Outline

Motivation: Census Problem
• Abstract Data Type (ADT) Table
• Solving Census Problem with CS1020 Knowledge
• The “performance issue”

Binary Search Tree (BST)
• Heavy usage of VisuAlgo Binary Search Tree Visualization
• Simple analysis of BST operations
• Java Implementation

PS2 Preview
Census is Important!

Source: http://www.singstat.gov.sg
Sun Tzu’s Art of War
Chapter 1 “The Calculations”

知彼知己百战不殆
zhī bǐ zhī jǐ bǎi zhàn bù dài

(If you know your enemies and know yourself, you will not be imperiled in a hundred battles)
‘[’ (or ‘]’) means that endpoint is included (closed)

1. \([24 \ldots \infty)\)
2. \([23 \ldots 24)\)
3. \([22 \ldots 23)\)
4. \([21 \ldots 22)\)
5. \([20 \ldots 21)\)
6. \([19 \ldots 20)\)
7. \([18 \ldots 19)\)
8. \([17 \ldots 18)\)
9. \([0 \ldots 17)\)

Mean = 3.9615
Your Major (2013 data)

1. Computer Science (CS)
2. Communications and Media (C&M)
3. Computer Engineering (CEG/CEC)
4. Comp. Biology (CB)
5. Information System (IS)
6. Science Maths (SCI)
7. None of the above :O
1. Singaporean (should be ≥ 70% according to MOE rules)
2. Chinese
3. Indian
4. Indonesian
5. Vietnamese
6. Malaysian
7. European
8. None of the above
Your CAP (2013 data)

1. [4.5 ... 5.0]
2. [4.25 ... 4.5)
3. [4.0 ... 4.25)
4. [3.75 ... 4.0)
5. [3.5 ... 3.75)
6. [3.25 ... 3.5)
7. [3.0 ... 3.25)
8. [0.0 ... 3.00)
9. I do not want to tell
What Happen After Census?

Data Mining

Statistical Analysis
Abstract Data Type (ADT) Table

Let’s deal with one aspect of our census: **Age**

To simplify this lecture, we assume that students’ age ranges from [0 … 100), all integers, and distinct

**Required operations:**
1. Search whether there is a student with a certain age?
2. Insert a new student (that is, insert his/her age)
3. Determine the youngest and oldest student
4. List down the ages of students in sorted order
5. Find a student slightly older than a certain age!
6. Delete existing student (that is, remove his/her age)
7. Determine the median age of students
8. How many students are younger than a certain age?
# CS1020: Unsorted Array

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>5</td>
<td>7</td>
<td>71</td>
<td>50</td>
<td>23</td>
<td>4</td>
<td>6</td>
<td>15</td>
</tr>
</tbody>
</table>

## Operations and Time Complexities

<table>
<thead>
<tr>
<th>No</th>
<th>Operation</th>
<th>Time Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Search(age)</td>
<td>O(n)</td>
</tr>
<tr>
<td>2</td>
<td>Insert(age)</td>
<td>O(1)</td>
</tr>
<tr>
<td>3</td>
<td>FindOldest()</td>
<td>O(n)</td>
</tr>
<tr>
<td>4</td>
<td>ListSortedAges()</td>
<td>O(n log n)</td>
</tr>
<tr>
<td>5</td>
<td>NextOlder(age)</td>
<td>O(n)</td>
</tr>
<tr>
<td>6</td>
<td>Remove(age)</td>
<td>O(n)</td>
</tr>
<tr>
<td>7</td>
<td>GetMedian()</td>
<td>O(n log n)/O(n)</td>
</tr>
<tr>
<td>8</td>
<td>NumYounger(age)</td>
<td>O(n log n)</td>
</tr>
</tbody>
</table>
## CS1020: Sorted Array

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>15</td>
<td>23</td>
<td>50</td>
<td>71</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No</th>
<th>Operation</th>
<th>Time Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Search(age)</td>
<td>(O(\log n))</td>
</tr>
<tr>
<td>2</td>
<td>Insert(age)</td>
<td>(O(n))</td>
</tr>
<tr>
<td>3</td>
<td>FindOldest()</td>
<td>(O(1))</td>
</tr>
<tr>
<td>4</td>
<td>ListSortedAges()</td>
<td>(O(n))</td>
</tr>
<tr>
<td>5</td>
<td>NextOlder(age)</td>
<td>(O(\log n))</td>
</tr>
<tr>
<td>6</td>
<td>Remove(age)</td>
<td>(O(n))</td>
</tr>
<tr>
<td>7</td>
<td>GetMedian()</td>
<td>(O(1))</td>
</tr>
<tr>
<td>8</td>
<td>NumYounger(age)</td>
<td>(O(\log n))</td>
</tr>
</tbody>
</table>
## With Just CS1020 Knowledge

<table>
<thead>
<tr>
<th>No</th>
<th>Operation</th>
<th>Unsorted Array</th>
<th>Sorted Array</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Search(age)</td>
<td>O(n)</td>
<td>O(log n)</td>
</tr>
<tr>
<td>2</td>
<td>Insert(age)</td>
<td>O(1)</td>
<td>O(n)</td>
</tr>
<tr>
<td>3</td>
<td>FindOldest()</td>
<td>O(n)</td>
<td>O(1)</td>
</tr>
<tr>
<td>4</td>
<td>ListSortedAges()</td>
<td>(n log n)</td>
<td>O(n)</td>
</tr>
<tr>
<td>5</td>
<td>NextOlder(age)</td>
<td>O(n)</td>
<td>O(log n)</td>
</tr>
<tr>
<td>6</td>
<td>Remove(age)</td>
<td>O(n)</td>
<td>O(n)</td>
</tr>
<tr>
<td>7</td>
<td>GetMedian()</td>
<td>O(n log n) / O(n)</td>
<td>O(1)</td>
</tr>
<tr>
<td>8</td>
<td>NumYounger(age)</td>
<td>O(n log n)</td>
<td>O(log n)</td>
</tr>
</tbody>
</table>

If $n$ is large, our queries are slow...
O(n) versus O(log n): A Perspective

Try larger n, e.g. n = 1000000...
A Versatile, Non-Linear Data Structure

BINARY SEARCH TREE (BST)
Binary Search Tree (BST) Vertex

For every vertex \( x \), we define:

- \( x.\text{left} = \) the left child of \( x \)
- \( x.\text{right} = \) the right child of \( x \)
- \( x.\text{parent} = \) the parent of \( x \)
- \( x.\text{key} \) (or \( x.\text{value} \), \( x.\text{data} \)) = the value stored at \( x \)

BST Property:

- \( x.\text{left}.\text{key} < x.\text{key} \leq x.\text{right}.\text{key} \)
- For simplicity, we assume that the keys are unique so that we can change \( \geq \) to \( > \)
BST: An Example, Keys = Ages

**Recursive definition**

- **Root**: 15
- **Leaves**: All other vertices other than 15, 5, 7, and 50 are Internal vertices.

Recursive definition:
- \( x \cdot \text{key} < x \cdot \text{key} \geq x \cdot \text{key} \)
BST: Search/Min/Max Operations

Ask VisuAlgo to perform various search operations on the sample BST, including find min and find max.

In the screen shot below, we show search(5)
BST: Succ/Predecessor Operations

Ask VisuAlgo to perform Succ/Pred operations on the sample BST

In the screen shot below, we show pred(15)
BST: Inorder Traversal Operation

Ask VisuAlgo to perform inorder traversal operation on the sample BST

In the screen shot below, we partial inorder traversal
BST: Select/Rank Operations

These 2 operations will be added to VisuAlgo BST visualization soon; for now, here are the concepts:

• Select(k) – Return the value v of k-th smallest* element
  – Examples: Select(1) = 4, Select(3) = 6, Select(8) = 71, etc (1-based index)

• Rank(v) – Return the ranking* k of element v
  – Examples: Rank(4) = 1, Rank(6) = 3, Rank(71) = 8, etc

• Details will be discussed in the next lecture
BST: Insert Operation

Ask VisuAlgo to perform various insert operations on the sample BST

In the screen shot below, we show insert(20)
BST: Delete/Remove Operation (1)

Ask VisuAlgo to perform various delete operations on the sample BST (3 cases, this is delete leaf)

In the screen shot below, we show remove(5) before deletion
BST: Delete/Remove Operation (2)

Ask VisuAlgo to perform various delete operations on the sample BST (this is delete vertex with one child).

In the screenshot below, we show remove(23) before relayout.
BST: Delete/Remove Operation (3)

Ask VisuAlgo to perform various delete operations on the sample BST (delete vertex with two children)

In the screenshot below, we show remove(6) before relayout.

Later, we will analyze on why replacing the deleted item with its successor (and delete the old successor) works.
ANALYSIS OF BST OPERATIONS
search(51)

Quick analysis:
search runs
in $O(h)$

BST: Search Analysis

$h = \text{Height of BST}$

51 is not found 😞
Quick analysis:
findMin/findMax
also runs in $O(h)$
**BST: Successor/Predecessor Analysis**

Assumption, we already done an \(O(h)\) search(71) before

**successor(71)**

Quick analysis:

\(O(h)\) again, similarly for predecessor

Keep going up until we make a ‘right turn’, but here we do not find such vertex, so there is no successor for 71

No right child
BST: Inorder Traversal Analysis

Using a *new* analysis technique

Ask this question:

• How many times a vertex is *touched* during inorder traversal from the start until the end?

Answer:

• Three times: from parent and from left + right children (even if one or both of them is/are empty/NUL

• $O(3n) = O(n)$
BST: Select/Rank Analysis

We have not explored the operations in detail yet.

This will be discussed in more details in the next lecture.
BST: Insertion Analysis

**insert** (50)

Quick analysis:

**insert** also runs in $O(h)$

1. **15 < 50**, go right
2. **23 < 50**, go right
3. **71 > 50**, go left

Insert 50 here
Why successor of $x$ can be used for deletion of a BST vertex $x$ with 2 children?

Claim: Successor of $x$ has at most 1 child!
   • Easier to delete and will not violate BST property

Proof:
   • Vertex $x$ has two children
   • Therefore, vertex $x$ must have a right child
   • Successor of $x$ must then be the minimum of the right subtree
   • A minimum element of a BST has no left child!!
   • *So, successor of $x$ has at most 1 child!* 😊
BST: Deletion Analysis

Delete a BST vertex \( v \), find \( v \) in \( O(h) \), then three cases:

- **Vertex \( v \) has no children:**
  - Just remove the corresponding BST vertex \( v \rightarrow O(1) \)

- **Vertex \( v \) has 1 child (either left or right):**
  - Connect \( v\).left (or \( v\).right) to \( v\).parent and vice versa \( \rightarrow O(1) \)
  - Then remove \( v \rightarrow O(1) \)

- **Vertex \( v \) has 2 children:**
  - Find \( x = \text{successor}(v) \rightarrow O(h) \)
  - Replace \( v\).key with \( x\).key \( \rightarrow O(1) \)
  - Then delete \( x \) in \( v\).right (otherwise we have duplicate) \( \rightarrow O(h) \)

**Running time:** \( O(h) \)
Now, after we learn BST...

<table>
<thead>
<tr>
<th>No</th>
<th>Operation</th>
<th>Unsorted Array</th>
<th>Sorted Array</th>
<th>BST</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Search(age)</td>
<td>$O(n)$</td>
<td>$O(\log n)$</td>
<td>$O(h)$</td>
</tr>
<tr>
<td>2</td>
<td>Insert(age)</td>
<td>$O(1)$</td>
<td>$O(n)$</td>
<td>$O(h)$</td>
</tr>
<tr>
<td>3</td>
<td>FindOldest()</td>
<td>$O(n)$</td>
<td>$O(1)$</td>
<td>$O(h)$</td>
</tr>
<tr>
<td>4</td>
<td>ListSortedAges()</td>
<td>$O(n \log n)$</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>5</td>
<td>NextOlder(age)</td>
<td>$O(n)$</td>
<td>$O(\log n)$</td>
<td>$O(h)$</td>
</tr>
<tr>
<td>6</td>
<td>Remove(age)</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>$O(h)$</td>
</tr>
<tr>
<td>7</td>
<td>GetMedian()</td>
<td>$O(n \log n)$</td>
<td>$O(1)$</td>
<td>$O(h)$</td>
</tr>
<tr>
<td>8</td>
<td>Rank(age)</td>
<td>$O(n \log n)$</td>
<td>$O(\log n)$</td>
<td>?</td>
</tr>
</tbody>
</table>

It is all now depends on ‘$h$’... → next lecture 😊
Worst case height of a BST

\[ h = O(n) \ldots 😞 \]

Can you spot one more worst case scenario using the same set of numbers?
Java Implementation

See BSTDemo.java (you can use this for PS2)

Concepts covered:
1. Java Object Oriented Programming (OOP) implementation of BST data structure
2. Java Error Handling: Throw & Catch Exception
The Baby Names Problem (PS2)

Given a list of male and female baby names suggestions (*from your parents, in-laws, friends, yourself, Internet, etc*), your task is to answer some queries (see the next slide).

*This problem is always* encountered by every parent with a new baby.

(Including the search for baby Joshua name, born on 16 July 2014)
(Note: Unlike this lecture with integer keys, the keys in PS1 are strings)

**Easy:** How many names start with a certain letter?

**Medium:** How many names start with a certain prefix?

*Definition:* A prefix of a string $T = T_0T_1...T_{n-1}$ with length $n$ is string $P = T_0T_1...T_m$ where $m < n$.

**Hard:** Can you do it without Java API library code?

**CS2010R:** How many names have a certain substring?

*Definition:* A substring of a string $T = T_0T_1...T_{n-1}$ with length $n$ is string $S = T_iT_{i+1}...T_{j-1}T_j$ where $0 \leq i \leq j < n$.

You need efficient DS(es) to answer those queries
End of Lecture Quiz 😊

After Lecture 03, I will set a random test mode @ VisuAlgo to see if you understand BST

Go to:
http://visualgo.net/test.html

Use your CS2010 account to try the 5 BST questions (medium difficulty, 5 minutes)

Meanwhile, train first 😊
http://visualgo.net/training.html