Greedy Algorithms

- An algorithm which always takes the best immediate or local solution while finding the answer.
- Greedy algorithms will find the overall or globally optimal solution for some optimization problems but may find less than optimal (suboptimal solutions) for some.
- These algorithms are very easy to design and complexity of greedy algorithms are generally less.
- Greedy is a strategy that works well on optimization problems with the following characteristics:

Greedy-choice property: A global optimum can be arrived at by selecting a local optimum.

Optimal substructure: An optimal solution to the problem contains an optimal solution to subproblems.

Greedy Approach Based Problems:

1. Minimum Spanning Tree Problem

(a) Kruskal Algorithm

(b) Prims Algorithm

2. Shortest Path Problem: Single Source Shortest Path Algorithm

(a) Bellman Ford Algorithm

(b) Dijkstra Algorithm

3. Activity Selection Problem

4. Huffman Coding Problem

5. Fractional Knapsack Problem

Activity Selection Problem

Let S = {1, 2, . . . , n} be the set of activities that compete for a resource. Each activity i has its starting time si and finish time fi with si ≤ fi , namely, if selected, i takes place during time [si , fi). No two activities can share the resource at any time point. We say that activities i and j are compatible if their time periods are disjoint. The activity-selection problem is the problem of selecting the largest set of mutually compatible activities.

GREEDY- ACTIVITY SELECTOR (s, f)

```
1. n ← length [s]
2. A ← {1}
3. j ← 1.
4. for i ← 2 to n
5. do if si ≥ fi
6. then A ← A ∪ {i}
7. j ← i
8. return A
```

Huffman Coding Problem

Storage space for files can be saved by compressing them, i.e. by replacing each symbol by a unique binary string. Here the codewords can differ in length. Then they need to be prefix-free in the sense that no codeword is a prefix of another code. Otherwise, decoding is impossible. Huffman coding problem is the problem of finding, given an alphabet C = {a1, . . . , an} and its frequencies f1, . . . , fn, a set of prefix-free binary code W = [w1, . . . , wn] that minimizes the average code.

Fractional Knapsack Problem

In Knapsack problem, weights and values of **N** items is given we have to put these items in a knapsack of capacity **W** to get the maximum total value in the knapsack.

Based on the nature of the items, Knapsack problems are categorized as

- Fractional Knapsack
- 0-1 Knapsack

- In **Fractional Knapsack Problem**, we can break items for maximizing the total value of the knapsack. We can solve Fractional Knapsack Problem using **Greedy** based approach.

- In the **0-1 Knapsack problem,** we are not allowed to break items. We either take the whole item or don't take it. We can solve **0-1 Knapsack problem** Dynamic programming-based approach.

Minimum Spanning Tree Problem

Spanning Tree: Given an undirected and connected graph G=(V,E), a spanning tree of the graph G is a tree that spans G (that is, it includes every vertex of G) and is a subgraph of G (every edge in the tree belongs to G

Minimum Spanning Tree: The cost of the spanning tree is the sum of the weights of all the edges in the tree. There can be many spanning trees. Minimum spanning tree is the spanning tree where the cost is minimum among all the spanning trees. There also can be many minimum spanning trees. Following algorithms can be used to find MST of a given graph.

1. Kruskal Algorithm
2. Prims Algorithm

Kruskal Algorithm

- In kruskal's algorithm,the set A is forest and safe edge added to A is always a least weight edge in the graph that connect two distint components.
- In kruskal algorithm we apply greedy approach on edges of given graph.
- Min-Heap data structure is use to implement krushkal's algorithm.

```
MST-KRUSKAL(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3      MAKE-SET(v)
4  sort the edges of G.E into nondecreasing order by weight w
5  for each edge (u, v) ∈ G.E, taken in nondecreasing order by weight
6      if FIND-SET(u) ≠ FIND-SET(v)
7          A = A ∪ {(u, v)}
8          UNION(u, v)
9  return A
```

Prims Algorithm

- In prims algorithm we apply greedy approach on vertices of given graph.
- Min-Heap and fibnocii heap data structure can be used to implement prim's algorithm.

```
1.  for each vertex u € V(G)
2.          do key[u] ← ∞
3.              ∏ [u] ← NIL
4.  key [r] ← 0
5.  Q ← V(G)
6.  while Q ≠ ϕ
7.          do u ← EXTRACT-MIN(Q)
8.                  for each vertex v € adj [ u] and v € Q
9.                      do if  w (u, v) < key [v]
10.                             then key [v] ← w (u, v)
11.                             ∏ [v] ← u
```

Single Source Shortest Path Problem

- Given a graph and a source vertex in the graph, find the shortest paths from the source to all vertices in the given graph.
- Following algorithms can be used to solve single source shortest path problem.
  1. Bellman Ford Algorithm
  2. Dijkstra Algorithm

Dijkstra Algorithm

- Dijkastra's algorithm is uded to solve non single source shortest path problem for non-negative weighted edge graph.
- In this algorithm, me make a set of vertices Q. at every step we select a vertices u on the basis of minimum shortest path estimate and apply relax procedure on all vertices adjacent to vertices u.

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```

```
RELAX(u, v, w)
1  if v.d > u.d + w(u, v)
2      v.d = u.d + w(u, v)
3      v.π = u
```

```
INITIALIZE-SINGLE-SOURCE(G, s)
1  for each vertex v ∈ G.V
2      v.d = ∞
3      v.π = NIL
4  s.d = 0
```