CS 225
Data Structures

October 13 – AVL Applications
G Carl Evans
Why Balanced BST?
Summary of Balanced BST

Pros:
- Running Time:
  - Improvement Over:

- Great for specific applications:
Every Data Structure So Far

<table>
<thead>
<tr>
<th></th>
<th>Unsorted Array</th>
<th>Sorted Array</th>
<th>Unsorted List</th>
<th>Sorted List</th>
<th>Binary Tree</th>
<th>BST</th>
<th>AVL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Find</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Insert</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Remove</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Traverse</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Summary of Balanced BST

Cons:
- Running Time:

- In-memory Requirement:
Iterators

Why do we care?

```cpp
1 DFS dfs(...);
2 for ( ImageTraversal::Iterator it = dfs.begin(); it != dfs.end(); ++it ) {
3   std::cout << (*it) << std::endl;
4 }
```
Iterators

Why do we care?

```cpp
1 DFS dfs(...);
2 for ( ImageTraversal::Iterator it = dfs.begin(); it != dfs.end(); ++it ) {
3   std::cout << (*it) << std::endl;
4 }
```

```cpp
1 DFS dfs(...);
2 for ( const Point & p : dfs ) {
3   std::cout << p << std::endl;
4 }
```
Iterators

Why do we care?

```cpp
DFS dfs(...);
for ( ImageTraversal::Iterator it = dfs.begin(); it != dfs.end(); ++it ) {
    std::cout << (*it) << std::endl;
}
```

```cpp
DFS dfs(...);
for ( const Point & p : dfs ) {
    std::cout << p << std::endl;
}
```

```cpp
ImageTraversal & traversal = /* ... */;
for ( const Point & p : traversal ) {
    std::cout << p << std::endl;
}
```
CS 225 Office Hours

Office Hours
• Must have online contact info

• Must have a specific question

• We will remove students that don’t do the above

• Purpose to get you unstuck not to fix your code
CS 225 Final Project

Working with data and using graphs
The Internet 2003
The OPTE Project (2003)
Map of the entire internet; nodes are routers; edges are connections.
Conflict-Free Final Exam Scheduling Graph

Unknown Source
Presented by Cinda Heeren, 2016
“Rush Hour” Solution
Unknown Source
Presented by Cinda Heeren, 2016
Class Hierarchy At University of Illinois Urbana-Champaign
A. Mori, W. Fagen-Ulmschneider, C. Heeren

Graph of every course at UIUC; nodes are courses, edges are prerequisites

http://waf.cs.illinois.edu/discovery/class_hierarchy_at_illinois/
MP Collaborations in CS 225

Unknown Source
Presented by Cinda Heeren, 2016
“Stanford Bunny”
Greg Turk and Mark Levoy (1994)
Final Project - Form a Team

• Team formation will be happening next week.

• If you don’t find a team we will match you up.

• You must fill out the form next week
Range-based Searches

**Q:** Consider points in 1D: \( p = \{p_1, p_2, ..., p_n\} \).
...what points fall in [11, 42]?

**Tree construction:**
Range-based Searches

Balanced BSTs are useful structures for range-based and nearest-neighbor searches.

Q: Consider points in 1D: \( p = \{p_1, p_2, ..., p_n\} \).
   ...what points fall in \([11, 42]\)?

Ex:

\[
\begin{array}{cccccccc}
3 & 6 & 11 & 33 & 41 & 44 & 55
\end{array}
\]
Range-based Searches

Q: Consider points in 1D: $p = \{ p_1, p_2, ..., p_n \}$.

...what points fall in $[11, 42]$?

Ex: 3 6 11 33 41 44 55
Range-based Searches

Q: Consider points in 1D: \( p = \{ p_1, p_2, \ldots, p_n \} \).

...what points fall in \([11, 42]\)?

Tree construction:
Range-based Searches
Range-based Searches
Range-based Searches

Q: Consider points in 1D: $p = \{p_1, p_2, \ldots, p_n\}$.
   ...what points fall in $[11, 42]$?
Range-based Searches
Running Time
Range-based Searches

Q: Consider points in 1D: $p = \{p_1, p_2, ..., p_n\}$.
   ...what points fall in $[11, 42]$?

Ex:

3 6 11 33 41 44 55
Range-based Searches

Consider points in 2D: \( p = \{ p_1, p_2, ..., p_n \} \).

**Q:** What points are in the rectangle: \([ (x_1, y_1), (x_2, y_2) ] \)?

**Q:** What is the nearest point to \((x_1, y_1)\)?
Range-based Searches

Consider points in 2D: \( p = \{p_1, p_2, \ldots, p_n\} \).

Tree construction:
Range-based Searches
kD-Trees
kD-Trees