# Quick Sort

* Partition all numbers based on a pivot, *p* (0th number chosen in class)
* Increase *i* until *a*[*i*] > *p* and decrease *j* until *a*[*j*] < *p*
  + Swap *i* and *j*
* When *i* and *j* intersect, swap *a*[*j*] with *p*
* Recurse with left and right side of *p*

### **Analysis of Quick Sort**

* Worst case: *O(n²)*
* Average case: *Θ(n \* log(n))*

# Topological Sorting

* Linear ordering of a direct acyclic graph
* For each edge from *u* → *v*, *u* comes before *v*

## DFS-based Implementation

1. Identify the starting nodes (nodes where in-degree = 0)
2. Perform DFS
3. Store the order nodes are popped off of the stack into another stack, *output*
4. Pop off each node from *output*, reversing the order. This is the topological ordering

A picture containing diagram

Description automatically generated

## Khan’s Algorithm (source removal)

1. Place all nodes where in-degree = 0 into queue, *queue*
2. While *queue* is not empty…
   1. Poll the node from *queue* and push value into list, *output*
   2. If node as children…
      1. For child in node’s children…
         1. Decrease child’s in-degree count
         2. If in-degree of child is 0…
            1. Add child to *queue*
3. The list, *output*, contains the topological ordering

### **Analysis of Topological Sort**

* Adjacency matrix: *Θ(|V|²)*
* Adjacency list: *Θ(|V| + |E|)*

# AVL Trees

* Balanced BST
* Disadvantages
  + Many rotations, convoluted code
* Difference in height between left and right subtrees is ⩽ 1
* Rotations
  + Local transformation of subtree whose balanced factor has become ±2

## LL-Problem RR-Problem

A picture containing background pattern

Description automatically generatedLogo, company name

Description automatically generated

## LR-Problem RL-Problem

A picture containing text, whiteboard

Description automatically generatedWhiteboard

Description automatically generated with medium confidence

### **Average height of AVL tree for large n:**

* Height: *1.01 \* log(n) + 0.1*

# 2-3 Trees

* Each node can have one or two values
* All leaves of a 2-3 tree are at the same level

Diagram

Description automatically generated

# Heaps

* Complete tree
* Value of each node is greater than children

### **Analysis of Heap operations:**

* Searching: *O(1)*, Removing: *O(log(n))*, Inserting: *O(log(n))*

### **Aside— LCM and GCD:**

* lcm(m, n) = (n \* m) / gcd(m, n) and gcd(m, n) = gcd(n, mod(m, n))

## Heapsort

* Apply remove operation n – 1 times
* O(n \* log(n)) time efficiency

## Heap Construction

* Top-down heap construction
  + Keep inserting, O(n \* log(n))
* Better approach is to place members in heap

Chart

Description automatically generated

## Warshall’s Algorithm

* Transitive closure of directed graph
  + Checks if path exists between two nodes
* Calculate n \* n matrices R(0), R(1), …, R(n) where R(k)[*i*][*j*] = 1 path *i* → *j* with only first *k* nodes allowed as intermediate nodes

Diagram

Description automatically generated

## Floyd’s Algorithm

* All pairs shortest path

*A picture containing diagram

Description automatically generated*

### **Analysis of Floyd and Warshall’s Algorithm:**

* *Θ(n3)*

## Prim’s Algorithm

* Find Minimum spanning tree (MST) in graph
* MST
  + Spanning tree with smallest weight among candidates

A picture containing diagram

Description automatically generated

## Kruskal’s Algorithm

* Sort edges of a graph in non-decreasing order
* Grow tree by adding an edge at a time from sorted unless edge would create a cycle

## Union-find’s Algorithm

* Used to find cycles in a graph
* Used in Kruskal’s algorithms

Text, letter

Description automatically generatedA picture containing watch, clock

Description automatically generated

## Rooted Trees

* Children point to parent
* Parent cannot have more than two children

### Represent with array

Diagram

Description automatically generated