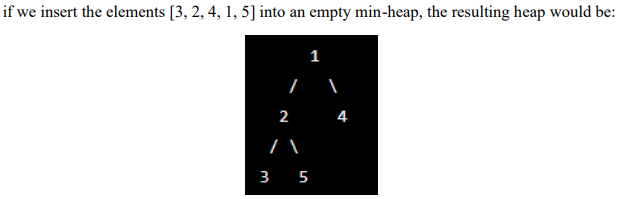


**Min-Heap Insertion Example:**



O(nlog(n))

**SiftDown/SiftUp:**

O(log(n))

**BFS/DFS/Topological Sorting:**

O(|E| + |V|)

**Prim/Kruskal:**

O(E log V)

**Dijkstra:**

O((V+E) log V)

**Ford-Fulkerson:**

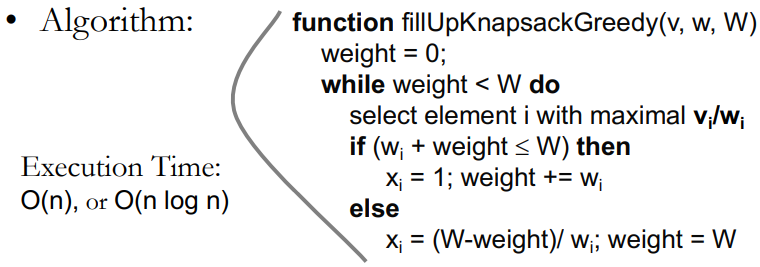
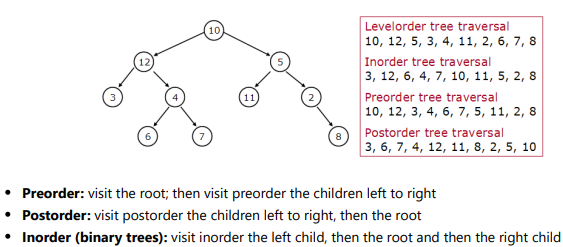
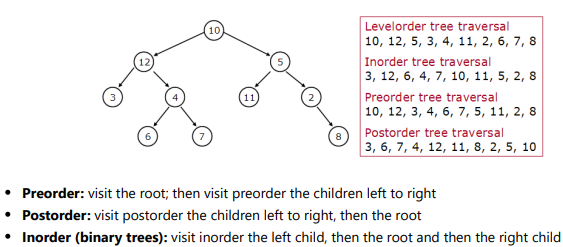
O(|E|\*f)

**Edmonds-Karp:**

O(VE^2)

**Maximal Bipartite:**

O(V E)



**Knapsack:**

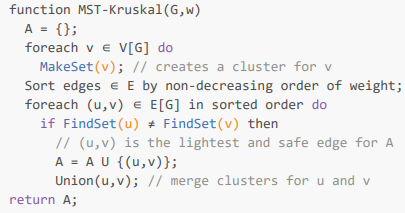
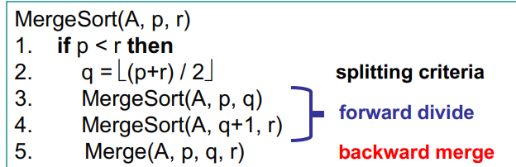
O algoritmo greedy encontra a solução ótima para o fractional Knapsack problem porque ordena os objetos com base na relação valor/peso e seleciona primeiro os objetos com maior ratio. Contudo, a mesma abordagem não encontra a solução ótima para o integer Knapsack problem, onde os objetos não podem ser divididos em partes fracionárias. Um objeto com um ratio elevado pode ter um peso superior à capacidade restante da mochila, impedindo a adição de objetos com uma relação valor/peso menor, mas com um peso que caiba na capacidade restante.

Fractional Knapsack: O(n log(n))

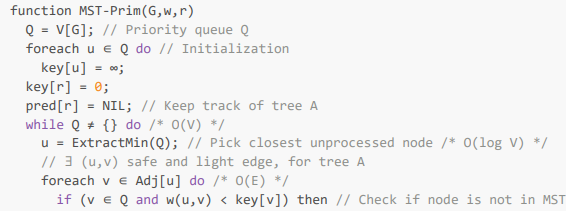


**Merge-Sort:**

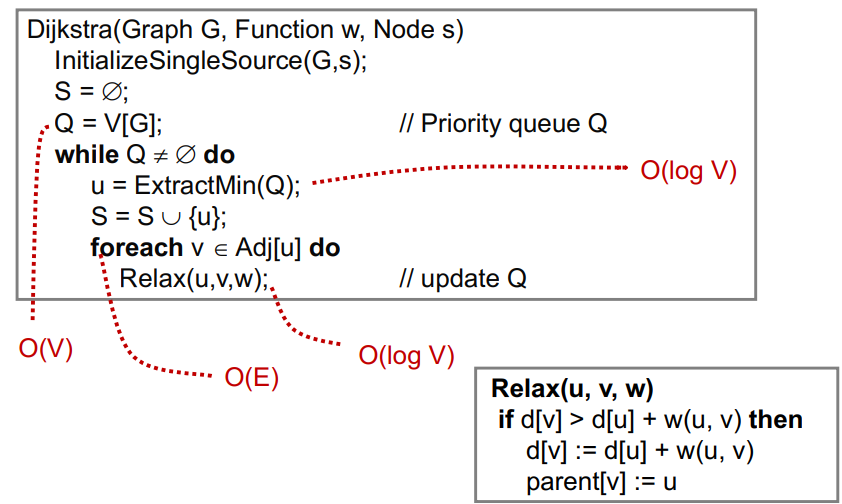
Time Complexity: O(n log n)



**Kruskal:**



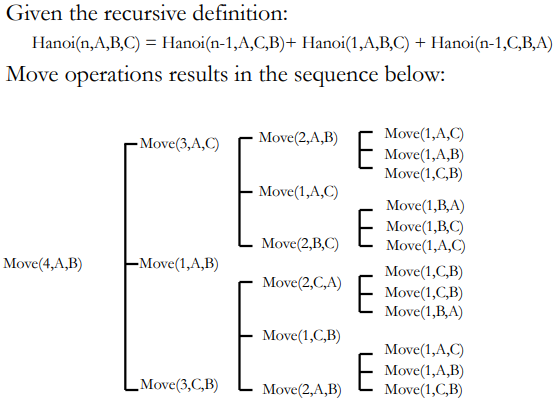
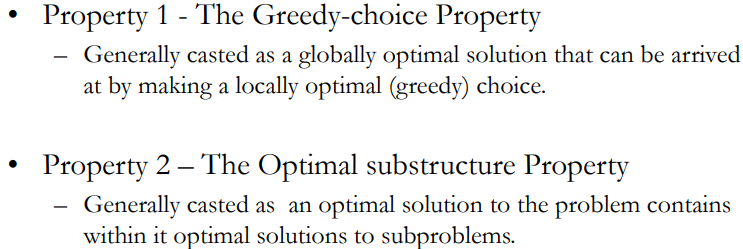
**Prim:**



**Dijkstra:**

Text

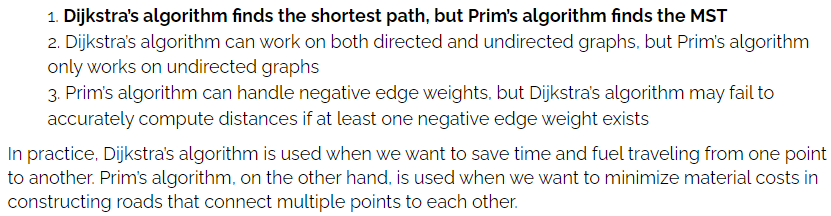
Description automatically generated with medium confidence



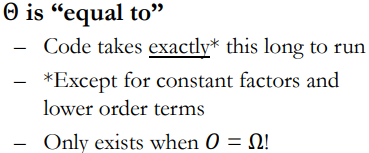
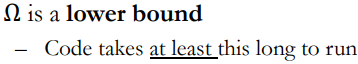
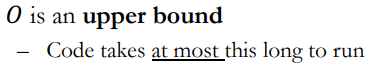
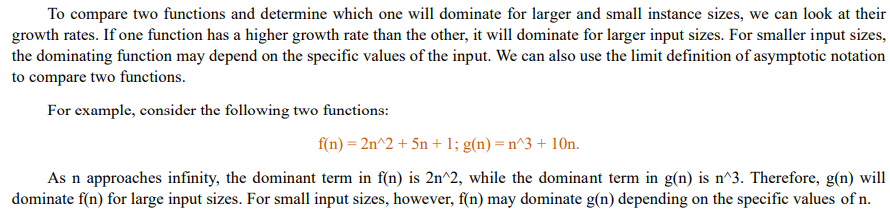
**Hanoi Towers:**

Time Complexity: O(2^n)

**Master Theorem:**

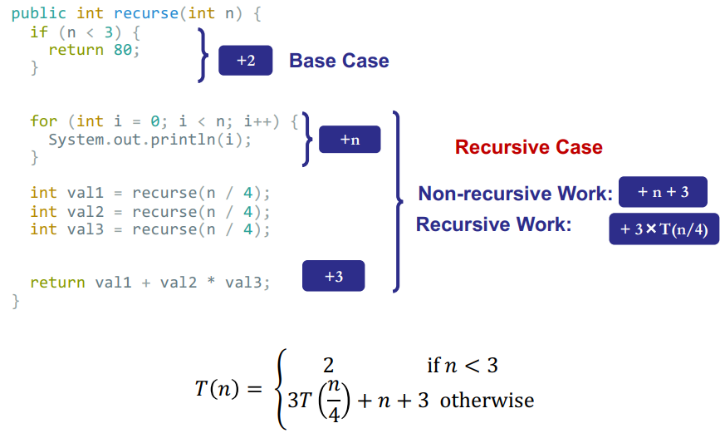


**Dijkstra vs Prim:**



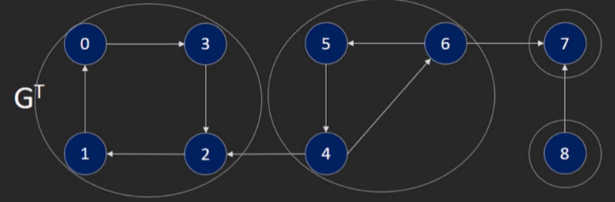
Text, letter

Description automatically generated



**Deriving Recurrences Example:**

A picture containing text, sign

Description automatically generatedTimeline

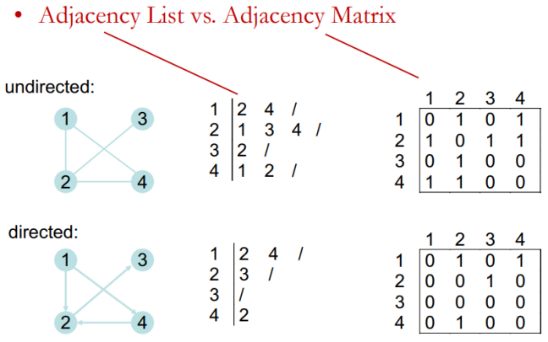
Description automatically generated with medium confidenceGraphical user interface, text, application

Description automatically generatedGraphical user interface, text

Description automatically generatedText

Description automatically generatedText

Description automatically generated

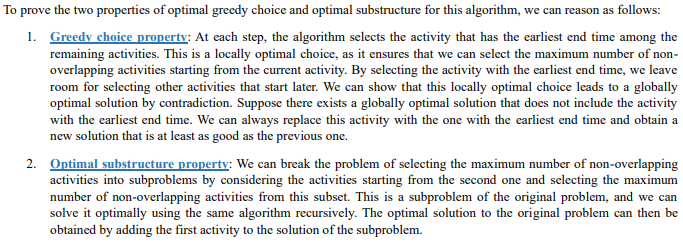
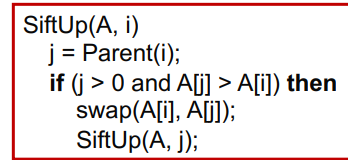


**DFS/BFS using Adjacency Matrix:** O(V^2)

Because each vertex needs to be visited at most once, and for each vertex, we need to examine all V adjacent vertices in the matrix.

**DFS/BFS using Adjacency List:** O(V + E)

**Strongly Connected Components (Kosaraju’s Algorithm Example):**



**Activity Selection Example: Prove the two properties of optimal greedy:**

A picture containing text, watch

Description automatically generated

**Dijkstra Example:**