**Greedy Algorithm**

A greedy algorithm is an approach for solving a problem by selecting the best option available at the moment, without worrying about the future result it would bring. In other words, the locally best choices aim at producing globally best results.

This algorithm may not be the best option for all the problems. It may produce wrong results in some cases.

This algorithm never goes back to reverse the decision made. This algorithm works in a top-down approach.

**What is Minimum Spanning Tree?**   
Given a connected and undirected graph, a spanning tree of that graph is a subgraph that is a tree and connects all the vertices together. A single graph can have many different spanning trees. A minimum spanning tree (MST) or minimum weight spanning tree for a weighted, connected and undirected graph is a spanning tree with weight less than or equal to the weight of every other spanning tree. The weight of a spanning tree is the sum of weights given to each edge of the spanning tree.  
How many edges does a minimum spanning tree has?   
A minimum spanning tree has (V – 1) edges where V is the number of vertices in the given graph.

## **FORD – FULKERSON ALGORITHM**

Ford-Fulkerson algorithm is a [greedy approach](https://www.programiz.com/dsa/greedy-algorithm) for calculating the maximum possible flow in a network or a graph.

A term, **flow network**, is used to describe a network of vertices and edges with a source (S) and a sink (T). Each vertex, except **S** and **T**, can receive and send an equal amount of stuff through it. **S** can only send and **T** can only receive stuff.

AUGMENTED-PATH : It is a path available in flow network where we consider

1. Non-full forward edges.
2. Non-empty backward edges.

The algorithm follows:

1. Initialize the flow in all the edges to 0.
2. While there is an augmenting path between the source and the sink, add this path to the flow.
3. Update the residual graph.

## **PRIM** **ALGORITHM**

The steps for implementing Prim's algorithm are as follows:

1. Initialize the minimum spanning tree with a vertex chosen at random.
2. Find all the edges that connect the tree to new vertices, find the minimum and add it to the tree
3. Keep repeating step 2 until we get a minimum spanning tree

## **KRUSKAL ALGORITHM**

**1.** Sort all the edges in non-decreasing order of their weight.   
**2.** Pick the smallest edge. Check if it forms a cycle with the spanning tree formed so far. If cycle is not formed, include this edge. Else, discard it.   
**3.** Repeat step#2 until there are (V-1) edges in the spanning tree.

## **HUFFMAN CODING**

Huffman Coding is a technique of compressing data to reduce its size without losing any of the details. It was first developed by David Huffman.

Huffman Coding is generally useful to compress the data in which there are frequently occurring characters.

Huffman coding first creates a tree using the frequencies of the character and then generates code for each character.

Once the data is encoded, it has to be decoded. Decoding is done using the same tree.

Huffman Coding prevents any ambiguity in the decoding process using the concept of **prefix code** ie. a code associated with a character should not be present in the prefix of any other code. The tree created above helps in maintaining the property.

* It is used for the lossless compression of data.
* It uses variable length encoding.
* It assigns variable length code to all the characters.
* The code length of a character depends on how frequently it occurs in the given text.
* The character which occurs most frequently gets the smallest code.
* The character which occurs least frequently gets the largest code.

## **Prefix Rule-**

* Huffman Coding implements a rule known as a **prefix rule**.
* This is to prevent the ambiguities while decoding.
* It ensures that the code assigned to any character is not a prefix of the code assigned to any other character.

STEPS:

1. Calculate the frequency of each character in the string.

2. Sort the characters in increasing order of the frequency. These are stored in a priority queue Q.

3. Make each unique character as a leaf node.

4.Create an empty node z. Assign the minimum frequency to the left child of z and assign the second minimum frequency to the right child of z. Set the value of the z as the sum of the above two minimum frequencies.

5. Remove these two minimum frequencies from Q and add the sum into the list of frequencies (\* denote the internal nodes in the figure above).

6. Insert node z into the tree.

7. Repeat steps 3 to 5 for all the characters.

## **DIJKSTRA ALGORITHM**

Dijkstra's algorithm allows us to find the shortest path between any two vertices of a graph.

It differs from the minimum spanning tree because the shortest distance between two vertices might not include all the vertices of the graph.

## Dijkstra's Algorithm Applications

* To find the shortest path
* In social networking applications
* In a telephone network
* To find the locations in the map

## Dijkstra's Algorithm Complexity

Time Complexity: O(E Log V)

where, E is the number of edges and V is the number of vertices.

Space Complexity: O(V)