Hashing -

- Anything to number
- Arbitrary value
- Multiplication, exponentiation, modular arithmetic, XOR
- Used for – Needs to be quick
  - Constant time lookup
  - Quick comparisons
  - Containers/Bags (probability, bloom filter)
  - Integrity (checksums)
  - Cryptography (needs to be slow)

Variations of Binary search trees for different operations.

Tree Rotation – Adjust binary tree in constant time

- Swap right or left child with a parent and change the children of nodes to ensure that the binary tree is valid
- Right rotation and left rotation
- show page – https://en.wikipedia.org/wiki/Tree_rotation
- Swap parent and child and swap middle subtree
- Many permutations in implementation (unless defined recursively)

Splay Tree – Self adjusting tree that optimizes for recently searched items.

- Most searched for things are at the top of tree
- Advantage - Elements at the top of the tree are accessed more quickly
- Disadvantage – Not balanced so some elements look like they are much longer for search (like a linked list)
- Show example - https://www.cs.usfca.edu/~galles/visualization/SplayTree.html

AVL Tree – Self balancing tree within 1 of height

- When adding items, ensure that sub tree heights are within one of each other
- Advantage – very balanced tree
- Disadvantage – Slower adding elements and many rotations (costly in time)
- Show example - https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

Red Black Tree – Self balancing, longest path is only twice shortest path

- When adding items, label as red or black and use swapping and changing labels to ensure structure of tree and labeling.
- Advantage – Mostly balanced tree
- Disadvantage – Variation in shortest and longest path
- show example – https://www.cs.usfca.edu/~galles/visualization/RedBlack.html

Prefix Tree – https://www.cs.usfca.edu/~galles/visualization/Trie.html

Suffix Tree – http://www.allisons.org/ll/AlgDS/Tree/Suffix/