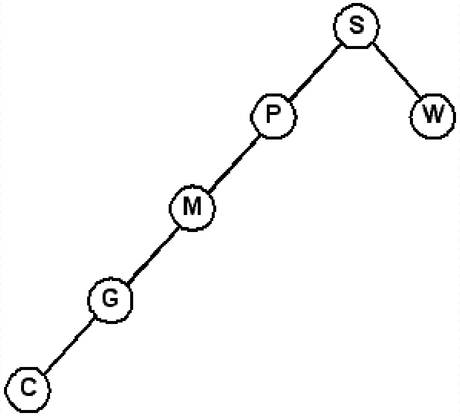
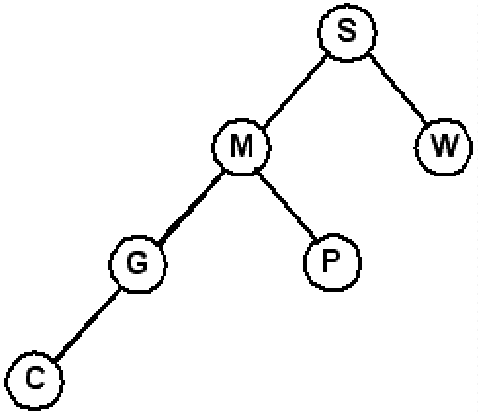
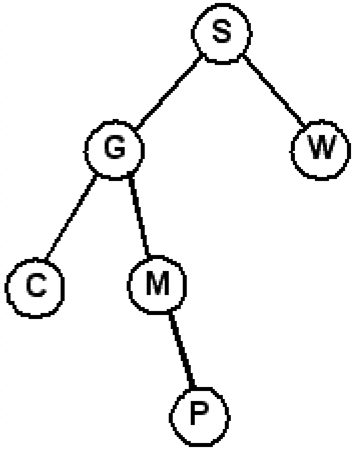
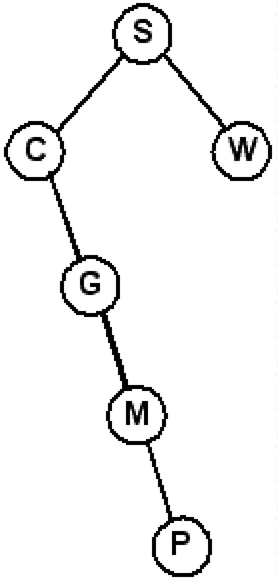
Splay Trees

Observations









- The trees above all contain the same data
  - Why are there four different representations of the data?

Observations

- Most of the time, we don't make any assumptions about the data.
- Usually assume equal distribution of data and random values.
- Non-random data can lead to worst case situations
  - Think of building a BST from already sorted data

Splay Trees

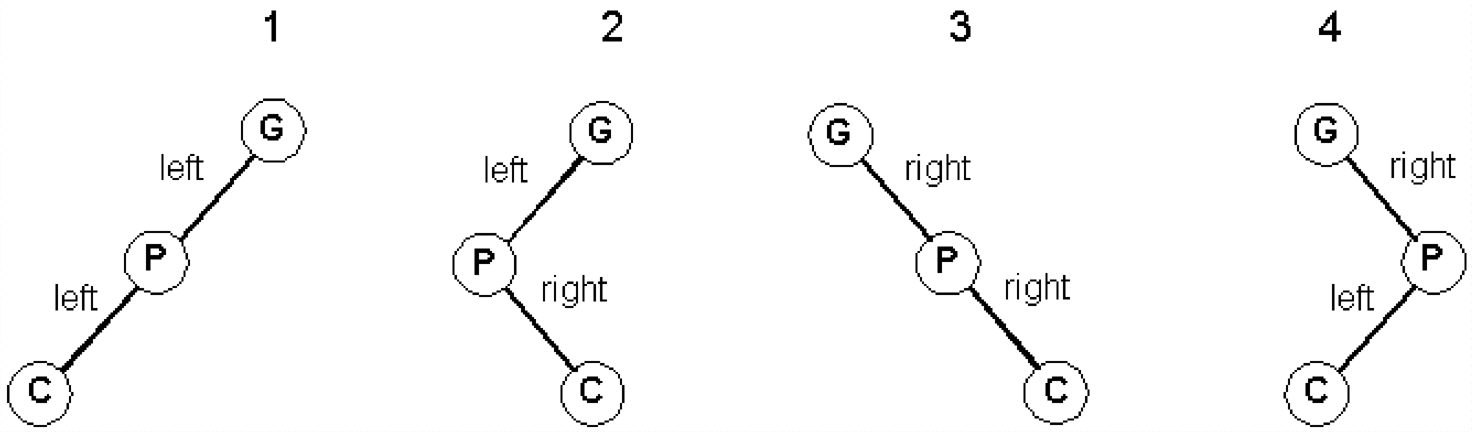
- A splay tree uses this knowledge to an advantage
- Newly inserted items are propagated to the root
  - Here "propagated" = "promoted"
- This propagation occurs when *writing* and *reading* an item (i.e. insert and access).
- We call this propagation of a node ***splaying***.

The basic idea is that frequently accessed data is always near the top

Splaying Algorithm

- We want to splay a node two levels at a time.
- This means we want to promote the node to the position of its grandparent (parent’s parent)
- The algorithm depends on the node’s orientation to its grandparent
- This leads to 4 possible orientations

Splaying Algorithm

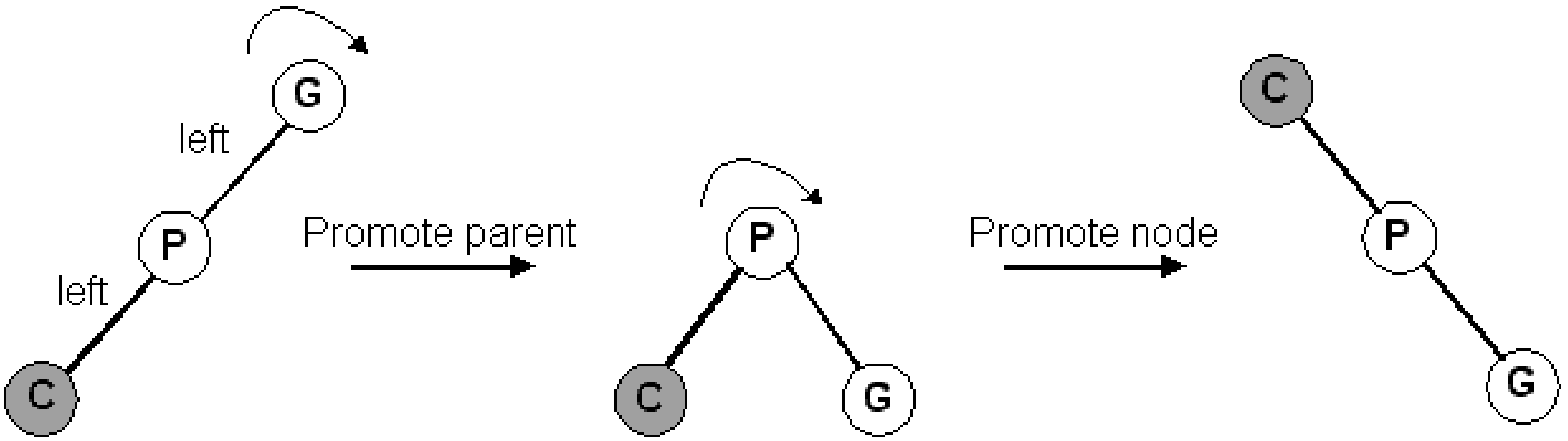


1. left-left, promote the parent, promote the node
2. left-right, promote the node, promote the node (node is promoted twice)
3. right-right, promote the parent, promote the node
4. right-left, promote the node, promote the node (node is promoted twice)

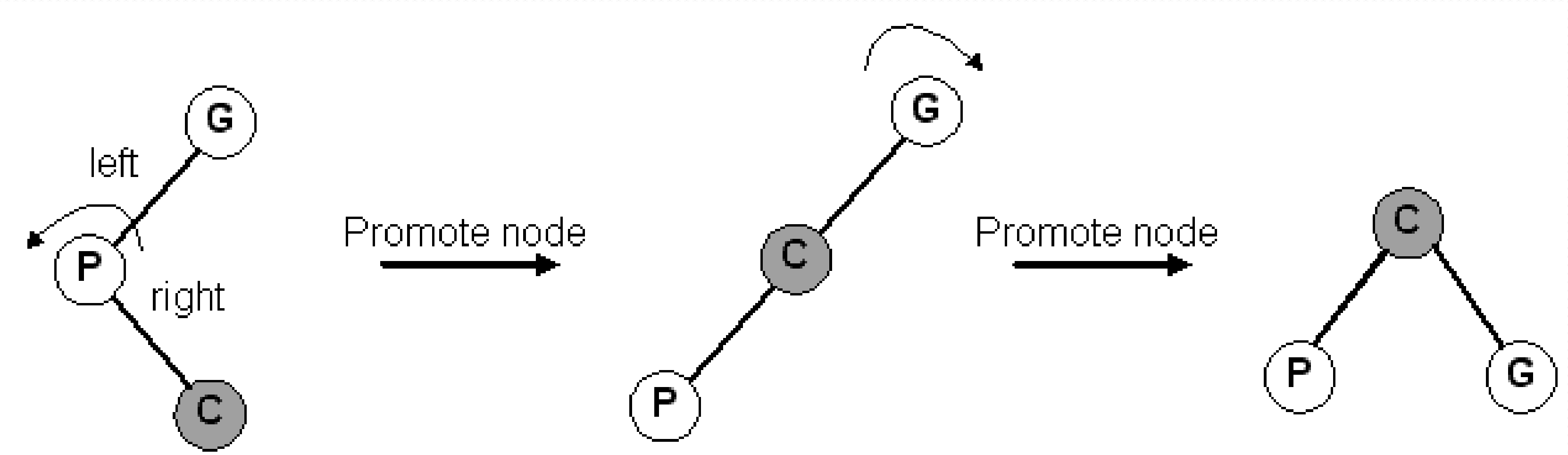
Splaying Algorithm

- Remember that promoting a node simply means rotating about the node’s parent.
  - Promoting doesn’t require to specify left or right. The direction is implied.
    - If a node is a right-child rotate parent LEFT.
    - If a node is a left-child rotate parent RIGHT.

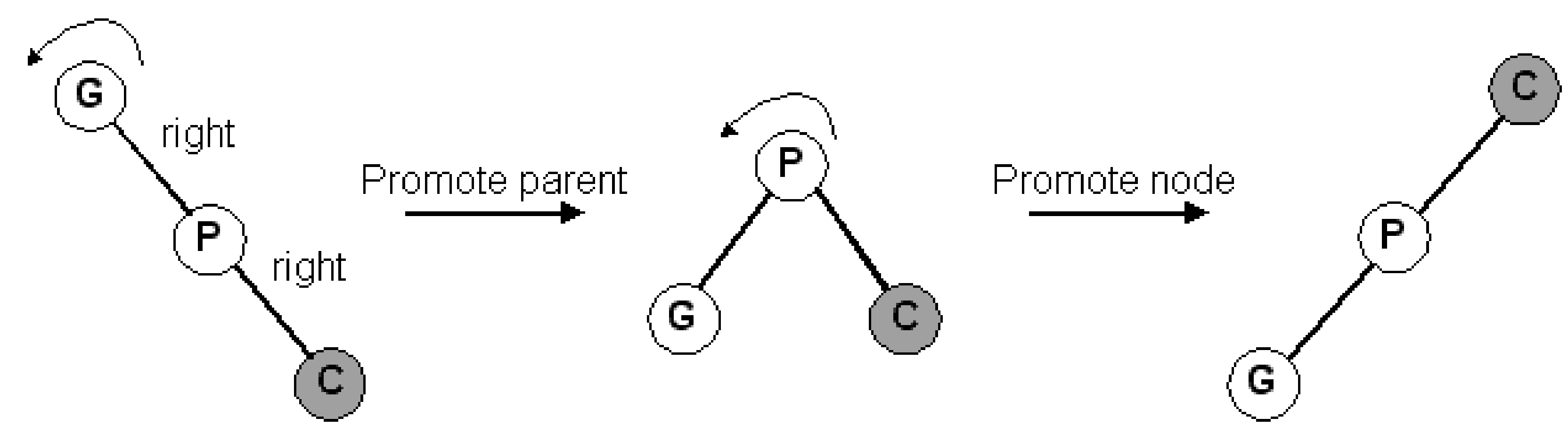
LEFT-LEFT orientation (ZIG-ZIG)



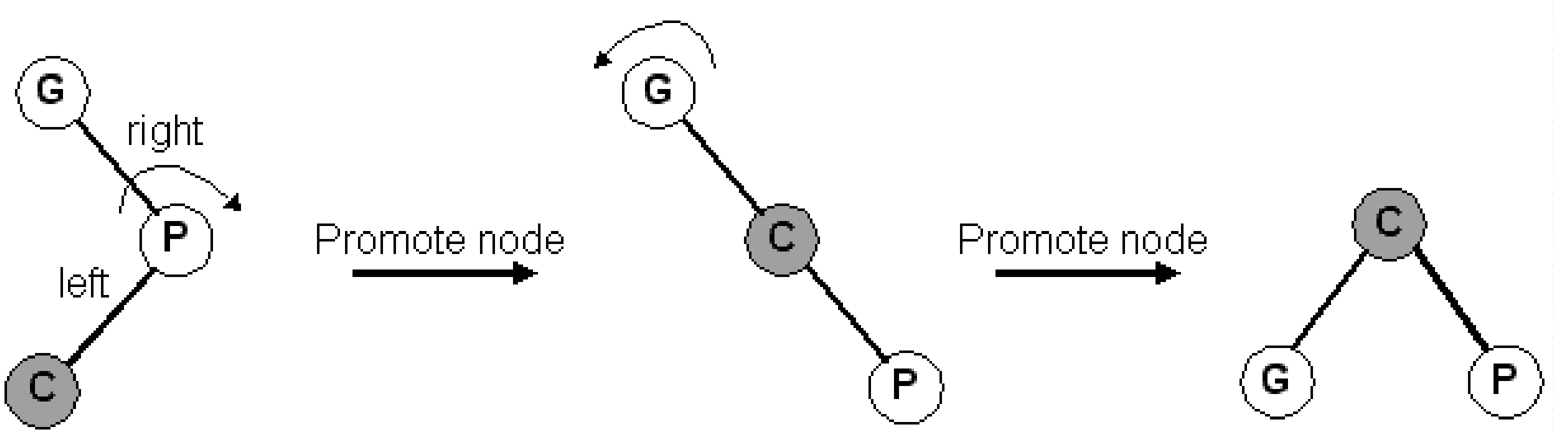
LEFT-RIGHT orientation (ZIG-ZAG)



RIGHT-RIGHT orientation (ZIG-ZIG)

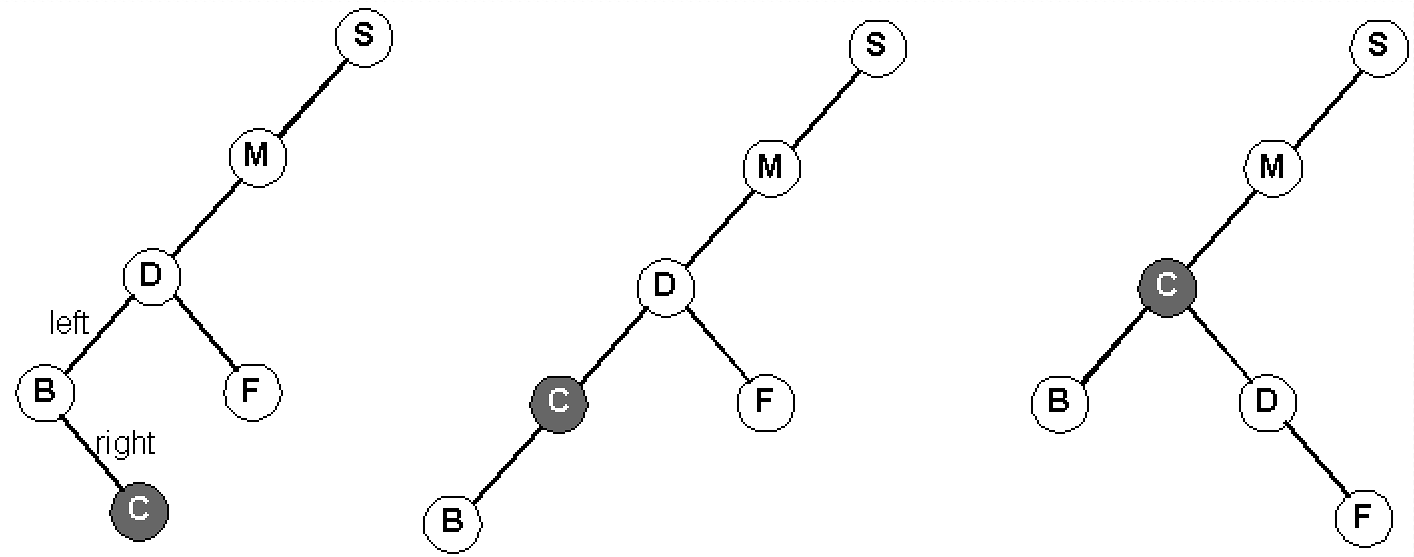


RIGHT-LEFT orientation (ZIG-ZAG)



Splaying Algorithm

- We continue to promote until we reach the root
- The “special case” is if our parent is the root
  - Simply perform a rotation to bring the node to the root

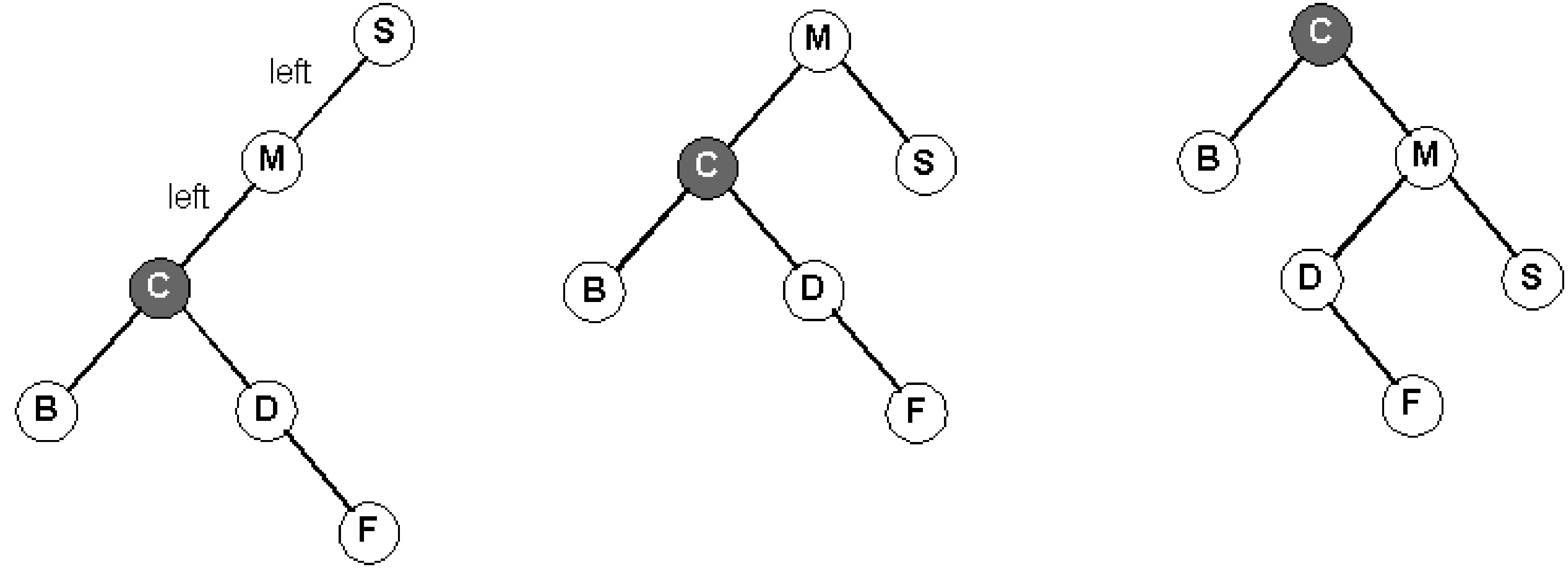


Orientation is left-right

Promote node

Promote node

Splaying Algorithm: Example



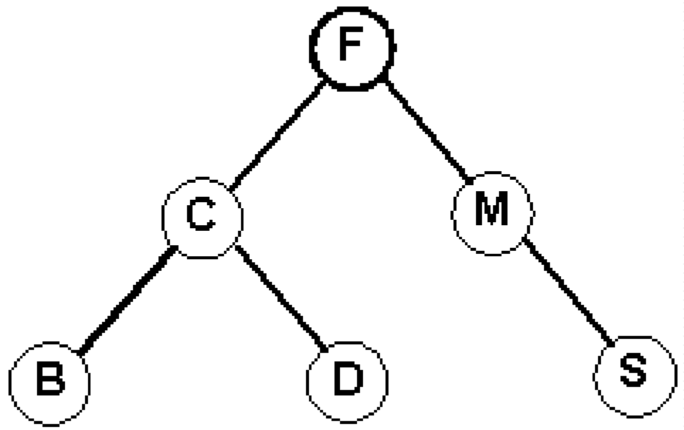
Orientation is left-left

Promote parent

Promote node

Exercise:

- What would be the result of splaying **F** to the root, assuming that we are using the previous tree after splaying **C**



Considerations

- Splay trees are not guaranteed to be balanced.
- Worst-case is not guaranteed to be “good”
- Average time may be excellent(this may be more important)
- Algorithms for splaying a node are simple.
    - Variation of the more general BST insertion
- Acts sort of like a built-in caching mechanism
- Works well with non-uniform access patterns over a “small” working set.

Animations

- http://www.ibr.cs.tu-bs.de/courses/ss98/audii/applets/BST/SplayTree-Example.html
- http://techunix.technion.ac.il/~itai/