Binary Search Trees: Advanced Topics

Semester 2, 2022

Lars Kulik

Binary search trees

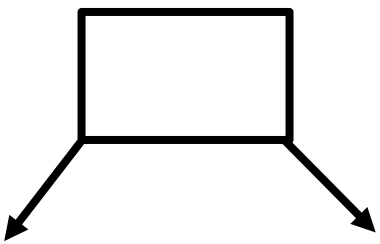
- Binary search tree: potentially O(log n) search, but basic insert method is not guaranteed to give a balanced tree
- AVL tree adds rotation steps to balance tree
- Many other ways to design search trees!

2-3-4 Trees

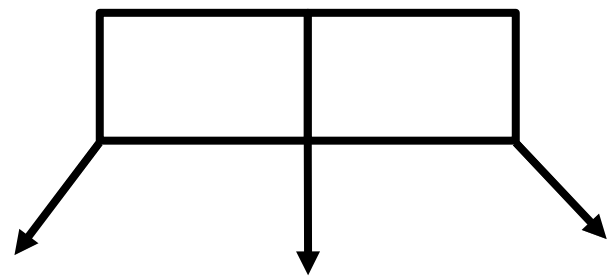
- Trees don’t have to be binary!
- Nodes in **2-3-4 trees** have:
  - 1, 2, or 3 keys
  - 2, 3, or 4 pointers, correspondingly
- More branches per node = fewer layers of nodes and lower tree height
- Also called **B-trees** of **order 4**

2-3-4 tree nodes

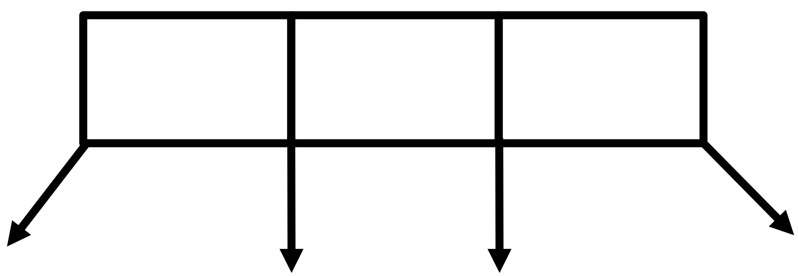
2-node



3-node



4-node



2-3-4 trees

- How to build the tree
  - Items are inserted only into leaf nodes
  - Insert up to 3 items in each node
- If a node is full (3 items) and you try to insert another item:
  - The middle item is promoted to become a parent of that node
  - The node splits to accommodate new item
- **Self-balancing** tree: promotion/split method ensures tree stays balanced

Example: 2-3-4 trees

Try it:
https://www.cs.usfca.edu/~galles/visualization/BTree.html

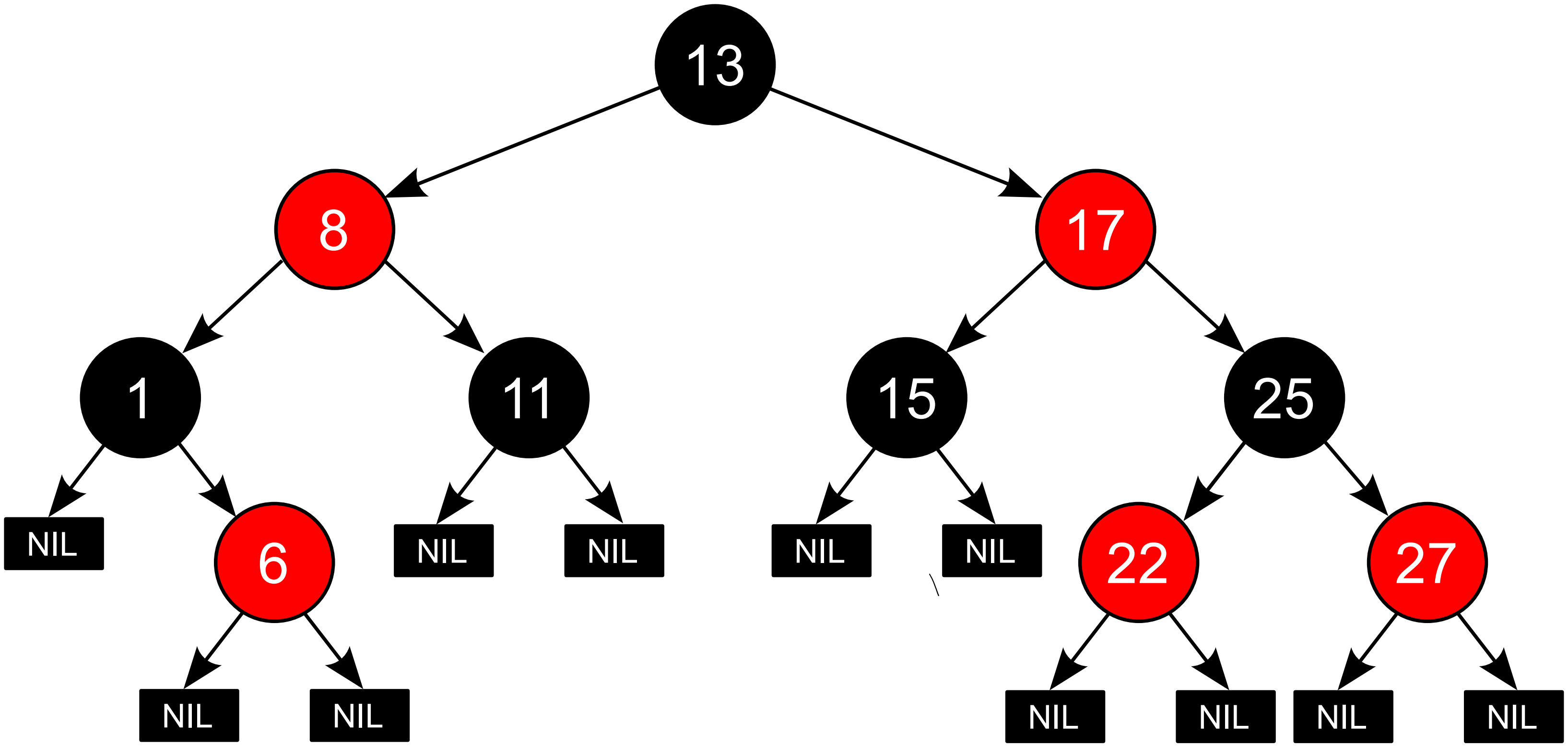
B and B+ trees

- B trees can be much higher order:
  - E.g., 256 or 512 pointers per node
  - Height of tree is lower than binary tree: e.g., $\log_{256}(n)$ or $\log_{512}(n)$
  - (Does that change the time complexity of search?)
- **B+ trees:** B tree variant:
  - Data is stored only in leaf nodes; non-leaf nodes only contain keys
  - Leaf nodes may include pointer to next leaf, to speed up sequential access
- Used for storing large databases on disk, where accesses are very expensive

Example: B+ trees

Try it:
https://www.cs.usfca.edu/~galles/visualization/BPlusTree.html

Red-black trees



src: https://upload.wikimedia.org/wikipedia/commons/6/66/Red-black_tree_example.svg

Red-black trees

- Red-black trees implement a 2-3-4 tree as a binary search tree, using rotation to keep balance
- Beyond the scope of this subject, but for details, see:
  - Sedgewick, *Algorithms in C, Parts 1-4*, Section 13.4
- Used in many BST applications, such as for job scheduling in Linux kernel:
  - https://www.kernel.org/doc/Documentation/rbtree.txt

Splay trees

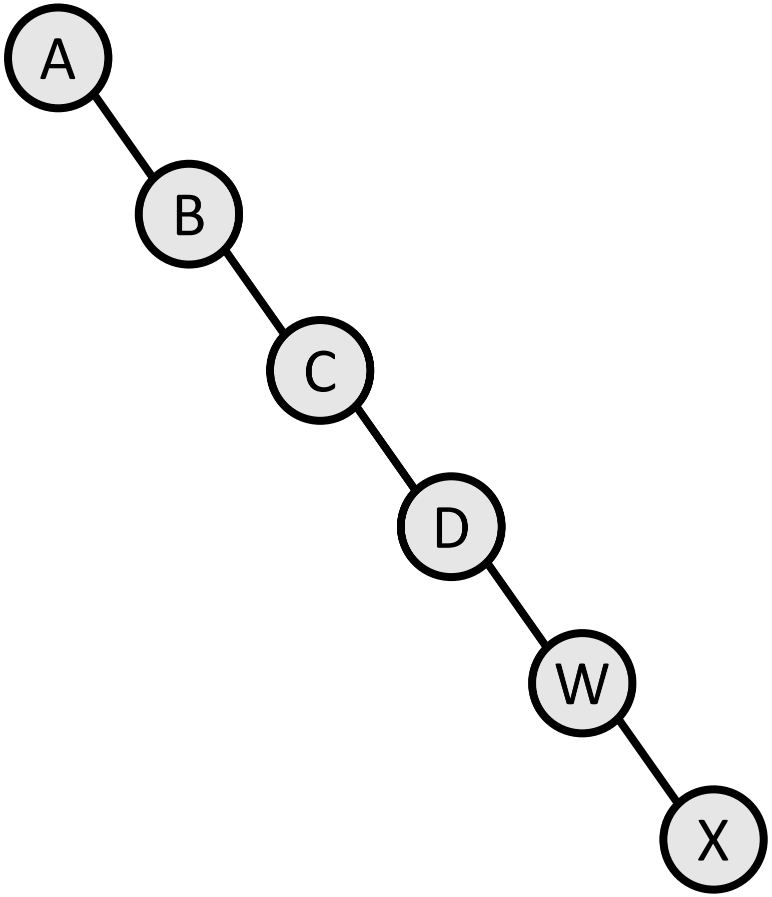
- A splay tree is a self-adjusting tree
- Insertion:
  - Insert as for BST
  - “Splay” new node to the root
- Search:
  - Search as for BST
  - “Splay” the searched node to the root
- **Splay** = do a series of rotations that bring the node closer to the root

Splay trees

- What does the splay operation accomplish?
- Spreads out nodes
  - First search might be O(n) in a long “stick” tree
  - But splaying reorders nodes so “stick” becomes more balanced; later searches should be faster
- Frequently-accessed nodes are moved closer to root
  - Good solution for data with **non-uniform access**

Sleator and Tarjan, *Self-Adjusting Binary Search Trees*, *JACM* **32(3)**, 1985, 652-686.

Example: Splay tree

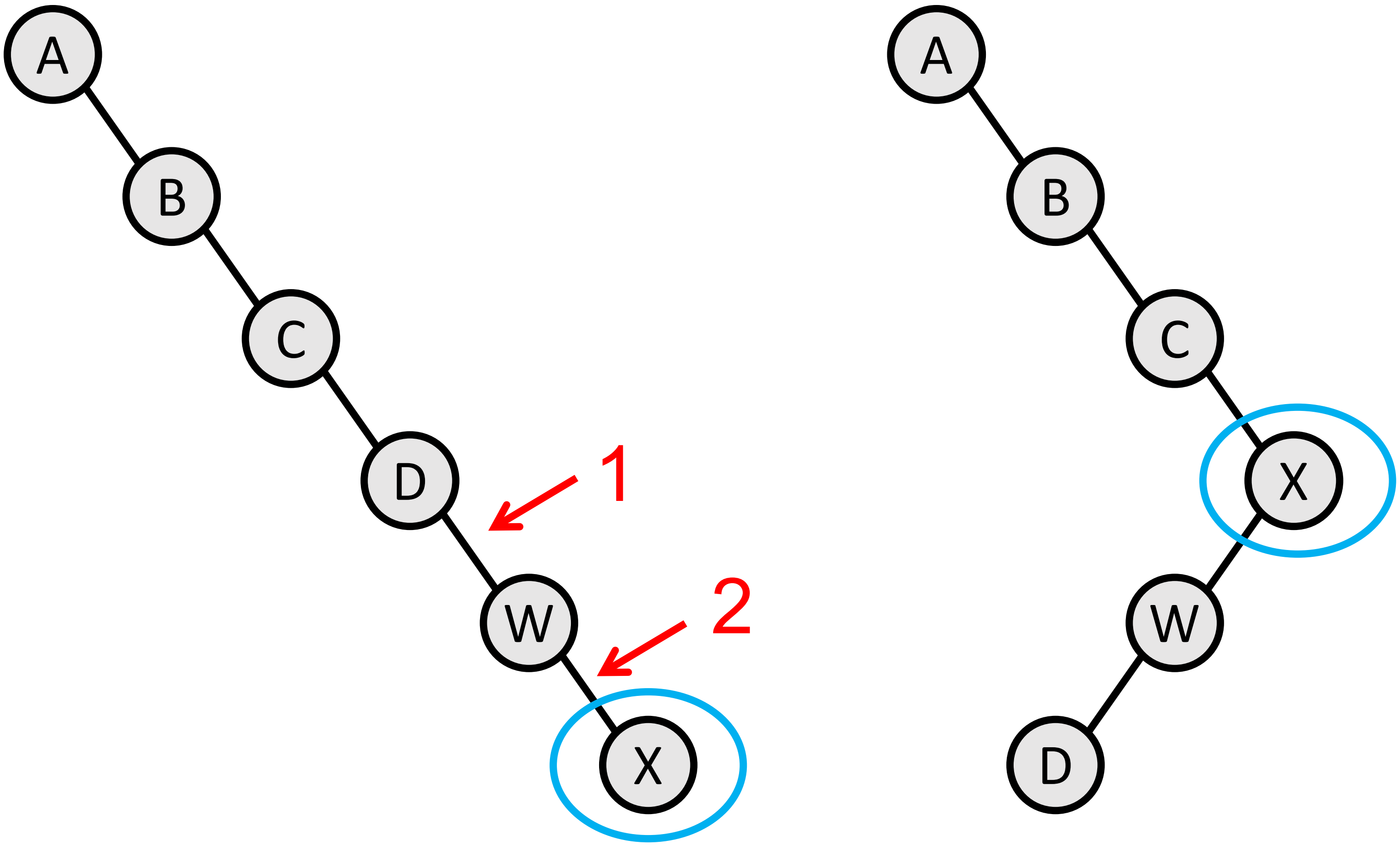


Shape: a stick, height 5

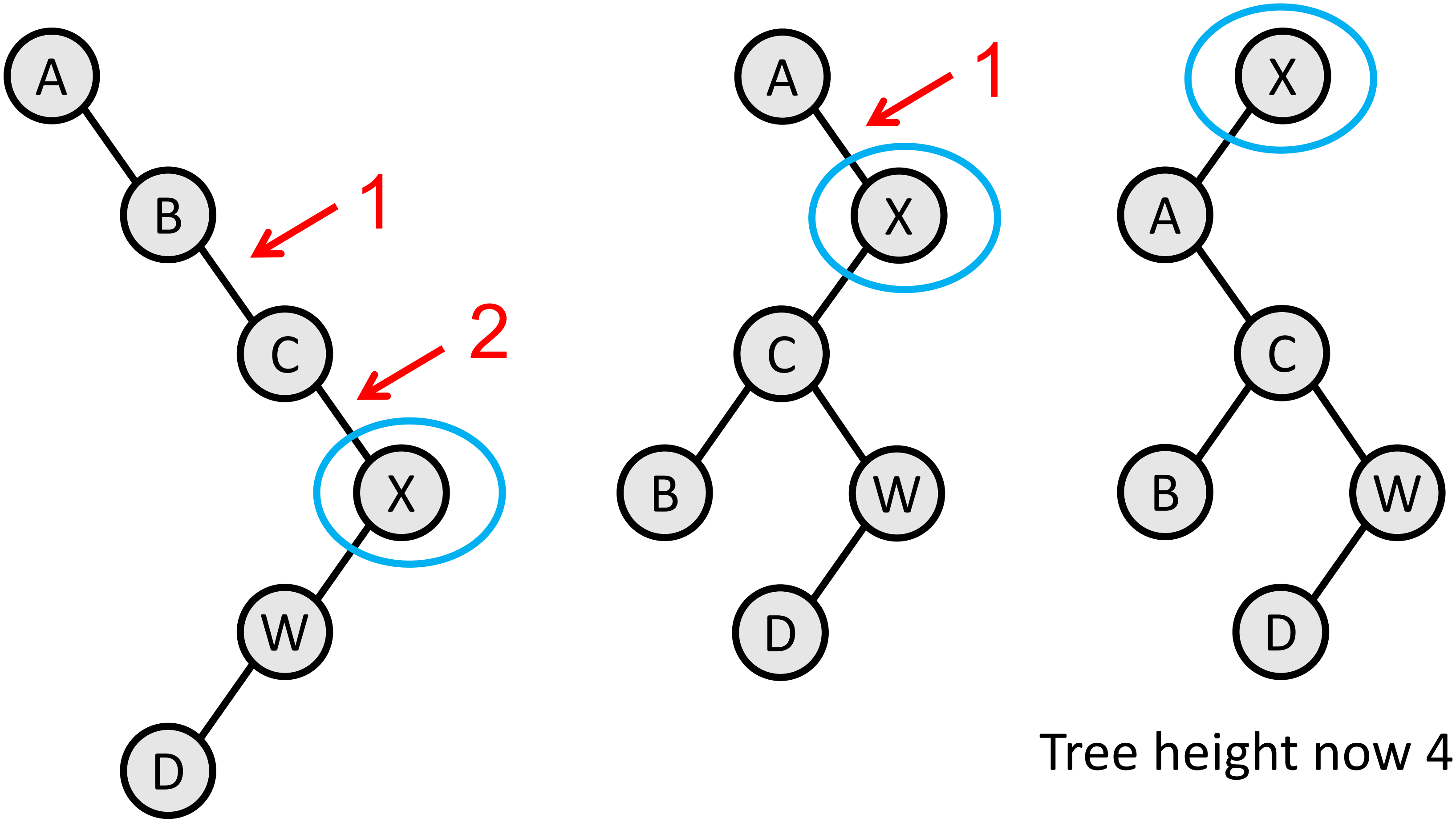
Search for node x:
How many comparisons?

Then splay x to root

Example: Splay tree



Example: Splay tree



Tree height now 4

Splay trees

- Data structure adapts to how you use it!
- Complexity analysis can be challenging
  - Early searches slow, later searches faster
  - Need to average over time, or measure at specific time points

Self-balancing trees

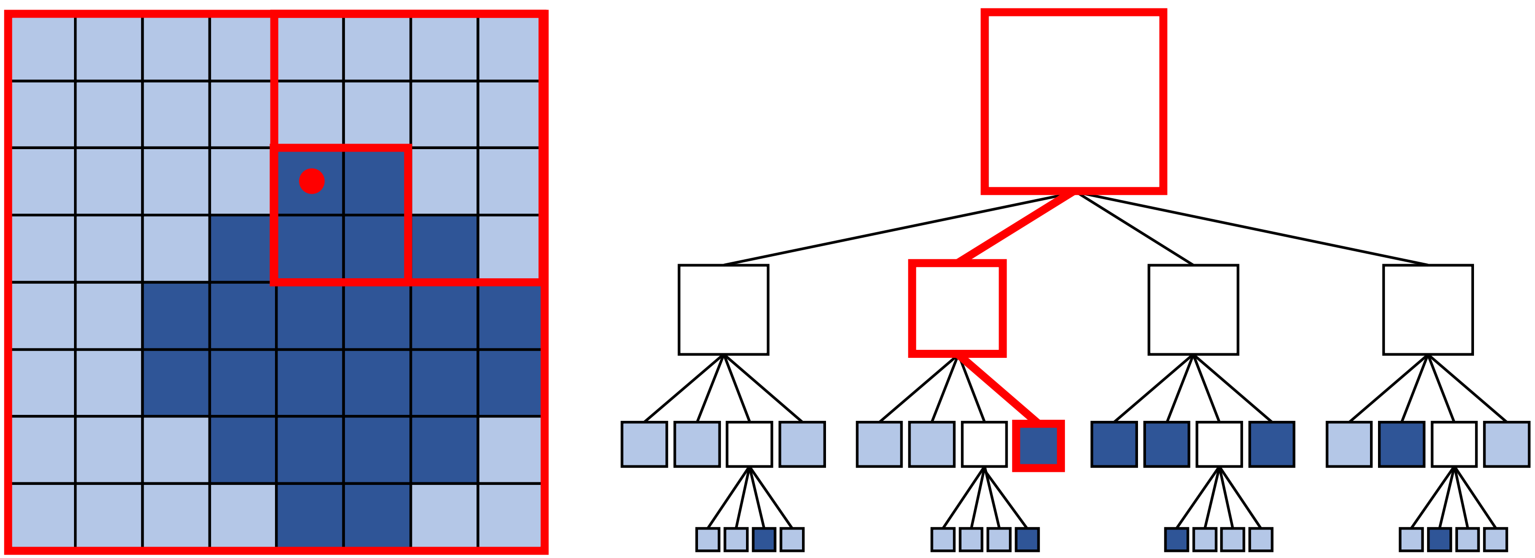
- Types of trees which are always (approximately) balanced:
  - AVL trees
  - B / B+ trees
  - Red-black trees
- On average, splay trees tend to be balanced

2D+ trees

- Traditional BST uses a 1D key
  - Examples: student id, product id
- But many problems involve higher-dimensional search:
  - Examples: image pixels, (x,y,z) coordinates
- BST can be extended to 2D, 3D, etc.

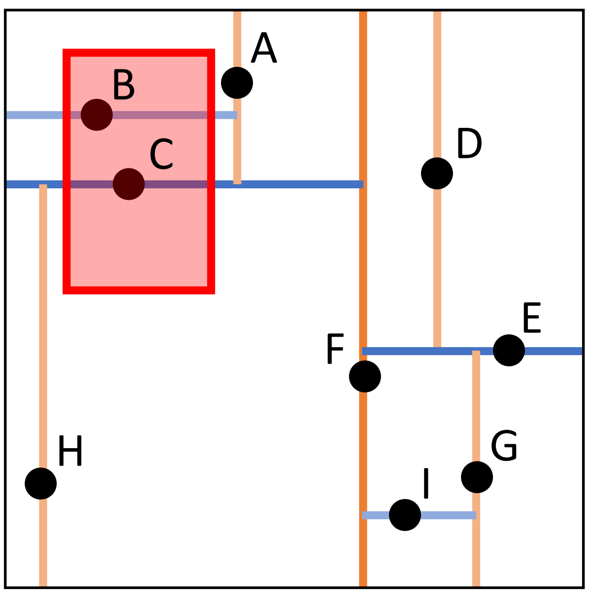
Quad trees

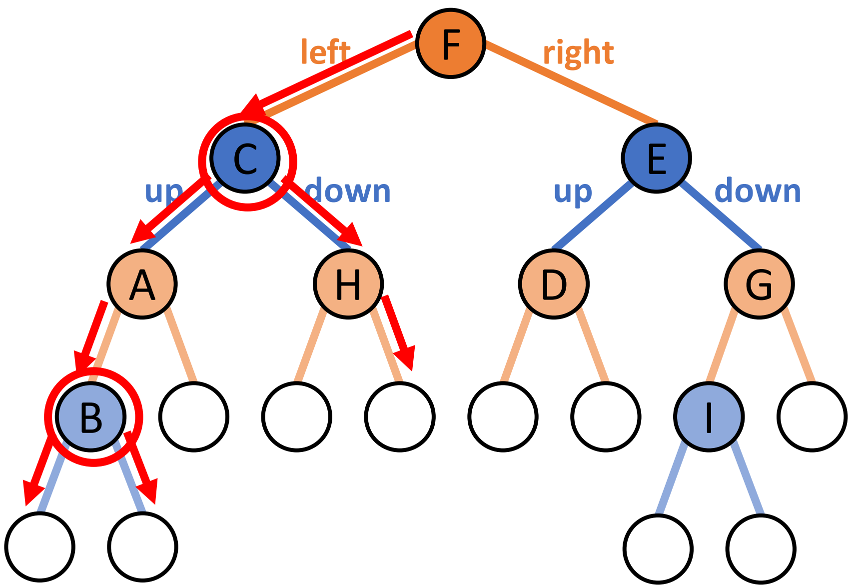
- Each node has 4 children corresponding to quadrants (NW, NE, SE, SW)
  - Nodes contain information (colour, label)
  - Split up regions with more detail



K-D trees

- Binary trees, but each level splits along a different dimension (e.g., x, y, x, y, …)
- Efficient search in high dimensions





Summary

- Lots of ways to extend the basic BST idea!
- Trees can be N-ary, not just binary:
  - 2-3-4 (B) trees, B+ trees, quad trees
- Keys can be multidimensional:
  - K-D trees, quad trees
- Trees can be dynamic:
  - splay trees