DISCUSSION #4
FRIDAY APRIL 27TH 2007

Sophie Engle
ECS20: Discrete Mathematics

Homework 4: Hints

Due Wednesday May 2nd

Motivation

Problem Statement:

How do we estimate and compare
the runtime of different algorithms?

Motivation

Problem Statement:

How do we estimate and compare
the runtime of different algorithms?

What is the "fastest" sort algorithm?

Fastest in theory?

Fastest on average?

Fastest in practice?

Fastest for big numbers?

Fastest for small numbers?

Fastest possible?

Motivation

Problem Statement:

How do we estimate and compare
the runtime of different algorithms?

Solution:

- Measure number of operations as size of input grows
  - Input size: n
  - Number of operations: $f(n)$
- Estimate the runtime class of algorithm
  - Estimate upper bound: $\mathcal{O}(\ f(n)\ )$
  - Estimate lower bound: $\Omega(\ f(n)\ )$

Homework 4: Hints

Estimating Number of Operations

Example: Bubble Sort

- Bubble Sort Algorithm
  - Method of sorting elements of a set
    - Small numbers "bubble" up to the top
    - Large numbers "sink" to the bottom
  - Visualization
    - www.wanginator.de/studium/applets/bubblesort_en.html

Example: Bubble Sort

- Algorithm:

procedure bubbleSort($a_1, a_2, \ldots, a_n$)
    for $i = 0$ to $n - 1$
        for $j = 0$ to $n - i$
            if $a_j > a_{j+1}$ then
                swap(a_j, a_{j+1})

Example: Bubble Sort

- Algorithm:

input of n elements

procedure bubbleSort($a_1, a_2, \ldots, a_n$)
    for $i = 0$ to $n - 1$
        for $j = 0$ to $n - i$
            if $a_j > a_{j+1}$ then
                swap(a_j, a_{j+1})

Example: Bubble Sort

- Algorithm:

outer loops n times

```
procedure bubbleSort( a_1, a_2 , … , a_n)
    for i = 0 to n − 1
        for j = 0 to n − i
            if a_j > a_{j+1} then
                swap( a_j, a_{j+1} )
```

Example: Bubble Sort

- Algorithm:

with inner loop $n(n-1)/2$ times

```
procedure bubbleSort( a_1, a_2 , … , a_n)
    for i = 1 to n − 1
        for j = 1 to n − i
            if a_j > a_{j+1} then
                swap( a_j, a_{j+1} )
```

Example: Bubble Sort

- Algorithm:

worst case executes every time

```
procedure bubbleSort( a_1, a_2 , … , a_n)
    for i = 1 to n − 1
        for j = 1 to n − i
            if a_j > a_{j+1} then
                swap( a_j, a_{j+1} )
```

(assume swap operation takes constant time)

Example: Bubble Sort

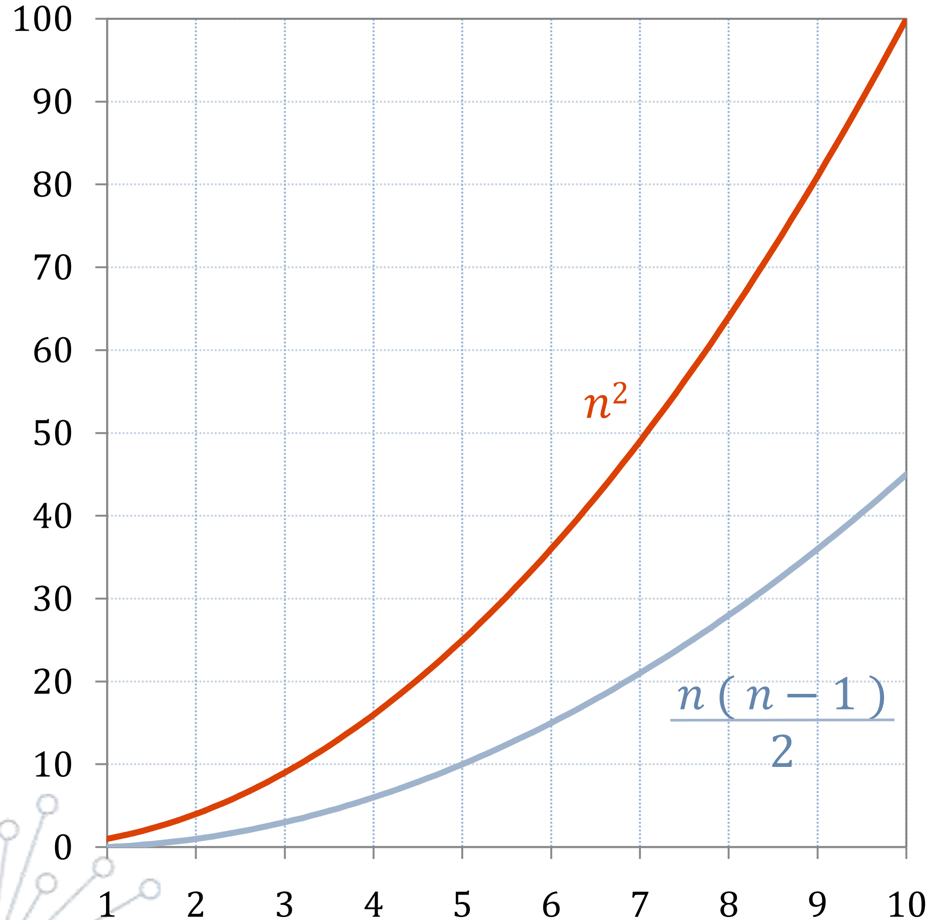
- Algorithm:

```
procedure bubbleSort( a_1, a_2, ..., a_n)
    for i = 1 to n − 1
        for j = 1 to n − i
            if a_j > a_{j+1} then
                swap( a_j, a_{j+1} )
```

makes approximately:

$$\frac{n(n-1)}{2} = \tfrac{1}{2}n^2 - \tfrac{1}{2}n \text{ operations}$$

Example: Bubble Sort



- Notice that n^2:
  - Bounds the number of operations
  - Provides approximation of operations

Homework 4: Hints

Growth of Functions

Growth of Functions

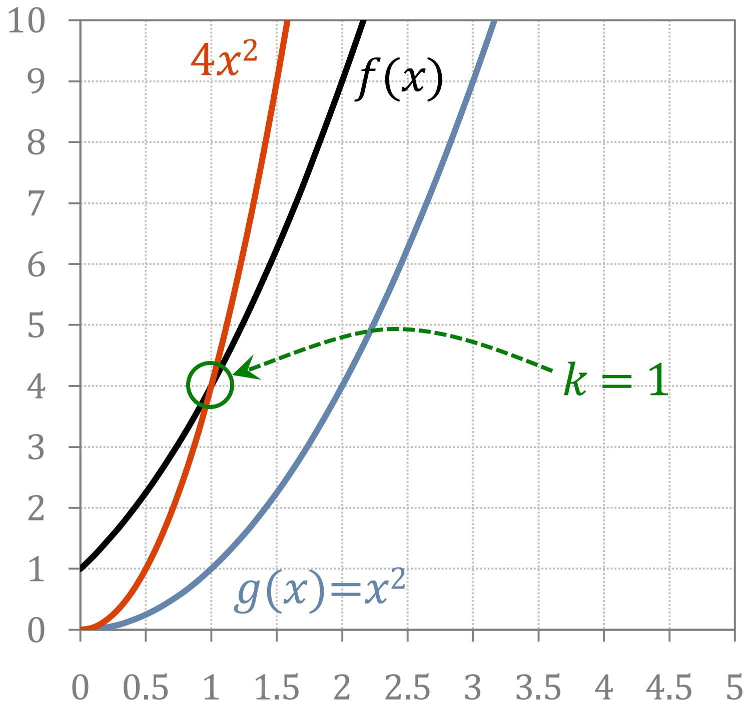
- General Idea:
  - Analyze algorithm
    - Come up with function $f(n)$ which returns the number of operations for an input of size n
  - Approximate number of operations
    - Use $\mathcal{O}(n)$ to find a upper bound
    - Use $\Omega(n)$ to find a lower bound
  - Determine class of function
    - Linear $\mathcal{O}(1)$
    - Logarithmic $\mathcal{O}(\log n)$
    - Polynomial $\mathcal{O}(n^k)$
    - Exponential $\mathcal{O}(k^n)$

Big-$\mathcal{O}$ Notation

- Upper bound estimate
  - Estimates growth for large inputs
    - Care more about exponents
    - Less about constants
- A function $f(x) \in \mathcal{O}(\ g(x)\ )$ when:
  - $\exists$ constants (called **witnesses**) C and k such that:
    - $|f(x)| \leq C\ |g(x)|$
    - whenever $x > k$
  - i.e. approximately whenever $g(x)$ bounds $f(x)$ without its constants

Big-$\mathcal{O}$ Notation

- Show $f(x) = x^2 + 2x + 1$ is $\mathcal{O}(x^2)$.
  - To show this, you must provide the witnesses!



With a graph, we can see that the following witnesses work:

$C = 4$

$k = 1$

Big-$\mathcal{O}$ Notation

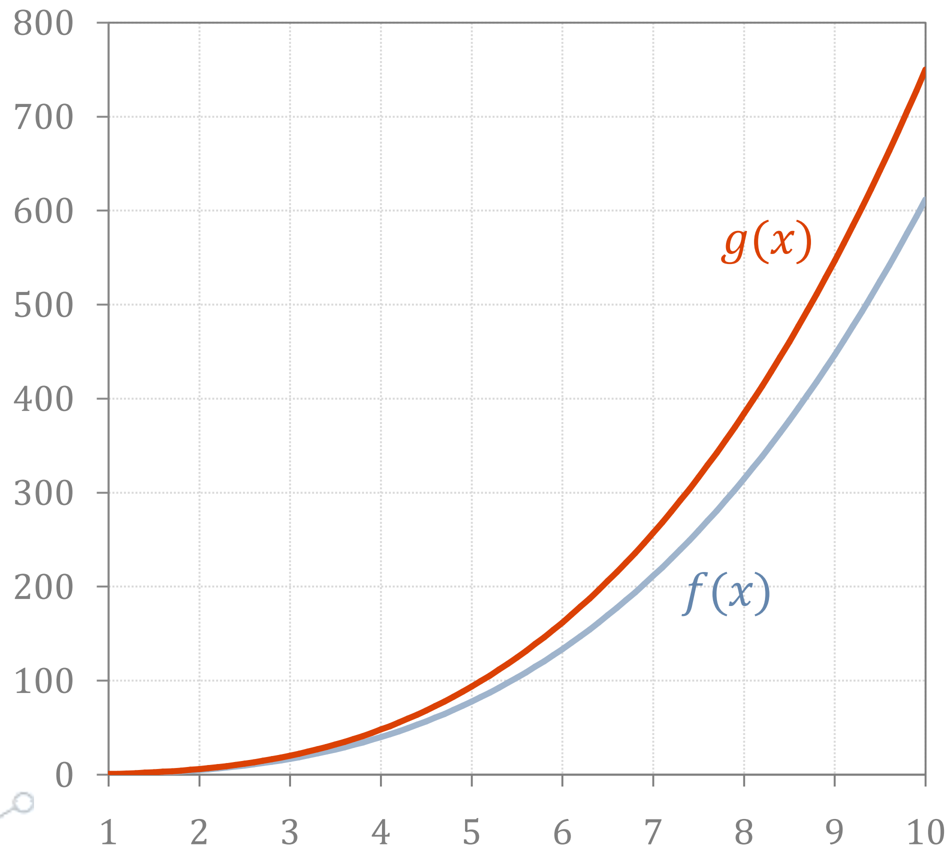
- Show $f(x) = x^2 + 2x + 1$ is $\mathcal{O}(x^2)$.
  - Without a graph, just start approximating:
    - The maximum exponent is 2, so should be able to find witnesses C and k for $g(x) = x^2$.
    - Notice when $x > 1$ then $x^2 > x$ and $2x^2 > 2x$
    - Thus we can write:
      - $x^2 + 2x^2 + x^2 > x^2 + 2x + 1$ which means…
      - $4x^2 > x^2 + 2x + 1$
    - Therefore we can set $C = 4$ and $k = 1$.
  - Is $f(x)$ also $\mathcal{O}(x^3)$?
    - Yes, but less useful as an upper bound!

Big-$\mathcal{O}$ Notation

- Finding the big-$\mathcal{O}$ estimate:
  - Don't need smallest C and k possible.
    - Just find witnesses that are easy to come by!
  - However, want tightest $g(x)$ possible.
    - With polynomial functions, choose a $g(x)$ with the lowest possible exponent.
  - For large x:
    - $1 < \log x < x < x \log x < x^2 < 2^x < x!$
    - (see graph in book)

Big-$\mathcal{O}$ Examples

- Find big-$\mathcal{O}$ for $f(x) = (3^4 - 2x) / (5x - 1)$.



$$f(x) = \frac{3x^4 - 2x}{5x - 1}$$

$$g(x) = x^3$$

$$C = \frac{3}{4}$$

$$k = 1$$

Big-$\mathcal{O}$ Examples

- Find big-$\mathcal{O}$ for $f(x) = \log_{10}(2^x) + 10^{10}\ x^2$.

$$g(x) = x^2$$

$$C = 2 + 10^{10}$$

$$k = 0$$

Big-Ω and Big-Θ Notation

- Big-Ω (Omega) Notation
  - Provides lower bound for large x
  - A function $f(x) \in \Omega(\, g(x) \,)$ when:
    - $|f(x)| \geq C\,|g(x)|$ for witnesses C, k whenever $x > k$
- Big-Θ (Theta) Notation
  - Provides both upper and lower bound for large x
  - A function $f(x) \in \Theta(\, g(x) \,)$ when:
    - $f(x) \in \mathcal{O}(\, g(x) \,)$
    - $f(x) \in \Omega(\, g(x) \,)$

Big-Ω and Big-Θ Example

- Show $f(x) = 7x^2 + 1$ is $\Theta(x^2)$.
  - Show $f(x)$ is $\mathcal{O}(x^2)$.
    - $7x^2 + 1 \leq 7x^2 + x^2 = 8x^2$ where $x \geq 1$
  - Show $f(x)$ is $\Omega(x^2)$.
    - $7x^2 + 1 \geq 7x^2$ where $x \geq 1$
  - Therefore, $f(x)$ is $\Theta(x^2)$.