i < len(A): at most n
        t = A[i] at most n-1
        j = i at most n-1
        # (line 3)
        # (line 4)
        while j > 0 and A[j-1] > t:
            # (line 5)
            sum of 3i+1 for i=1 to n-1
            j = j-1
            # (line 6)
            # (line 7)
            A[j] = t at most n-1
            i = i+1 at most n-1
            # (line 8)
            # (line 9)
```
Running Time Analysis for Insertion Sort (IS)

**Claim:** $T_{IS} \in \Omega(n^2)$

Assume $n \in \mathbb{N}$, $A = [n - 1, n - 2, \ldots, 0]$.
Then lines 5–7 execute $3i$ steps each iteration of the outer loop, plus 1 step for the final loop check.
Then lines 2–9 take $n + 4(n - 1) + 3n(n - 1)/2 + (n - 1) = 3n^2/2 + 9n/2 - 5$ steps. (add the explanations from the previous slide for justification)
Then lines 1–9 take $3n^2/2 + 9n/2 - 4$ steps.
Then $\forall n \in \mathbb{N}$ and all possible inputs $A$, $t_p(A)$ is less than $3n^2/2 + 9n/2 - 4$ steps.

$\exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq B \Rightarrow 3n^2/2 + 9n/2 - 4 \geq cn^2$

Let $c = 1$. Let $B = 1$. Then $c \in \mathbb{R}^+$ and $B \in \mathbb{N}$.
Assume $n \in \mathbb{N}$, $A$ is an array of length $n$, and $n \geq B$.
Then $3n^2/2 \geq n^2$. # $3/2 > 1$
Then $3n^2/2 + 9n/2 - 4 \geq n^2$. # $n \geq 1$, so $9n/2 - 4 > 0$
Then $\exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq B \Rightarrow T_{IS}(n) \geq cn^2$. # definition of $T_{IS}$
Then $T_{IS} \in \Omega(n^2)$. # by definition of $\Omega(n^2)$