COT 6405: Analysis of Algorithms

Giri NARASIMHAN

## More Dynamic Operations

<table>
<thead>
<tr>
<th></th>
<th>Search</th>
<th>Insert</th>
<th>Delete</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unsorted Arrays</strong></td>
<td>$O(N)$</td>
<td>$O(1)$</td>
<td>$O(N)$</td>
<td></td>
</tr>
<tr>
<td><strong>Sorted Arrays</strong></td>
<td>$O(\log N)$</td>
<td>$O(N)$</td>
<td>$O(N)$</td>
<td></td>
</tr>
<tr>
<td><strong>Unsorted Linked Lists</strong></td>
<td>$O(N)$</td>
<td>$O(1)$</td>
<td>$O(N)$</td>
<td></td>
</tr>
<tr>
<td><strong>Sorted Linked Lists</strong></td>
<td>$O(N)$</td>
<td>$O(N)$</td>
<td>$O(N)$</td>
<td></td>
</tr>
<tr>
<td><strong>Binary Search Trees</strong></td>
<td>$O(H)$</td>
<td>$O(H)$</td>
<td>$O(H)$</td>
<td>$H = O(N)$</td>
</tr>
<tr>
<td><strong>Balanced BSTs</strong></td>
<td>$O(\log N)$</td>
<td>$O(\log N)$</td>
<td>$O(\log N)$</td>
<td>As $H = O(\log N)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Se/In/De</th>
<th>Rank</th>
<th>Select</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Balanced BSTs</strong></td>
<td>$O(\log N)$</td>
<td>$O(N)$</td>
<td>$O(N)$</td>
<td></td>
</tr>
<tr>
<td><strong>Augmented BBSTs</strong></td>
<td>$O(\log N)$</td>
<td>$O(\log N)$</td>
<td>$O(\log N)$</td>
<td></td>
</tr>
</tbody>
</table>
Room Scheduling Problem

- Given a set of requests to use a room
  - \([0,6], [1,4], [2,13], [3,5], [3,8], [5,7], [5,9], [6,10], [8,11], [8,12], [12,14]\)
- Schedule largest number of above requests in the room
- Different approaches
  - Try by hand, exhaustive search, improve an initial solution, iterative methods, divide and conquer, greedy methods, etc.

- Simple Greedy Selection
  - Sort by start time and pick in “greedy” fashion
  - Does not work. WHY?
    - \([0,6], [6,10]\) is the solution you will end up with.

- Other greedy strategies
  - Sort by length of interval
  - Does not work. WHY?
Room Scheduling – Improved Solution

- [0, 6], [1, 4], [2, 13], [3, 5], [3, 8], [5, 7], [5, 9], [6, 10], [8, 11], [8, 12], [12, 14]
- [1, 4], [3, 5], [0, 6], [5, 7], [3, 8], [5, 9], [6, 10], [8, 11], [8, 12], [2, 13], [12, 14]

-- Sorted by finish times

- [1, 4], [3, 5], [0, 6], [5, 7], [3, 8], [5, 9], [6, 10], [8, 11], [8, 12], [2, 13], [12, 14]
- [1, 4], [3, 5], [0, 6], [5, 7], [3, 8], [5, 9], [6, 10], [8, 11], [8, 12], [2, 13], [12, 14]
- [1, 4], [3, 5], [0, 6], [5, 7], [3, 8], [5, 9], [6, 10], [8, 11], [8, 12], [2, 13], [12, 14]
- [1, 4], [3, 5], [0, 6], [5, 7], [3, 8], [5, 9], [6, 10], [8, 11], [8, 12], [2, 13], [12, 14]
- [1, 4], [3, 5], [0, 6], [5, 7], [3, 8], [5, 9], [6, 10], [8, 11], [8, 12], [2, 13], [12, 14]
Greedy Algorithms

- Given a set of activities \((s_i, f_i)\), we want to schedule the maximum number of non-overlapping activities.

**GREEDY-ACTIVITY-SELECTOR** \((s, f)\)

1. \(n = \text{length}[s]\)
2. \(S = \{a_1\}\)
3. \(i = 1\)
4. for \(m = 2\) to \(n\) do
5. if \(s_m\) is not before \(f_i\) then
6. \(S = S \cup \{a_m\}\)
7. \(i = m\)
8. return \(S\)
Why does it work?

- **THEOREM**
  Let $A$ be a set of activities and let $a_1$ be the activity with the earliest finish time. Then activity $a_1$ is in some maximum-sized subset of non-overlapping activities.

- **PROOF**
  Let $S'$ be a solution that does not contain $a_1$. Let $a'_1$ be the activity with the earliest finish time in $S'$. Then replacing $a'_1$ by $a_1$ gives a solution $S$ of the same size.

  Why are we allowed to replace? Why is it of the same size?

  Then apply induction! How?
First choice was a good choice. Why?
  - Because it can be extended to an optimal soln.
If our first choice was a good choice, then?
  - Then we can recursively apply correctness to the remainder