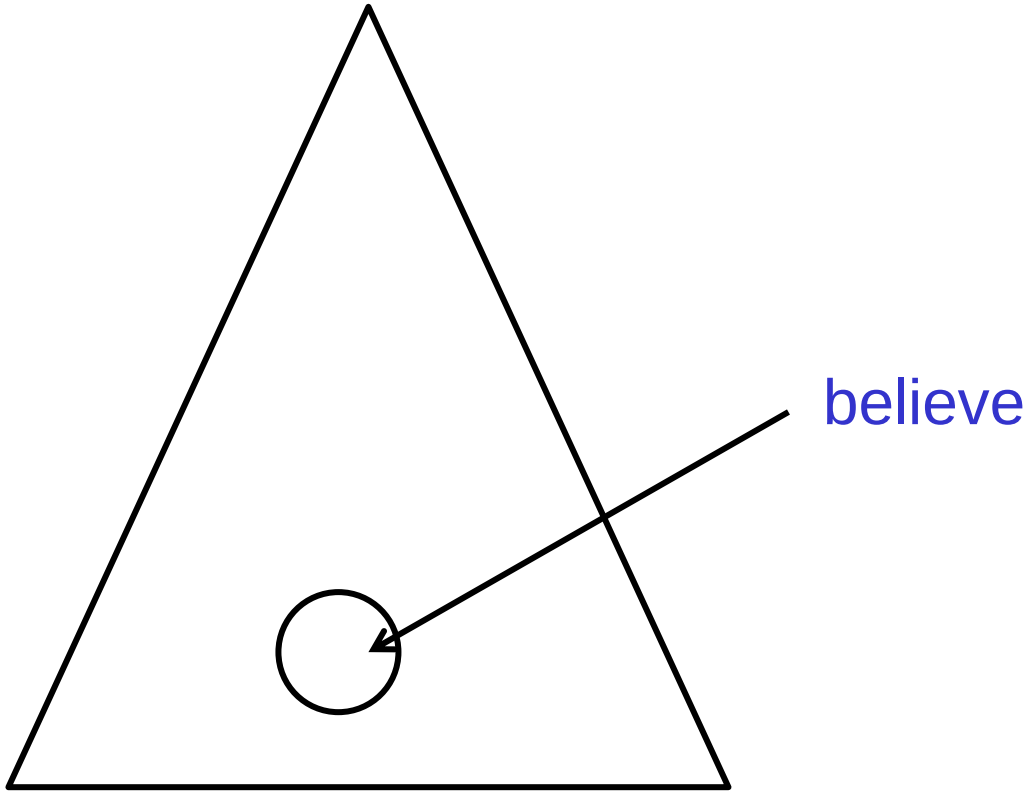
More on Search Trees

- AVL trees do have a O(log n) height in all situations.
- So, each operation takes O(log n) time in the worst case.
- So, better solution than hash tables.
  - Plus works for operations such as findMin() and findMax()
- Further optimizations as follows.

Application to Spell Checker



- Consider creating an AVL tree of all words in the English language.
- Can then use this tree for:
  - Verifying the spelling of a word, and
  - Suggesting alternatives

Application to a Spell Checker

- Total number of words in English is roughly 250,000.
- Using an AVL tree, the height of the tree would be about 28.
- Several words are more often mis-spelt.
  - Believe, accommodate, …
- These words are therefore searched more often.

More on Search Trees

- Notice that a (successful) search operation can stop as soon as the element is found.

More on Search Trees

- Notice that a (successful) search operation can stop as soon as the element is found.
- If the element being searched is a leaf node, then search operation on that node takes the longest time.

More on Search Trees

- Notice that a (successful) search operation can stop as soon as the element is found.
- If the element being searched is a leaf node, then search operation on that node takes the longest time.
- A successive search to the same element still takes the same amount of time.

More on Search Trees

- Notice that a (successful) search operation can stop as soon as the element is found.
- If the element being searched is a leaf node, then search operation on that node takes the longest time.
- A successive search to the same element 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 a splay tree.

Splay Trees

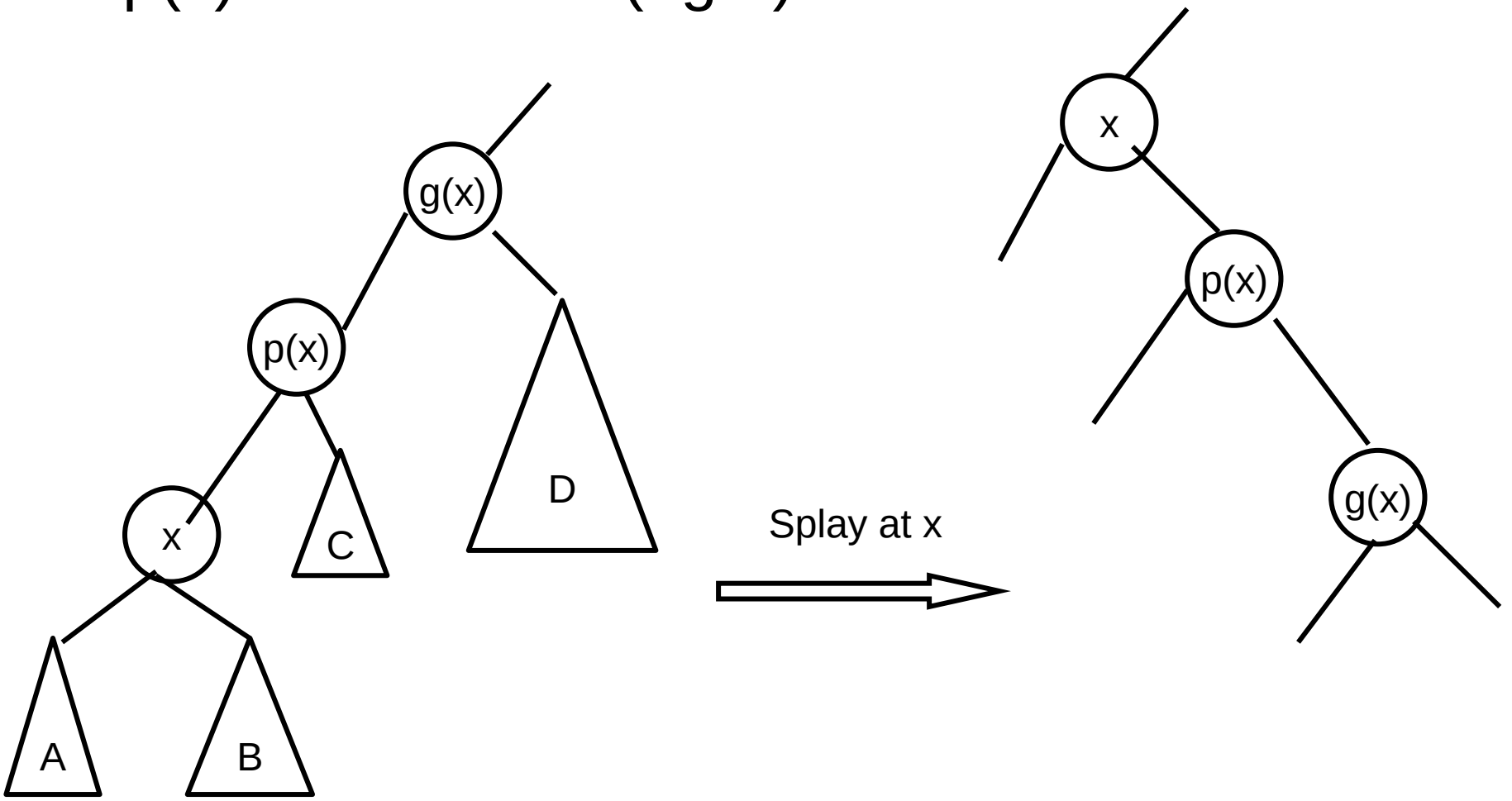
- In a splay tree, during every operation, including a search(), the current (search) tree is modified.
- The item searched is made as the root of the tree.
- During this process, other nodes also change their height.

The Splay 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) denote the parent node of x.
  - The following cases are used depending on whether x is a left child of p(x), etc.

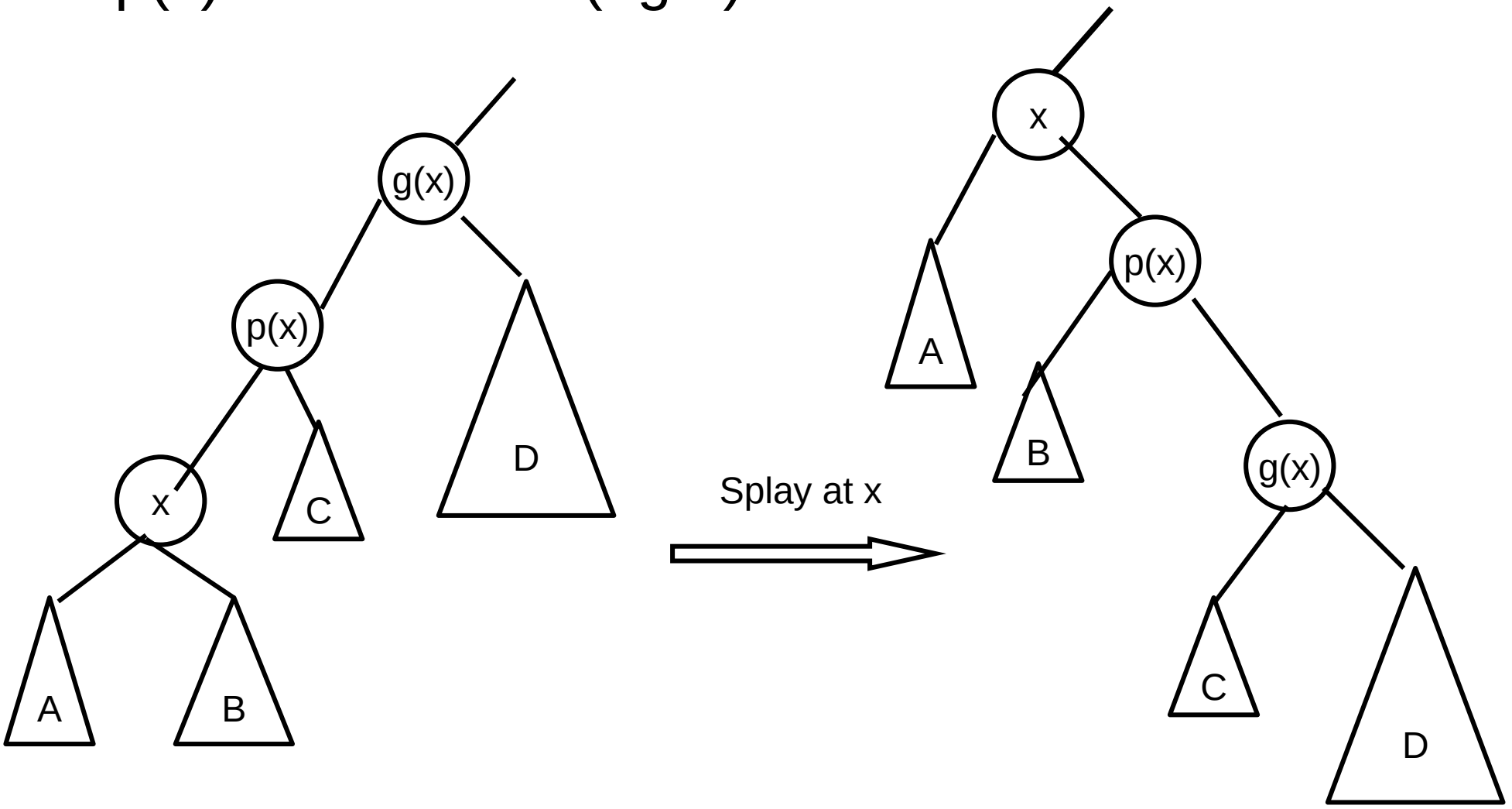
Four Cases

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



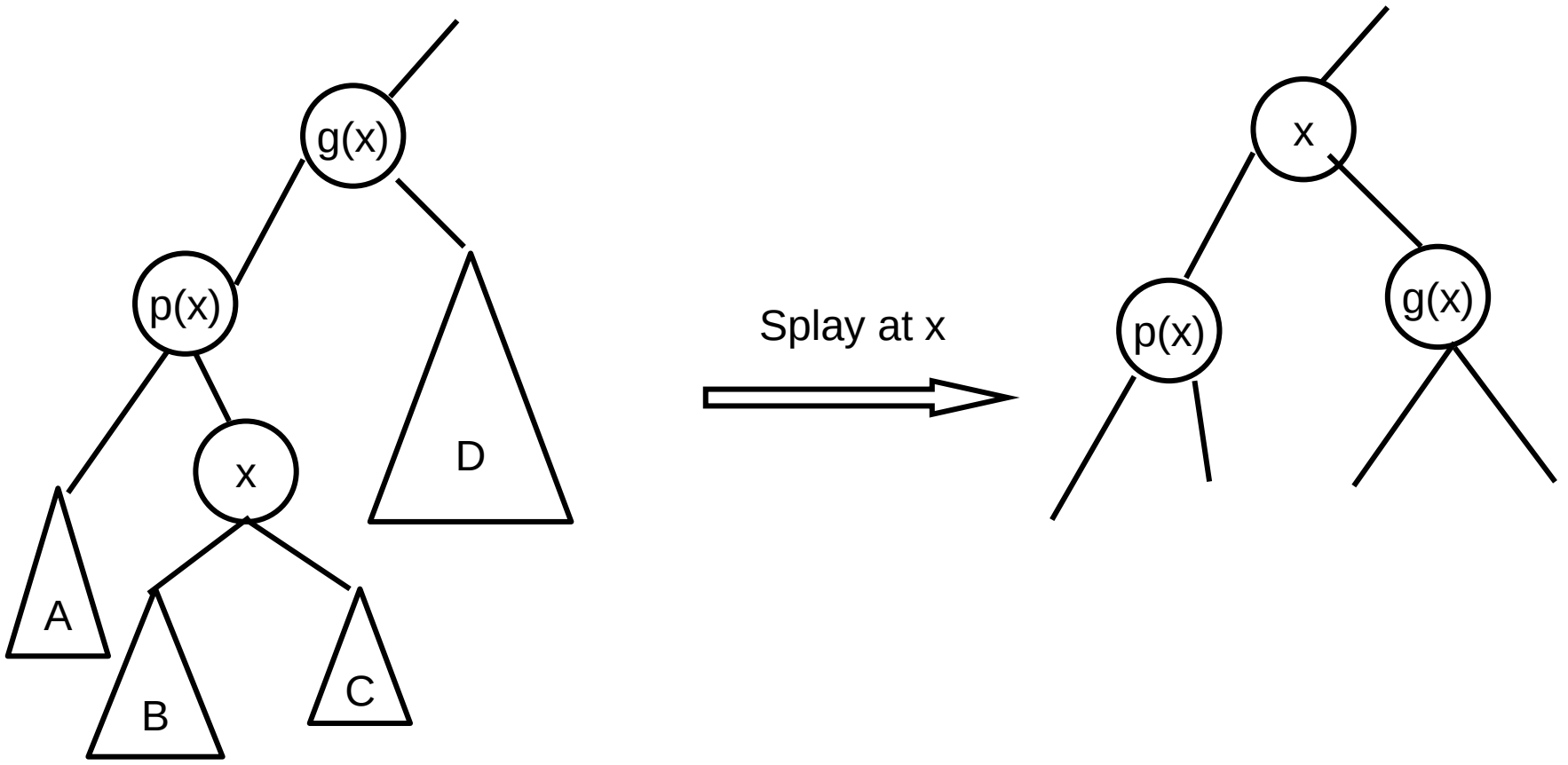
Four Cases

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



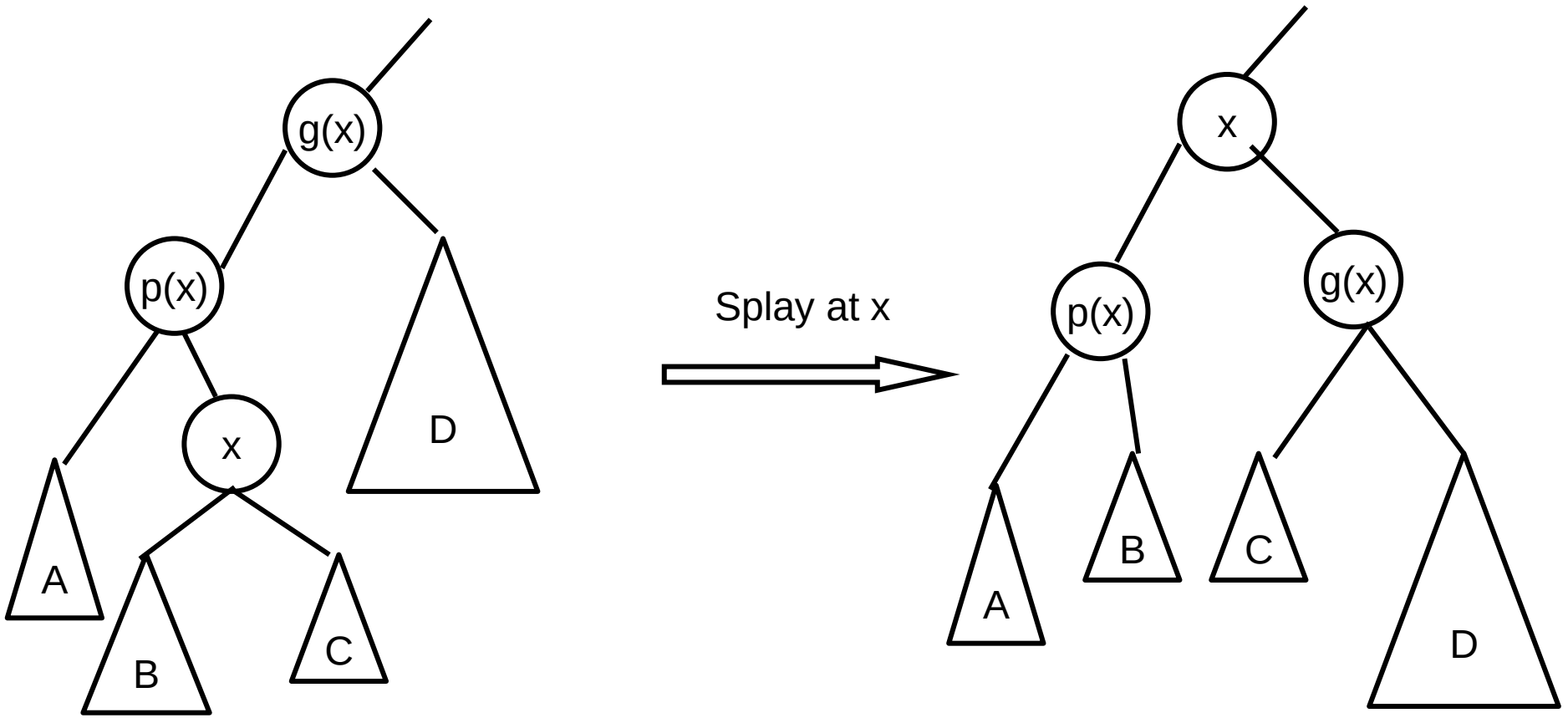
Four Cases

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



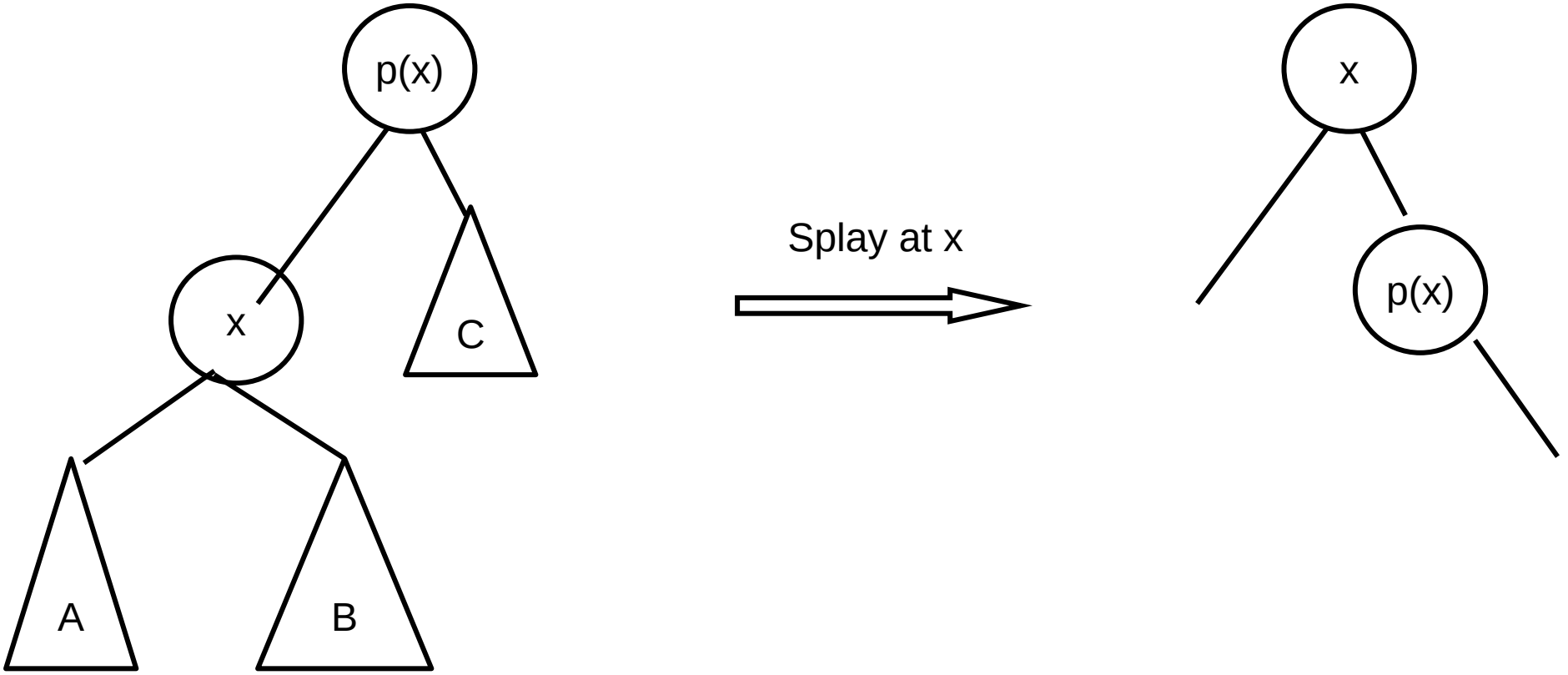
Four Cases

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



Two More Cases

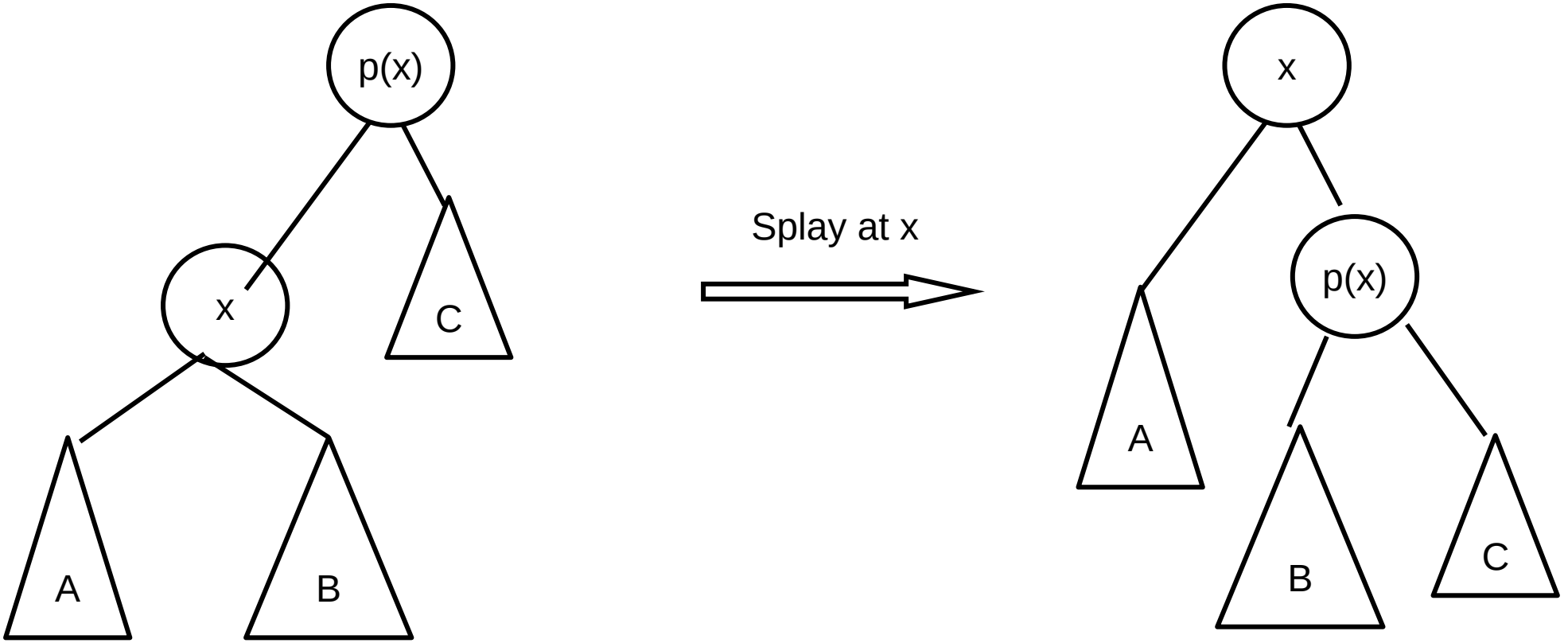
- What if p(x) is the root? g(x) is not defined.
- If x is the left child of p(x), proceed as follows.



- The other case is easy to figure out.

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)

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

Delete(x)

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

Practice Problem

- Show the resulting splay tree obtained by the following operations:

Insert(6), Insert(10), Insert(12), search(10), insert(3), search(12), insert(9), search(6), and search(12)

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).
- Other claims such as working set claims, also hold.
- Topic for advanced classes.

Splay Tree vs. AVL Trees

- AVL trees come with strict height invariant, and are sometimes difficult to program, and could be a bit slower too.
- Splay trees on the other hand are very simple.

- Splay trees are called self-adjusting data structures.
  - Self adjusting refers to the property that the data structure changes according to the application.