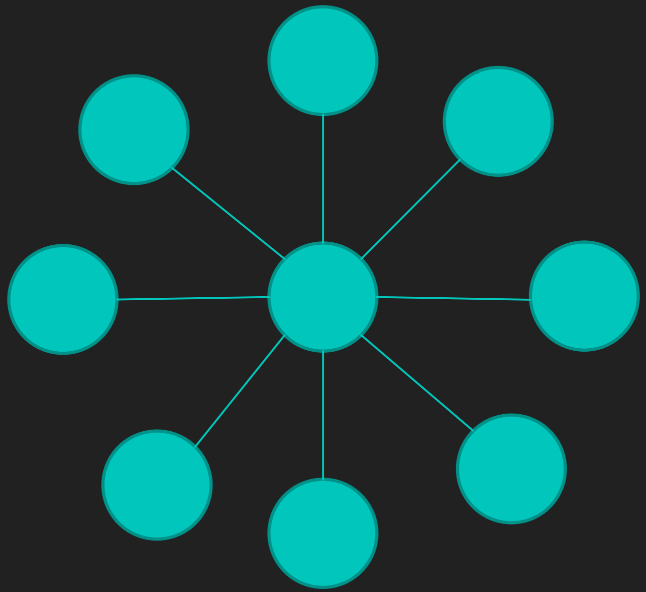
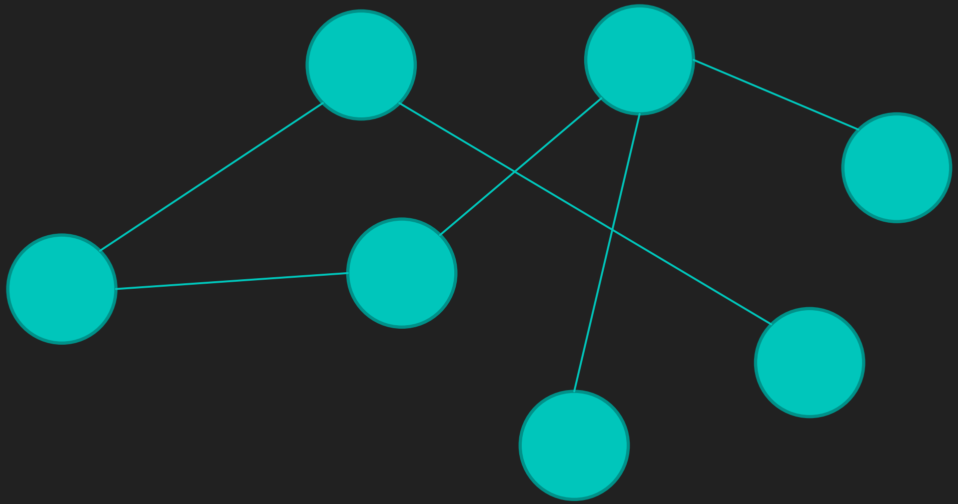
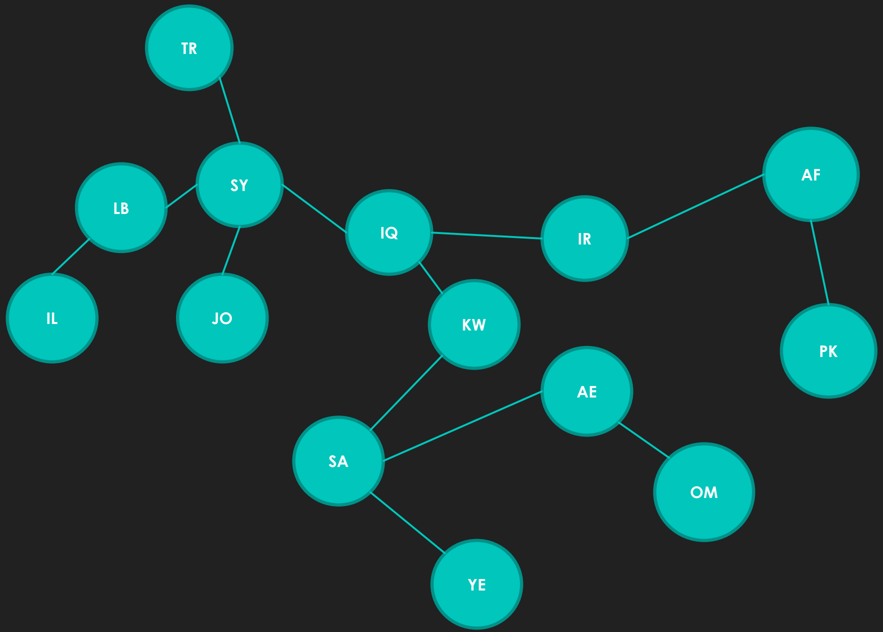
Tree

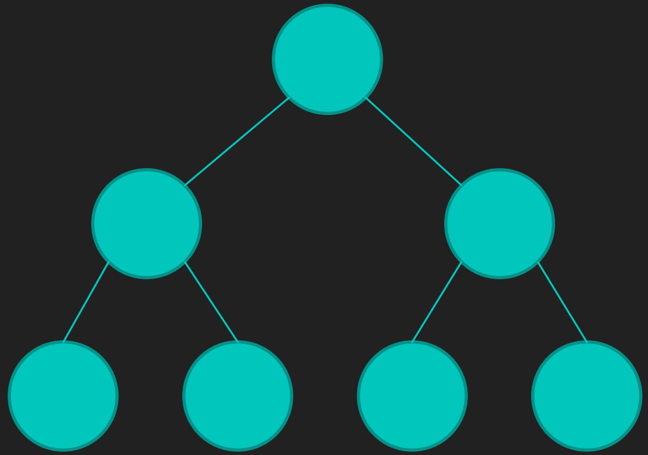
Mohammad Ghoddosi

Tree







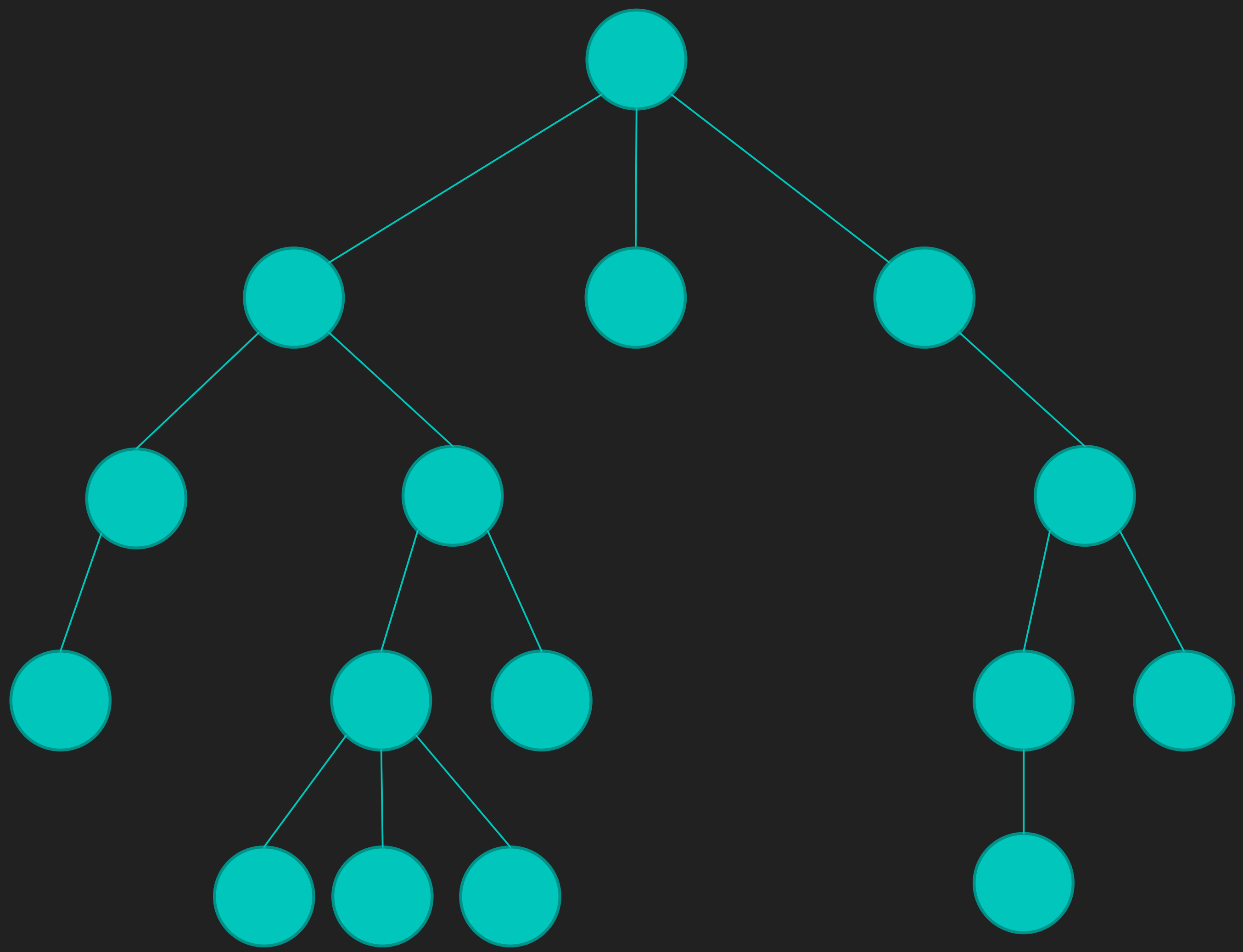


Tree

- Undirected graph
- Any two vertices are connected with exactly one path
- No cycles
- Connected

Terminology

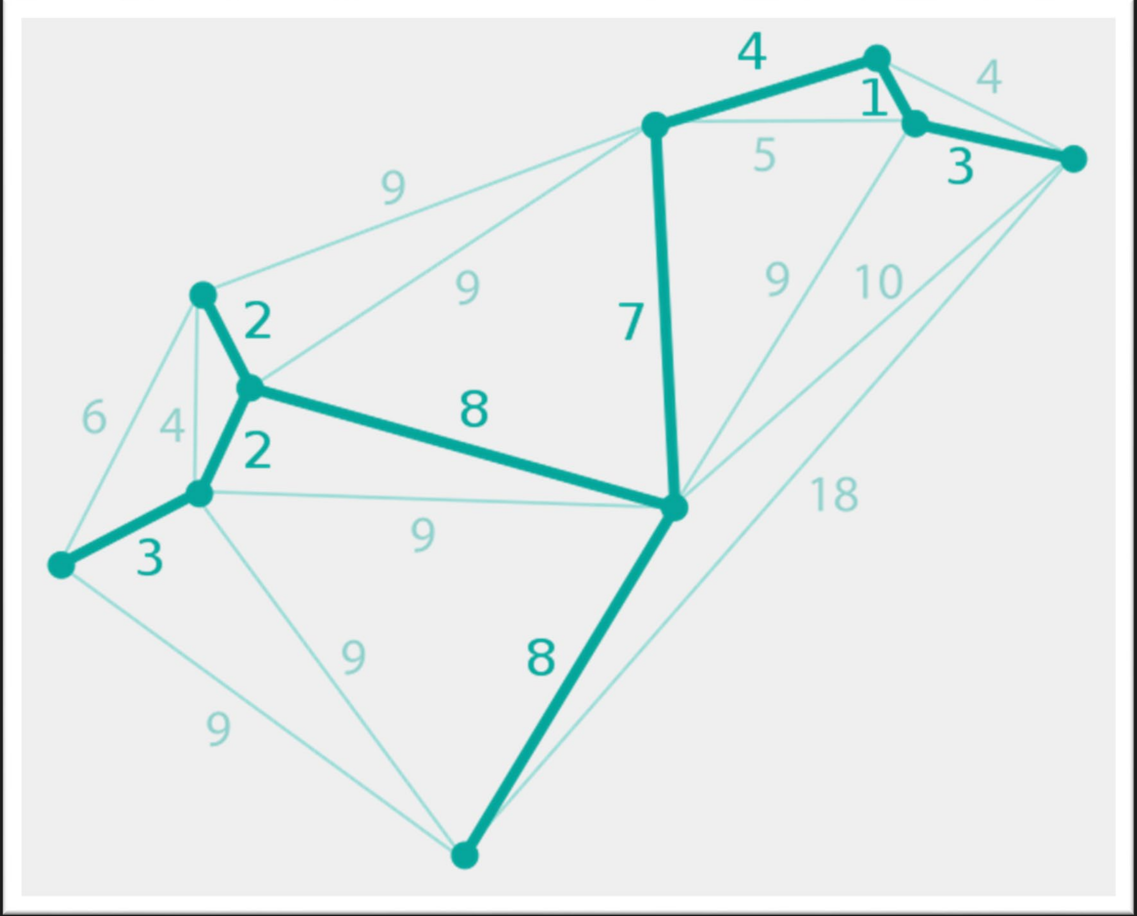
- Root
- Children
- Parent
- Sibling
- Uncle
- Leaf
- Height
- Level



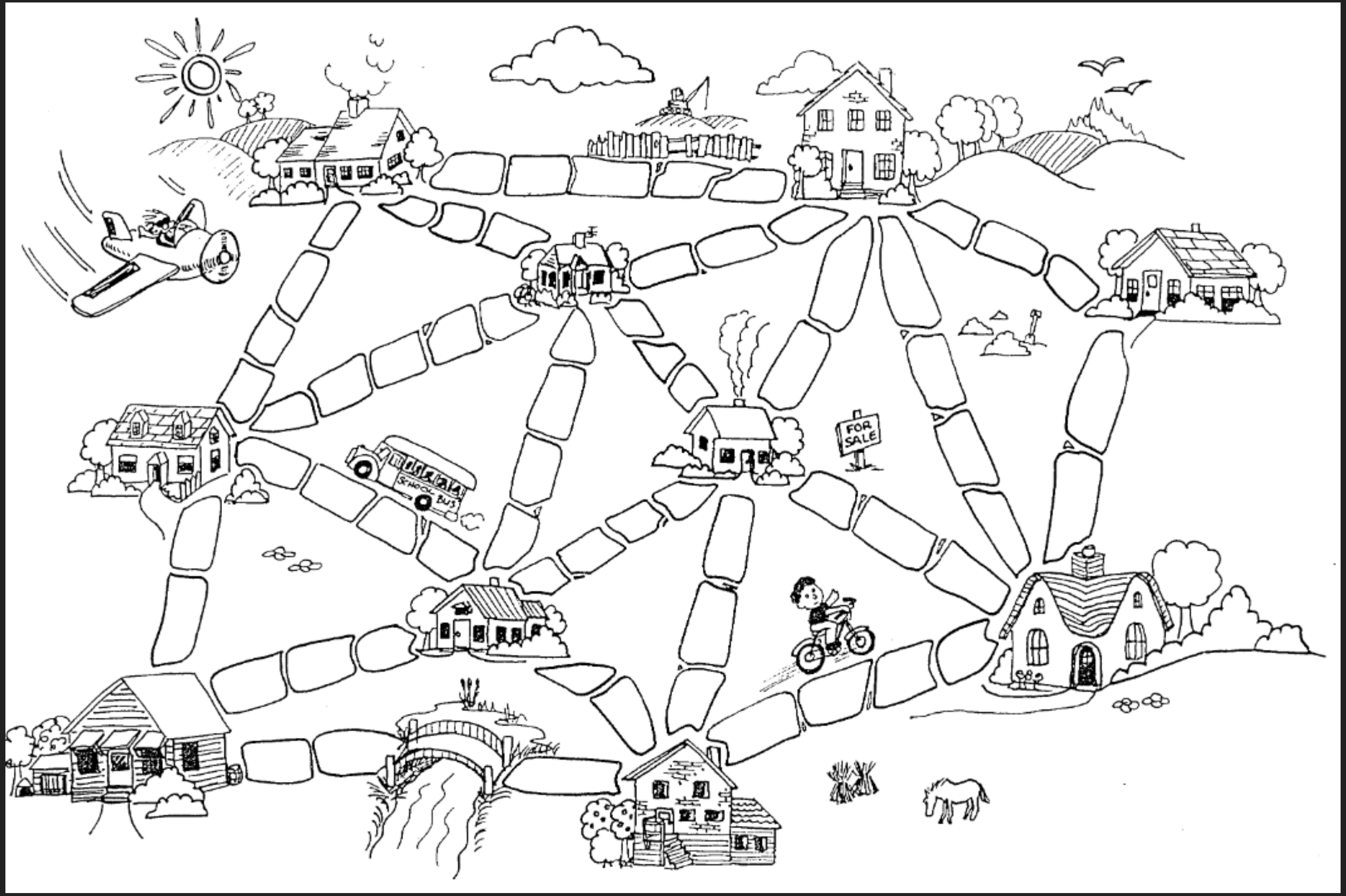
Minimum Spanning Tree (MST)

- For a weighted, connected, undirected graph
- Spanning tree
- Minimum total edge weight

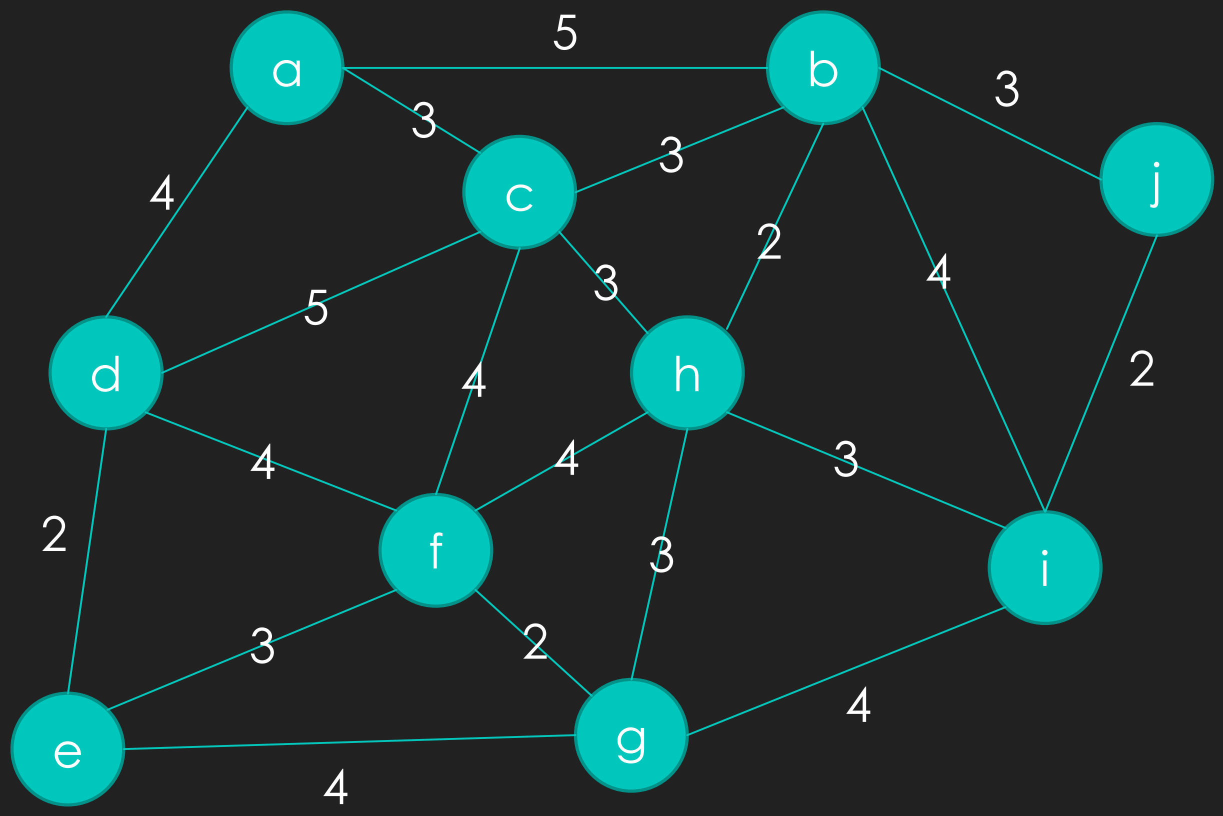
- Network design
- Clustering
- Image segmentation
- Handwriting recognition
- …



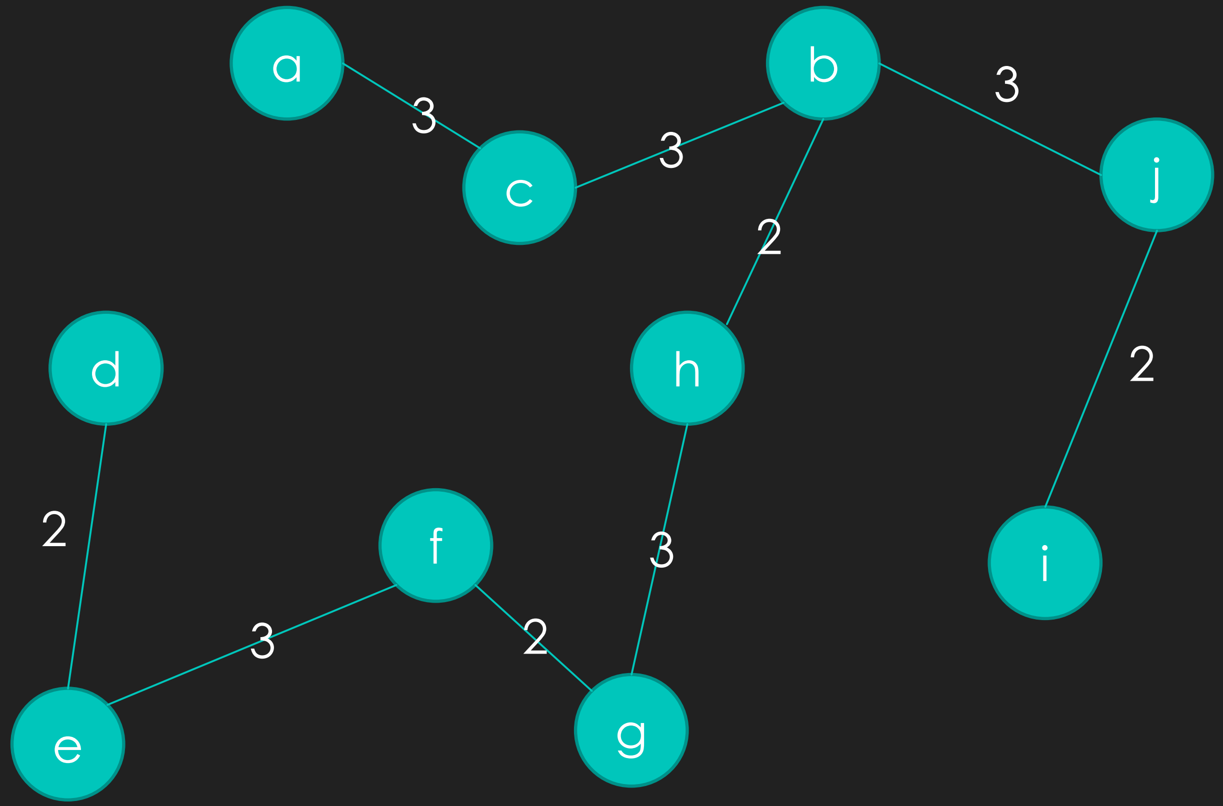
Muddy city problem



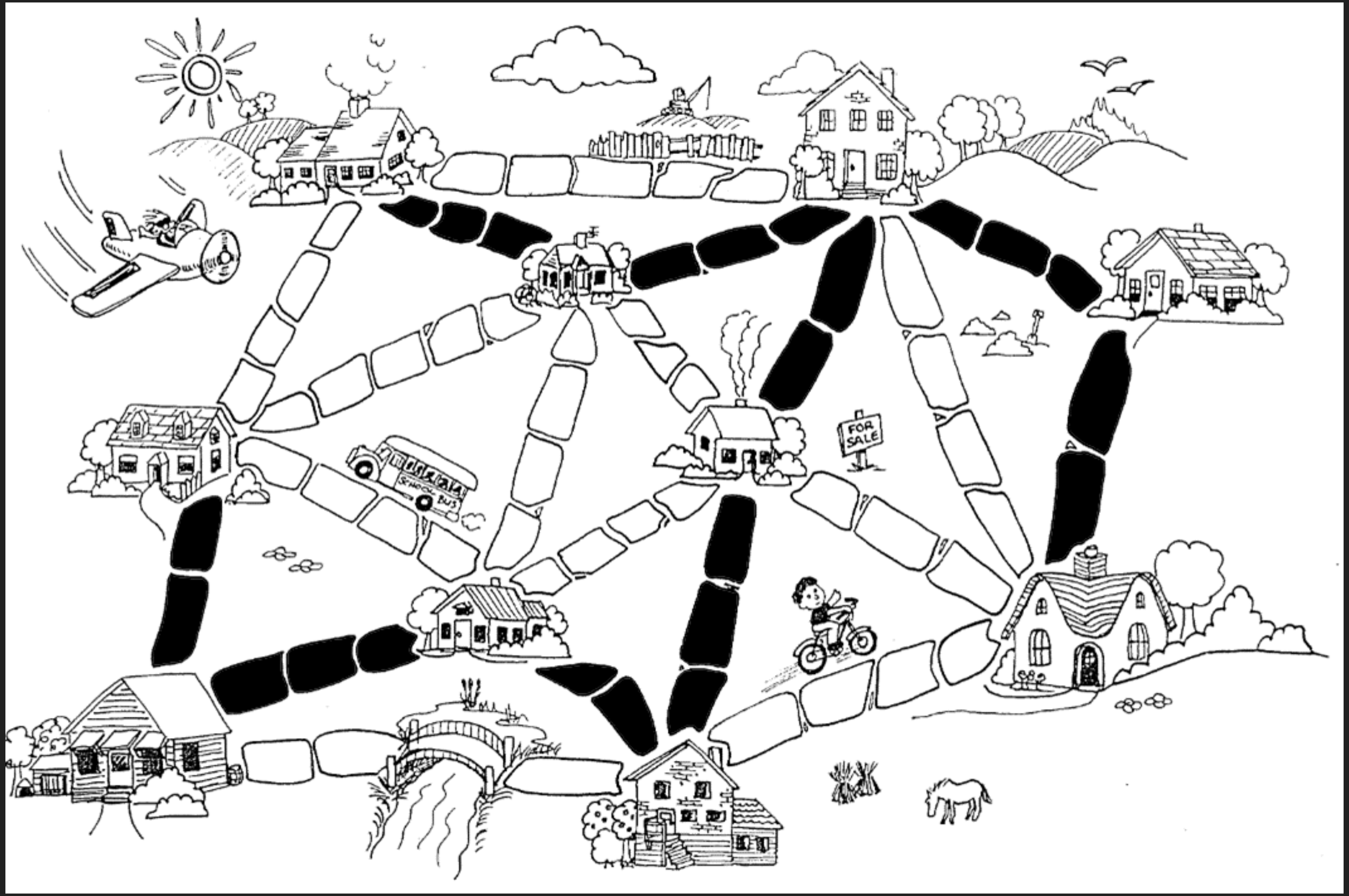
Muddy city problem



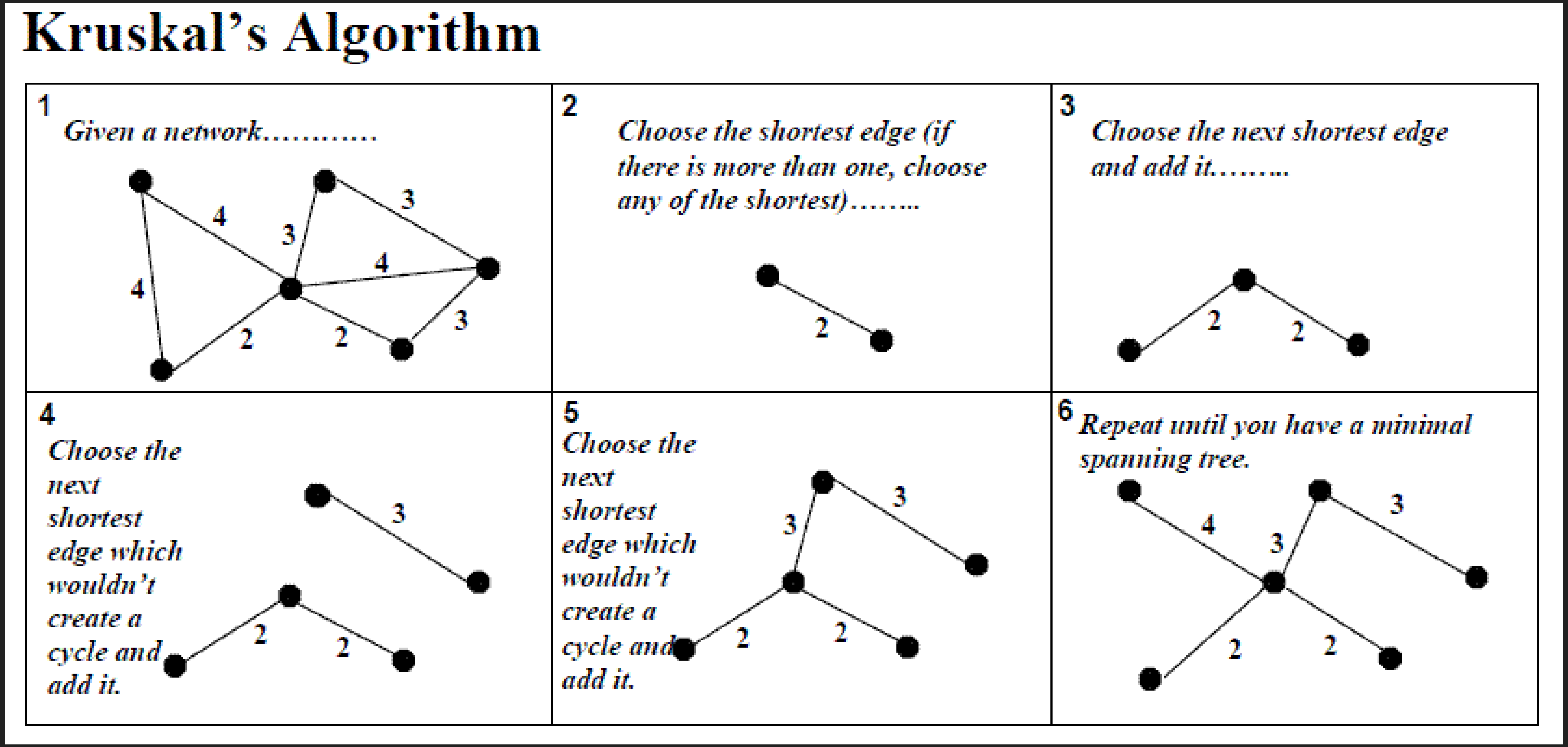
Muddy city problem



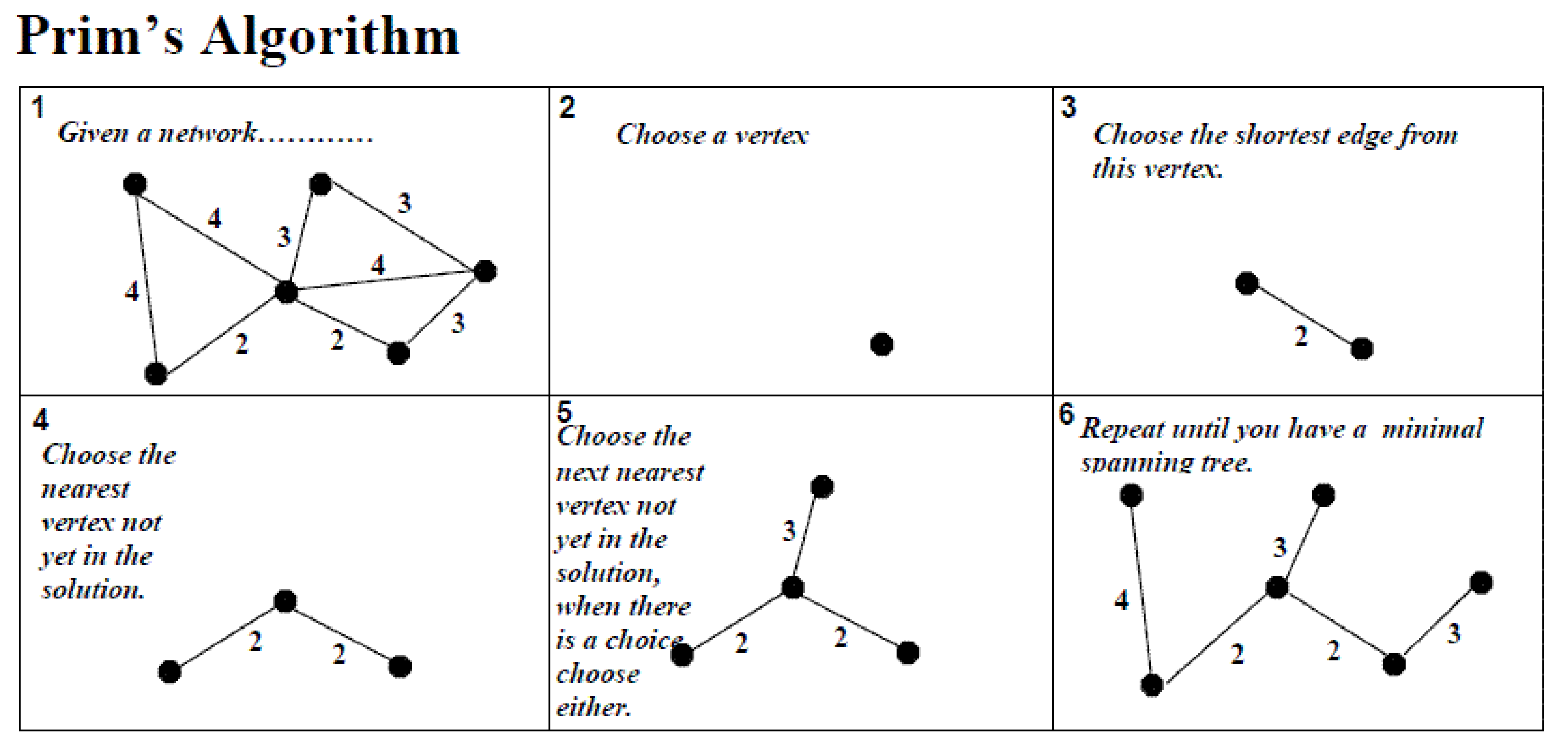
Muddy city problem



Kruskal


Kruskal's Algorithm
1 Given a network…………
2 Choose the shortest edge (if there is more than one, choose any of the shortest)……..
3 Choose the next shortest edge and add it……..
4 Choose the next shortest edge which wouldn't create a cycle and add it.
5 Choose the next shortest edge which wouldn't create a cycle and add it.
6 Repeat until you have a minimal spanning tree.

Prime

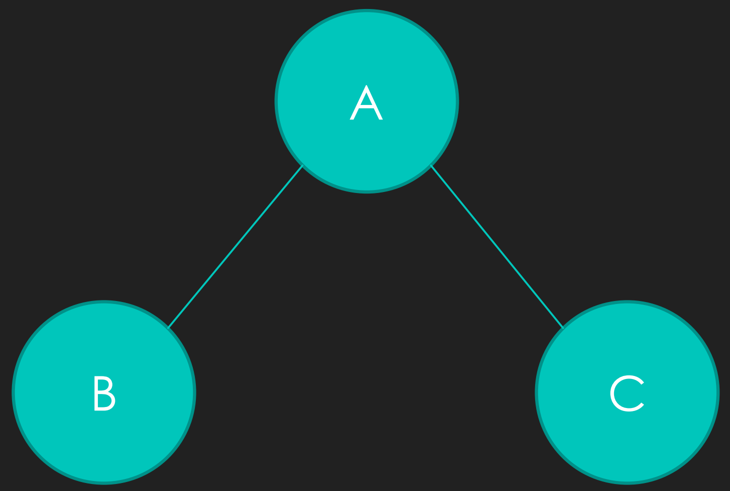

Prim's Algorithm
1
Given a network…………
4
3
3
4
4
3
2
2
2
Choose a vertex
3
Choose the shortest edge from this vertex.
2
4
Choose the nearest vertex not yet in the solution.
2
2
5
Choose the next nearest vertex not yet in the solution, when there is a choice choose either.
3
2
2
6
Repeat until you have a minimal spanning tree.
4
3
2
2
3

Binary tree

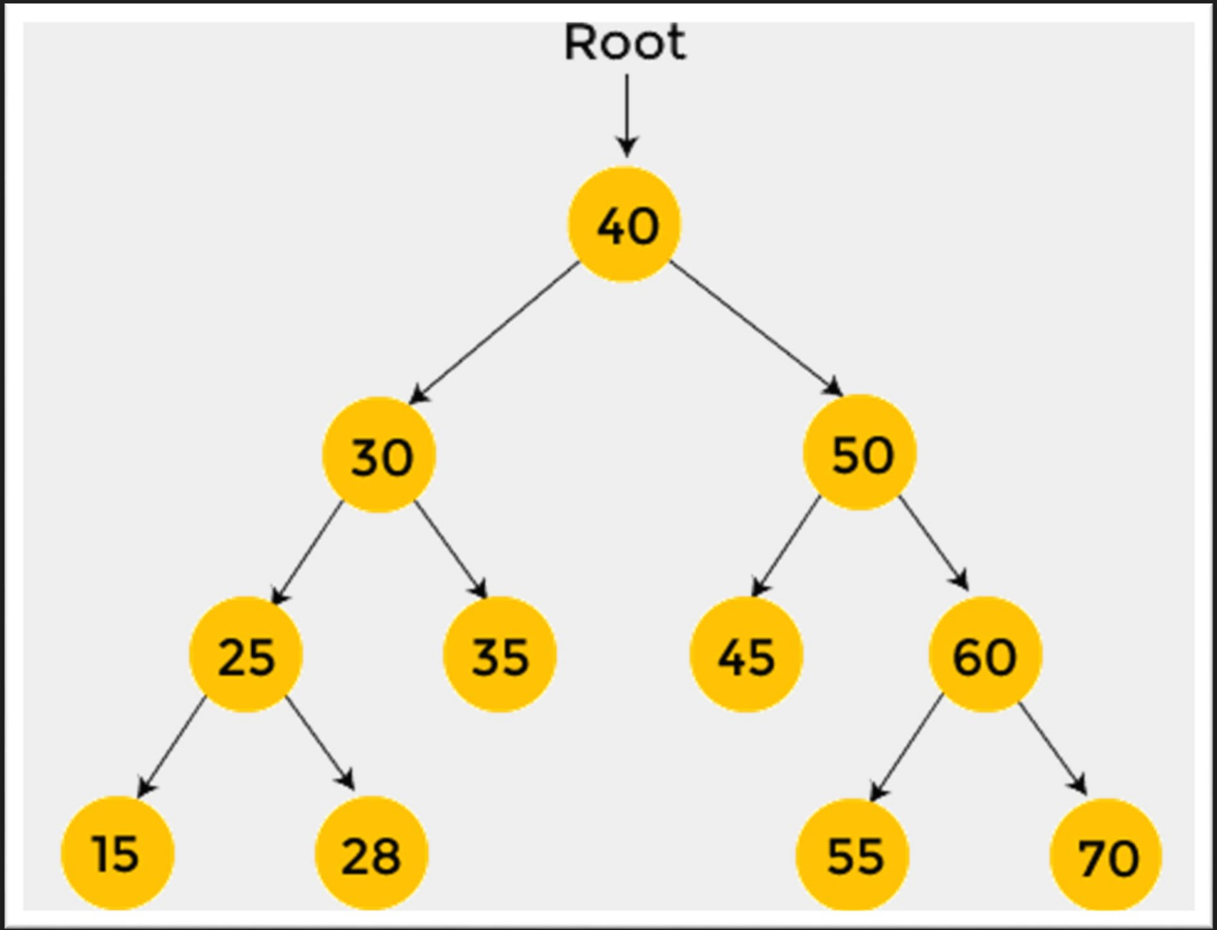
- Each nodes has max of 2 children
- Full binary tree
  - Every node has 0 or 2 children
- Used for:
  - Searching
  - Sorting
  - Compression algorithm
  - Decision tree
- Array implementation

Traversal

- Pre-order
  - ABC
- Post-order
  - BCA
- In-order
  - BAC



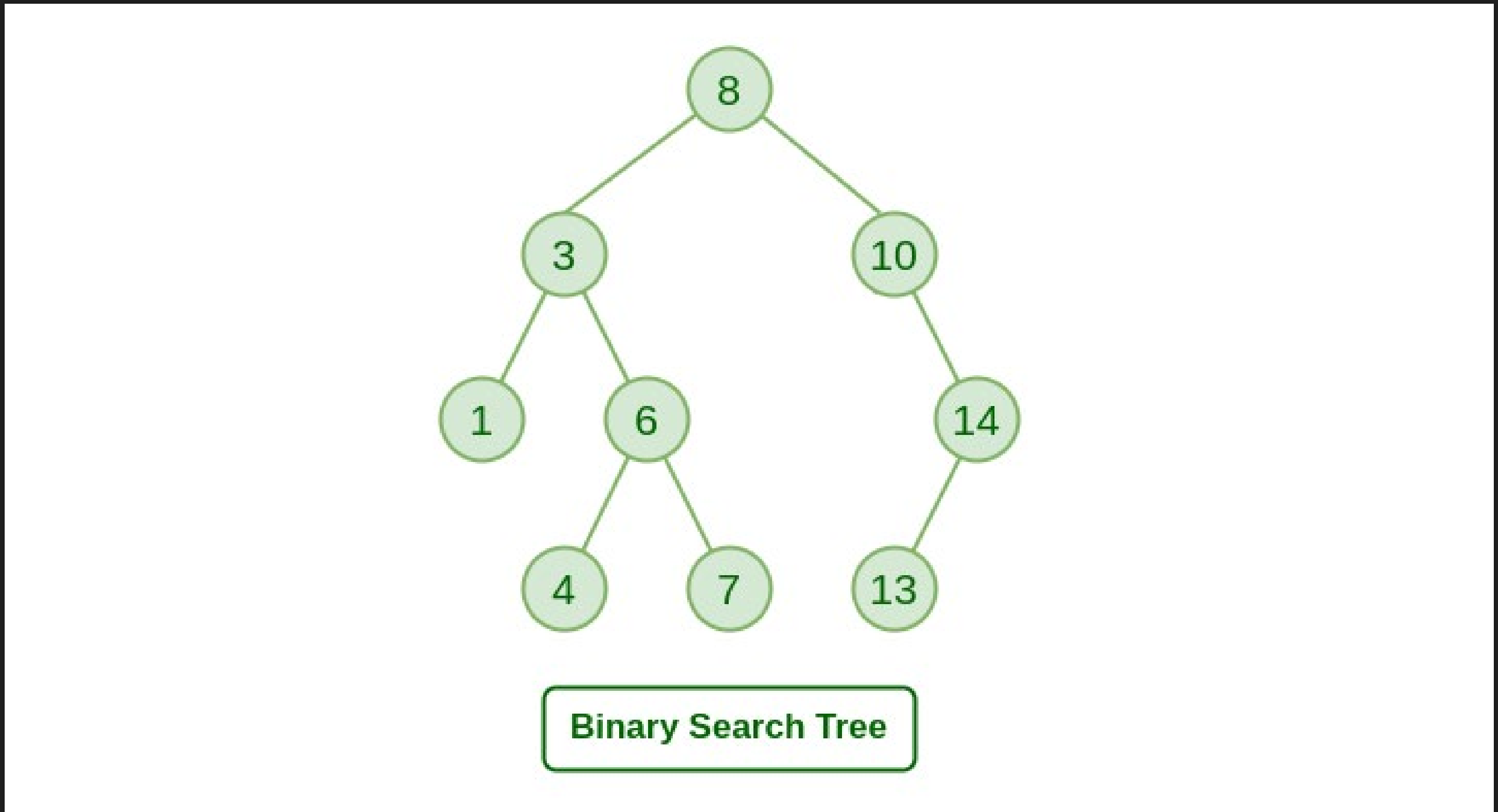
Traversal



Binary search tree

- Fast insertion and removal of elements
- Fast search
- Binary search
- Binary tree
  - Each node has 2 children
- Left subtree: only values less than the node
- Right subtree: only values grater than the node

Binary search tree



Binary search tree

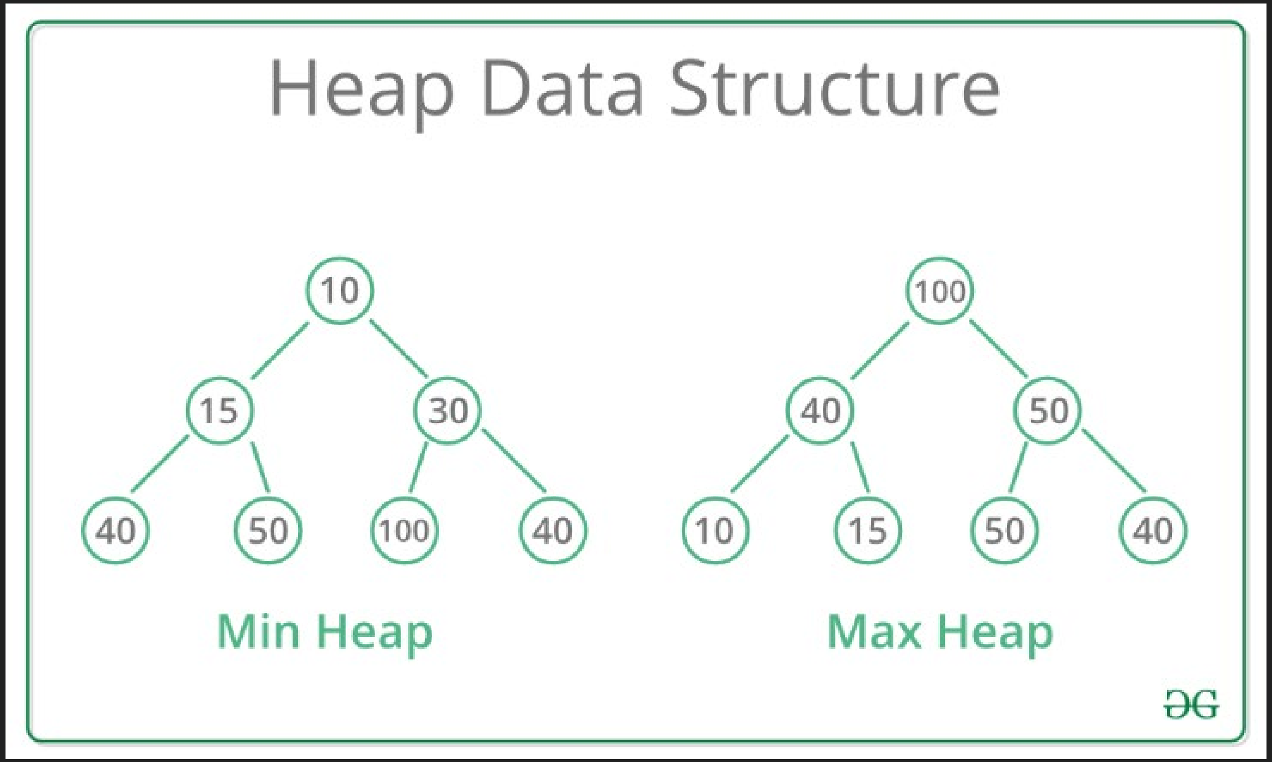
- Insertion
- Search
  - Binary search
- Deletion
  - In-order successor

Self balancing Binary Search Tree

- Search, Insert, Delete, … is O(h)
- Sort in O(n) (without insertion cost)
- Height problem in BST
- Self balancing BSTs
  - AVL
  - Red and black trees
  - H = O(log n)
- Simple to find all numbers greater than … or smaller than …

Heap

- Complete binary tree
- Max heap
  - Root is always greater than its children
- Min heap
  - Root is always smaller than its children
- Operations:
  - Heapify
  - Insertion
  - Deletion
  - peak



Heap

- Fast access to maximum/minimum in (O(1))
- Efficient insertion and deletion (O(log n))
- Efficient implementation with arrays
- Priority queue
- Not good at searching for a value (O(n))