Data Structures & Algorithms for Problem Solving (CS1.304)

Lecture # 17

Avinash Sharma

Center for Visual Information Technology (CVIT),

IIIT Hyderabad

Another Way Towards Balanced Trees

- Should we even try to achieve balance at all times?
- In some settings, a few elements are accessed more often.
  - Do not know which are those elements.
  - Any example systems for dictionary search ?
- Does it suffice to keep these elements closer?

More on Search Trees

- Notice that a successful search operation can stop as soon as the element is found.
- If the element is a leaf node, then search operation on that node takes the longest time.
- **A successive search to the same node still takes the same amount of time.**
- In some settings, a few elements are searched more often than the others.
  - should focus on optimizing these searches.

More on Search Trees

- One way to make future search operations on the same node is to bring that node (closer) to the root.
- This is what we will do. Called as splaying.
- The search tree using this technique is called as splay tree.

Splay Trees

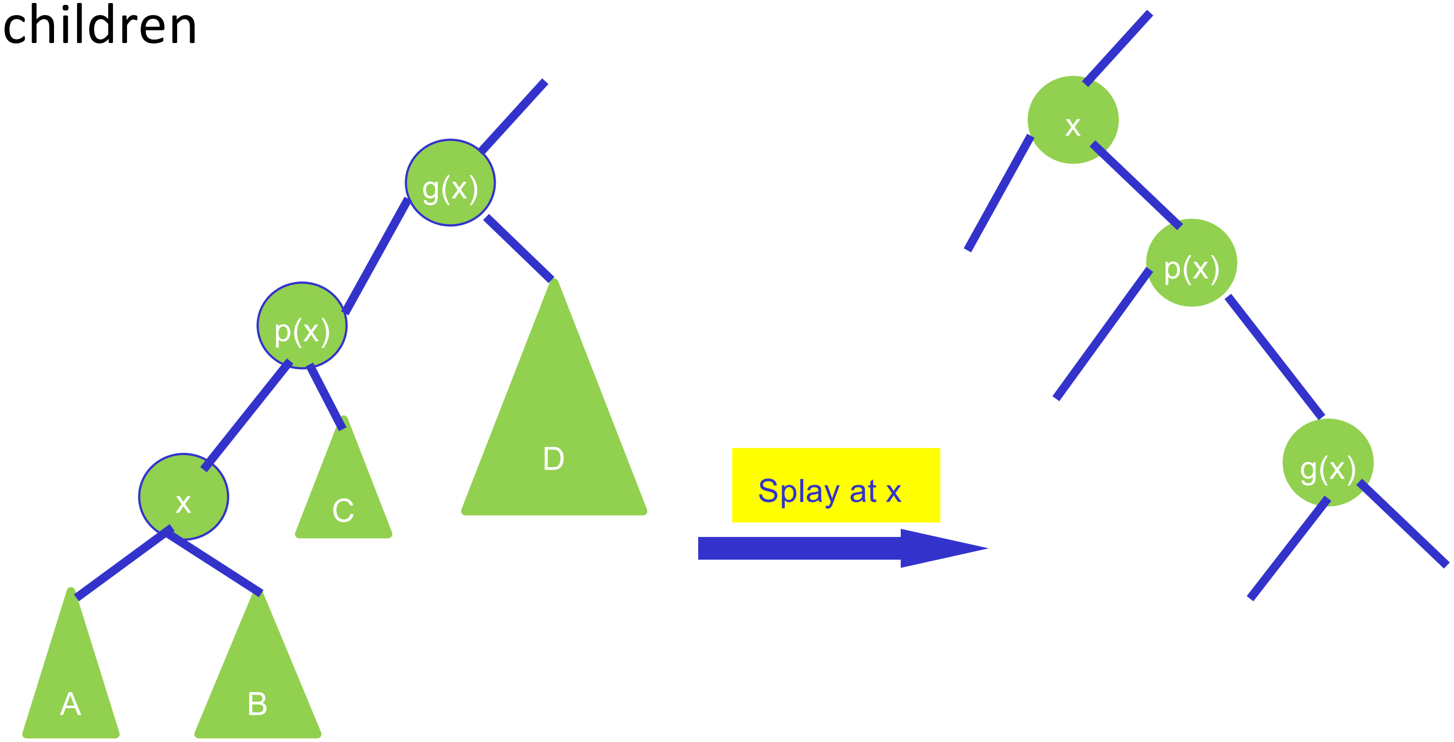
- In a splay tree, during every operation, including a search(), the current (search) tree is modified.
- The item searched is made as the root of the tree.
- During this process, other nodes also change their height.

Splay Trees Operation

- Let x be a node in the search tree.
- To make x as the root, we use operations similar to that of rotations.
- To splay a tree at node x, repeat the following splaying step until x is the root of the tree.
  - Let p(x) denotes the parent node of x and g(x) denotes the parent node of p(x) i.e., grand-parent of x.
  - The following cases are used depending on whether x is a left child of p(x), etc.

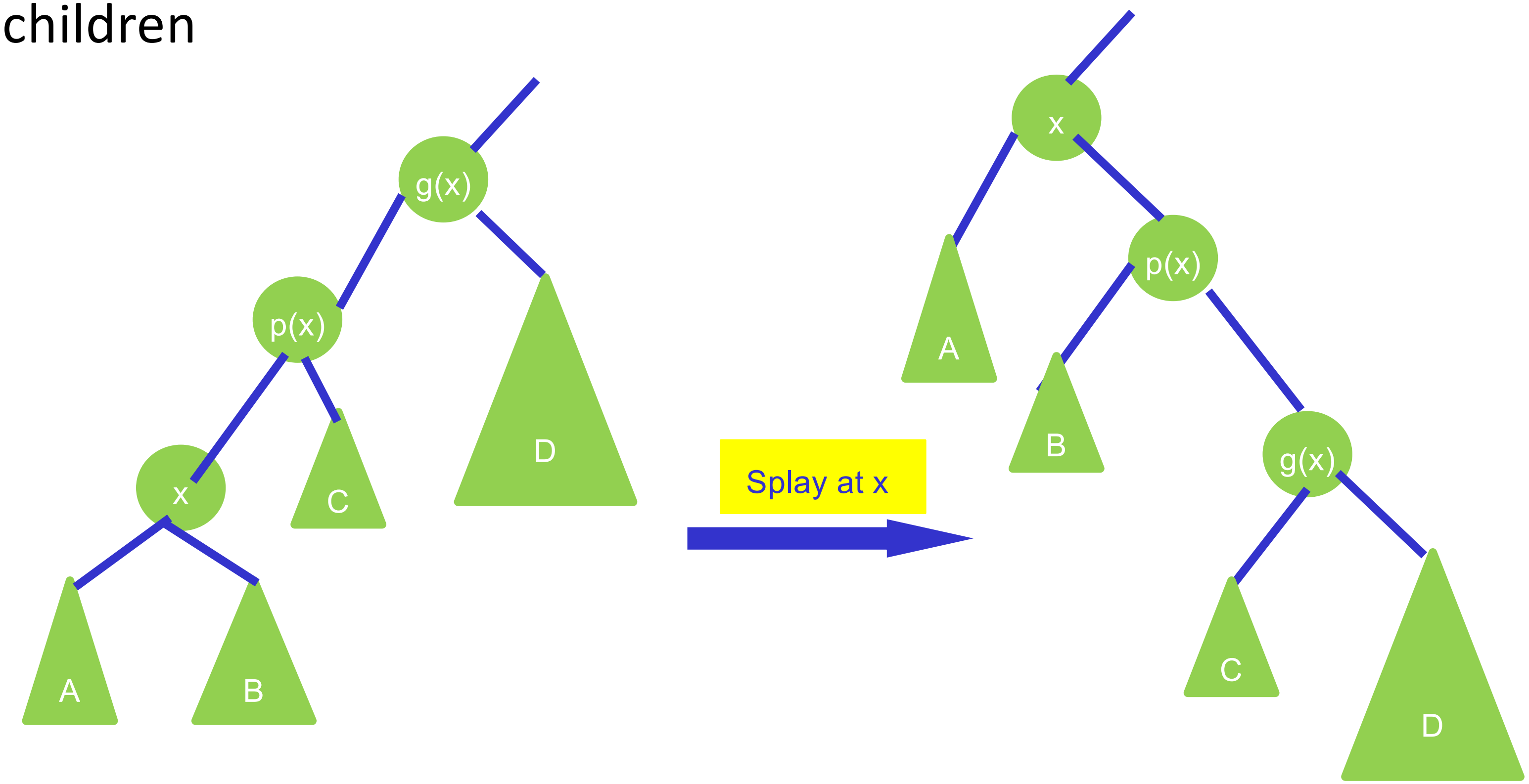
Four Cases

- Case Zig – Zig : If p(x) is not the root, and x and p(x) are both left (right) children



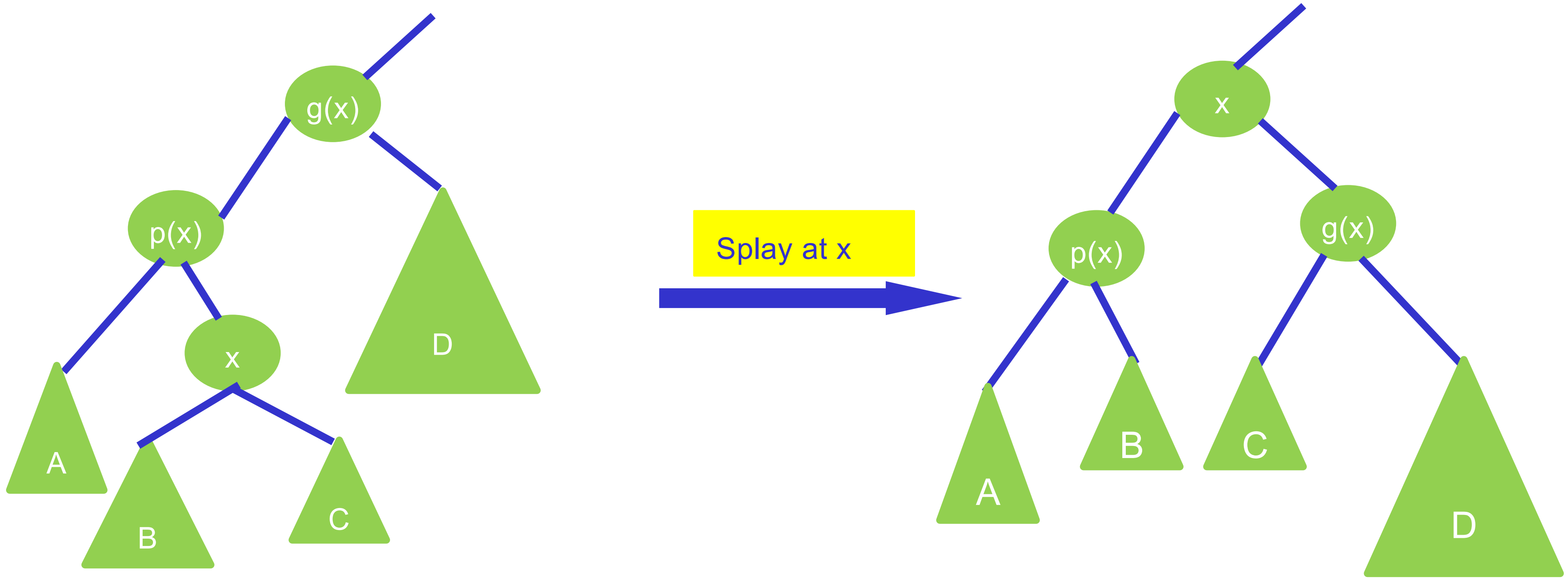
Four Cases

- Case Zig – Zig : If p(x) is not the root, and x and p(x) are both left (right) children



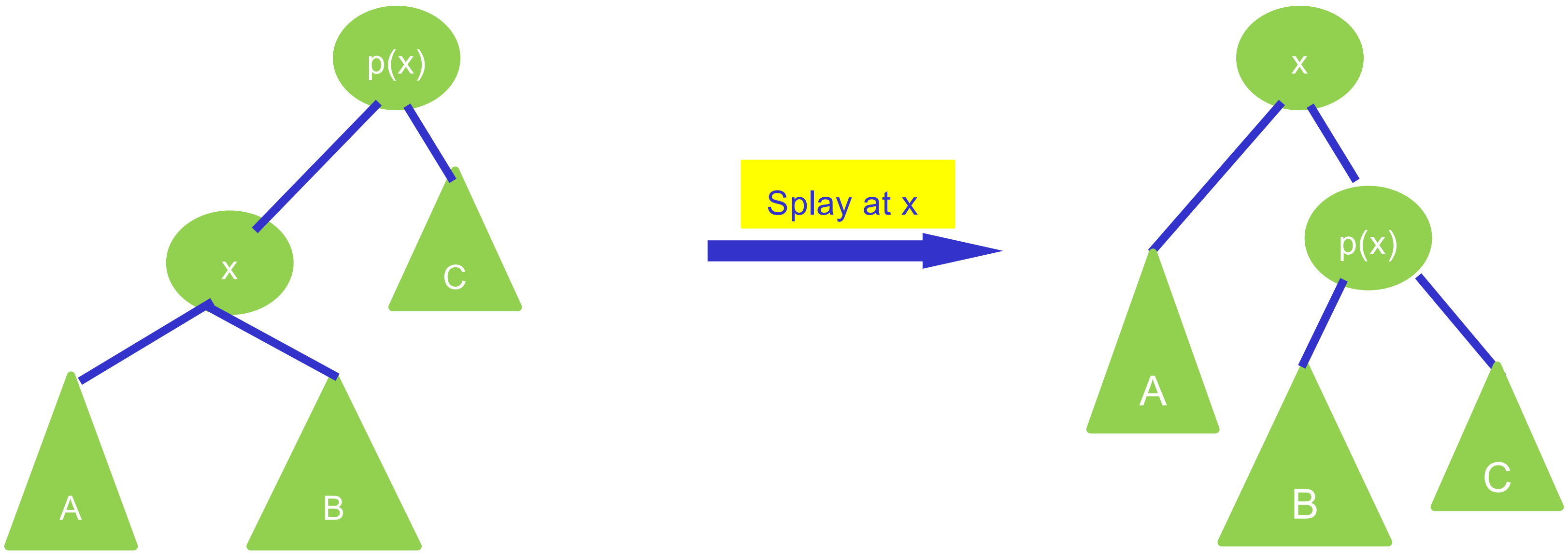
Four Cases

- Case Zig – Zag - If p(x) is not the root, and x is right (left) child and p(x) is left (right) child.



Two More Cases

- What if p(x) is the root? g(x) is not defined.
- If x is the left child of p(x), proceed as follows.



- The other case is easy to figure out.

Search(x) in a Splay Tree

- Proceed as search in a binary search tree.
- Once x is found, spaly(x) till x is the root.
- Splay uses the above cases.

Insert(x) & Delete(x)

Insert

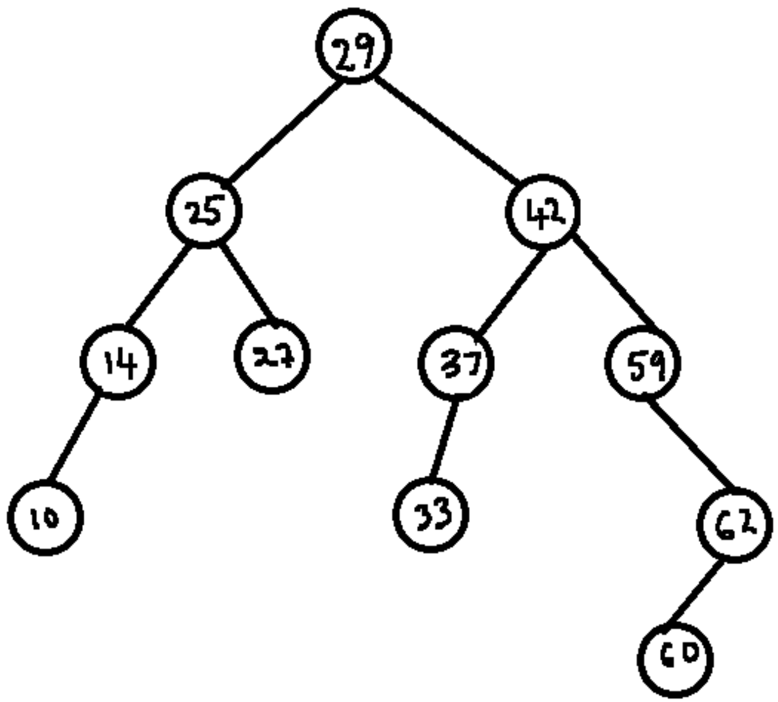
- Make x the root after inserting as in a binary search tree.

Delete

- Delete x as in a binary search tree.
- If y is the node physically deleted, then make the parent of y, p(y), as the root., i.e., splay(p(y))
- This is a bit artificial, but required for analysis to go through.

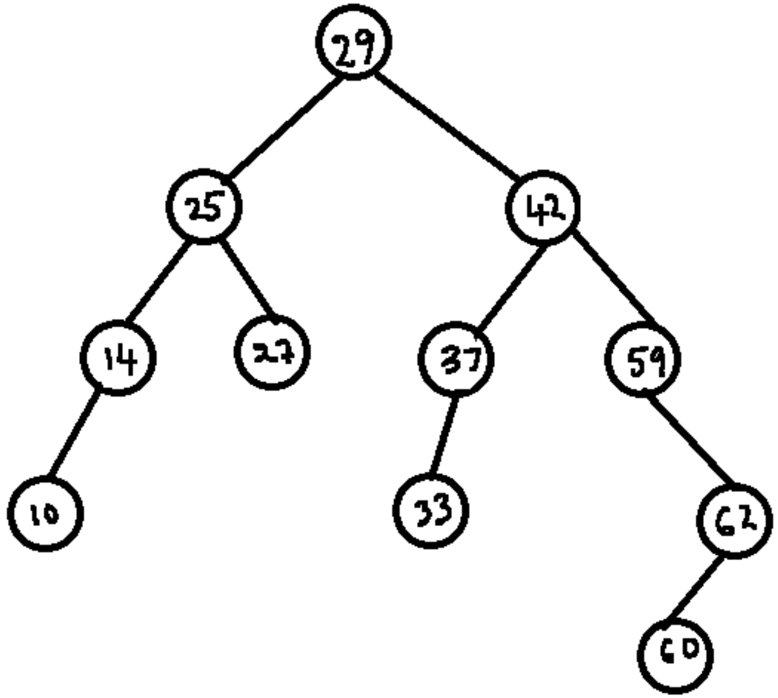
Practice Problem on Splay Tree

- Consider the following splay tree. Splay the nodes 37, then 14 in the resulting tree, and then 29 in the resulting tree.



Practice Problem on Splay Tree

- Consider the following splay tree. Splay the nodes 37, then 14 in the resulting tree, and then 29 in the resulting tree.



Analysis

- Analyzing the splay tree is a bit tough at this stage.
- Here are a few results:
- Any sequence of m operations on a splay tree can be completed in time $O((m+n) \log n)$.
- Topic for advanced classes.

Thank You