Objectives

- Review: Asymptotic running times
- Implementing Gale-Shapley algorithm
- Classes of running times

Review Asymptotic Bounds

- What does $O(f(n))$ mean?

Asymptotic Order of Growth: Upper Bounds

- $T(n)$ is the worst case running time of an algorithm
- We say that $T(n)$ is $O(f(n))$ if there exist constants $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$, we have $T(n) \leq c \cdot f(n)$.

$\Rightarrow T$ is asymptotically upperbounded by $f$.

Upper Bounds Example

- Find an upperbound for $T(n) = pn^2 + qn + r$
  - $p$, $q$, $r$ are positive constants

Idea: Let's inflate the terms in the equation so that all terms are $n^2$.

- $T(n)$ is bounded above by a constant multiple of $f(n)$.

$\Rightarrow T(n) \leq cn^2$, where $c = p + q + r$

$T(n) = O(n^3)$
- Also correct to say that $T(n) = O(n^3)$. 

Upper Bounds Example

- $T(n) = pn^2 + qn + r$
  - $p$, $q$, $r$ are positive constants
  - For all $n \geq 1$,

<table>
<thead>
<tr>
<th>$T(n)$</th>
<th>$= pn^2 + qn + r$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\leq pn^2 + qn^2 + rn^2$</td>
</tr>
<tr>
<td></td>
<td>$= (p + q + r) n^2$</td>
</tr>
<tr>
<td></td>
<td>$= c n^2$</td>
</tr>
</tbody>
</table>

$\Rightarrow T(n) \leq cn^2$, where $c = p + q + r$

$T(n) = O(n^3)$
- Also correct to say that $T(n) = O(n^3)$. 

Notation
- $T(n) = O(f(n))$ is a slight abuse of notation
  - Asymmetric: $f(n) = 5n^2$, $g(n) = 3n^2$
  - But $f(n) 
eq O(g(n))$
  - Better notation: $T(n) \in O(f(n))$
- Meaningless statement. Any comparison-based sorting algorithm requires at least $O(n \log n)$ comparisons
  - Use $\Omega$ for lower bounds

Example: Lower Bound
- $T(n) = pn^2 + qn + r$
  - $p$, $q$, $r$ are positive constants
- Idea: Deflate terms rather than inflate
- For all $n \geq 0$,
  - $T(n) = pn^2 + qn + r \geq pn^2$
    - $T(n) \geq \varepsilon n^2$, where $\varepsilon = p > 0$
    - $T(n) = \Omega(n^2)$
- Also correct to say that $T(n) = \Omega(n)$

Asymptotic Order of Growth: Lower Bounds
- Complementary to upper bound
  - $T(n)$ is $\Omega(f(n))$ if there exist constants $\varepsilon > 0$ and $n_0$ such that for all $n \geq n_0$, we have $T(n) \geq \varepsilon \cdot f(n)$
  - $T(n)$ is bounded below by a constant multiple of $f(n)$
  - $T(n)$ is asymptotically lower bounded by $f(n)$

Tight bounds
- $T(n)$ is $\Theta(f(n))$ if $T(n)$ is both $O(f(n))$ and $\Omega(f(n))$
  - The “right” bound

Property: Transitivity
- If $f = O(g)$ and $g = O(h)$ then $f = O(h)$
- If $f = \Omega(g)$ and $g = \Omega(h)$ then $f = \Omega(h)$
- If $f = \Theta(g)$ and $g = \Theta(h)$ then $f = \Theta(h)$

Property: Additivity
- If $f = O(h)$ and $g = O(h)$ then $f + g = O(h)$
- If $f = \Omega(h)$ and $g = \Omega(h)$ then $f + g = \Omega(h)$
- If $f = \Theta(h)$ and $g = \Theta(h)$ then $f + g = \Theta(h)$

Sketch proof for $O$:
- By defn, $f \leq c \cdot h$
- By defn, $g \leq d \cdot h$
- $f + g \leq c \cdot h + d \cdot h = (c + d) \cdot h = c' \cdot h$
- $\Rightarrow f + g \in O(h)$
Practice: Asymptotic Order of Growth

What are the upper bounds, lower bounds, and tight bound on $T(n)$?

- $T(n) = 32n^2 + 17n + 32$

Jan 19, 2016  Sprenkle - CSCI211

Asymptotic Bounds for Polynomials

- $a_0 + a_1 n + \ldots + a_d n^d \in \Theta(n^d)$ if $a_d > 0$
  ➔ Runtime determined by highest-order term

- Polynomial time. Running time is $O(n^d)$ for some constant $d$ that is independent of the input size $n$

- Other examples of polynomial times:
  - $O(n^{1/2})$
  - $O(n^{1.58})$
  - $O(n \log n) \leq O(n^2)$

Jan 19, 2016  Sprenkle - CSCI211

Asymptotic Bounds for Logarithms

- Logarithms. $\log_b n = x$, where $b^x = n$
  ➔ Approximate: To represent $n$ in base-$b$, need $x + 1$ digits

<table>
<thead>
<tr>
<th>$n$</th>
<th>$b$</th>
<th>$x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>1000</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>100</td>
<td>2</td>
<td>6.64</td>
</tr>
<tr>
<td>1000</td>
<td>2</td>
<td>9.92</td>
</tr>
</tbody>
</table>

Jan 19, 2016  Sprenkle - CSCI211

Asymptotic Bounds for Logarithms

- Logarithms. $\log_b n = x$, where $b^x = n$
  ➔ Approximate: To represent $n$ in base-$b$, need $x + 1$ digits

<table>
<thead>
<tr>
<th>$n$</th>
<th>$b$</th>
<th>$x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>1000</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>100</td>
<td>2</td>
<td>6.64</td>
</tr>
<tr>
<td>1000</td>
<td>2</td>
<td>9.92</td>
</tr>
</tbody>
</table>

Describe the running time of an $O(\log n)$ algorithm as the input size grows. Compare with polynomials.
Asymptotic Bounds for Logarithms

- **Logarithms.** $\log_b n = x$, where $b^x = n$
  - $x$ is number of digits to represent $n$ in base-$b$ representation
  - **Slowly growing functions**
  - **Identity:** $\log_b n = \frac{\log_a n}{\log_a b}$
  - Means that $\log_a n = \frac{1}{\log_b a} \cdot \log_b n$
  - Constant!
  - $O(\log_a n) = O(\log_b n)$ for any constants $a, b > 0$

Asymptotic Bounds for Exponentials

- **Exponentials:** functions of the form $f(n) = r^n$ for constant base $r$
  - Faster growth rates as $n$ increases
  - For every $r > 1$ and every $d > 0$, $n^d = O(r^n)$
  - **Every exponential grows faster than every polynomial**

Summary of Asymptotic Bounds

- In terms of growth rates ....
  - $\text{Logarithms} < \text{Polynomials} < \text{Exponentials}$
- Practice comparing functions on next problem set
  - See Chapter 2 solved exercise

Review: Our Process

1. Understand/identify problem
   - Simplify as appropriate
2. Design a solution
3. Analyze
   - Correctness, efficiency
   - May need to go back to step 2 and try again
4. Implement
   - Within bounds shown in analysis
IMPLEMENTING GALE-SHAPLEY ALGORITHM

Jan 19, 2016
Sprenkle - CSCI211

Review:
Gale-Shapley Stable Matching Algorithm

Initialize each person to be free
while (some man is free and hasn't proposed to every woman)
    Choose such a man \( m \)
    \( w = 1^{st} \) woman on \( m \)'s list to whom \( m \) has not yet proposed
    if \( w \) is free
        assign \( m \) and \( w \) to be engaged
    else if \( w \) prefers \( m \) to her fiancé \( m' \)
        assign \( m \) and \( w \) to be engaged and \( m' \) to be free
    else
        \( w \) rejects \( m \)

How Can We Implement The Algorithm Efficiently?

• What is our goal for the implementation’s runtime?
• What do we need to model?
• How should we represent them?

Stable Matching Implementation

• What do we need to represent?
• How should we represent them?

<table>
<thead>
<tr>
<th>Data</th>
<th>How represented</th>
</tr>
</thead>
<tbody>
<tr>
<td>Men, Women</td>
<td></td>
</tr>
<tr>
<td>Preference lists</td>
<td></td>
</tr>
<tr>
<td>Unmatched men</td>
<td></td>
</tr>
<tr>
<td>Who men proposed to</td>
<td></td>
</tr>
<tr>
<td>Engagements</td>
<td></td>
</tr>
</tbody>
</table>

Arrays

• Fixed number of elements
• What is the runtime of
  ➢ Determining the value of the \( i^{th} \) item in the array?
  ➢ Determining if a value \( e \) is in the array?
  ➢ Determining if a value \( e \) is in the array if the array is sorted?
Array Operations’ Running Times

<table>
<thead>
<tr>
<th>Operation</th>
<th>Running Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value of (i^{th}) item</td>
<td>(O(1)) → direct access</td>
</tr>
<tr>
<td>If (e) is in the array</td>
<td>(O(n)) → look through all the elements</td>
</tr>
<tr>
<td>If (e) is in the array if sorted</td>
<td>(O(\log n)) → binary search</td>
</tr>
</tbody>
</table>

Limitation of arrays?

Fixed size, so difficult to add/delete elements

Lists

- Dynamic set of elements
  - Linked list
  - Doubly linked list
- What is the running time to
  - Add an element to the list?
  - Delete an element from the list?
  - Find an element \(e\) in the list?
  - Find the \(i^{th}\) element in the list?

List Operations’ Running Time

<table>
<thead>
<tr>
<th>Operation</th>
<th>Running Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add element</td>
<td>(O(1))</td>
</tr>
<tr>
<td>Delete element</td>
<td>(O(1))</td>
</tr>
<tr>
<td>Find element</td>
<td>(O(n))</td>
</tr>
<tr>
<td>Find (i^{th}) element</td>
<td>(O(i))</td>
</tr>
</tbody>
</table>

Disadvantage of list instead of array?

Finding \(i^{th}\) element is slower

Converting between Lists and Arrays (and Vice Versa)

- What is the running time of converting a list to an array?
- An array to a list?
  \(O(n)\)

Stable Matching Implementation

- What do we need to represent? How should we represent them?

Data | How represented
---|---
Men, Women | Integers (like ids)
Preference lists | Array of arrays (2d array)
Unmatched men | List
Who men proposed to | Integer for each man → Array of integers
Engagements | 2 Arrays
Looking Ahead

• Review Chapter 2
  ➢ Finishing up on Thursday
• Return surveys – Thursday class
• Problem Set 1 due by 5 p.m. Friday