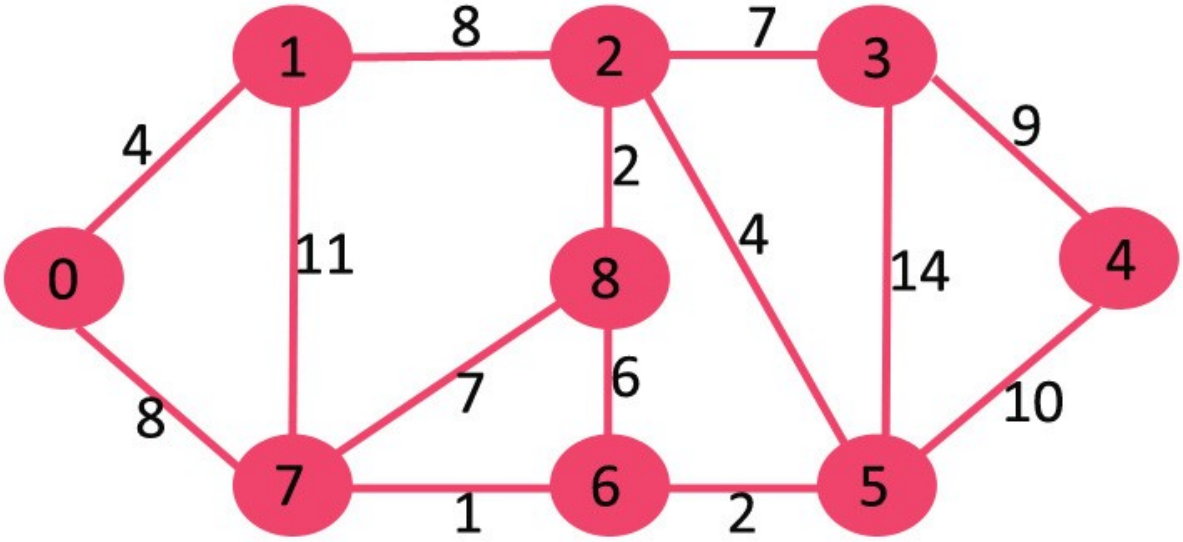
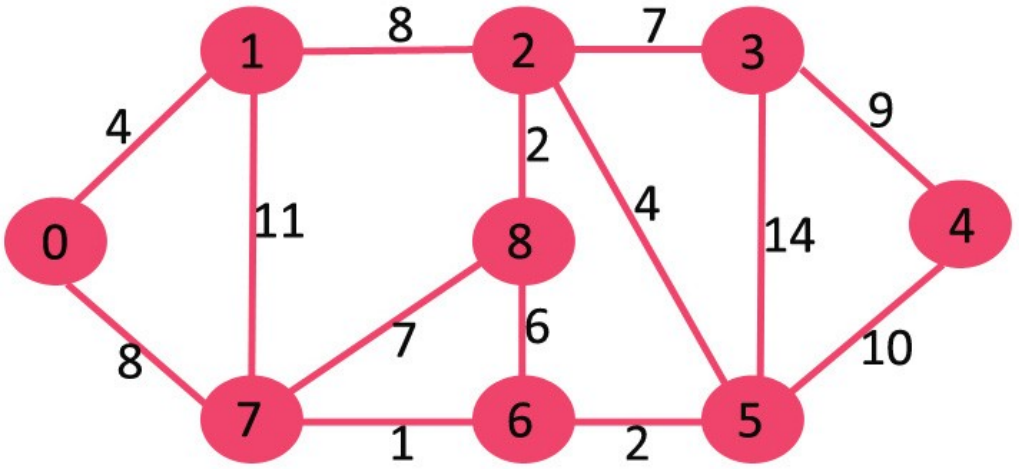
Greedy Approaches (Cont.)

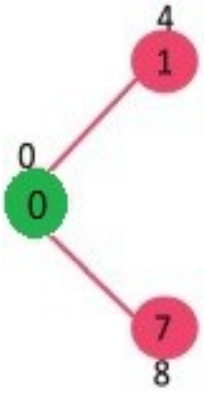
Problem: Shortest Path

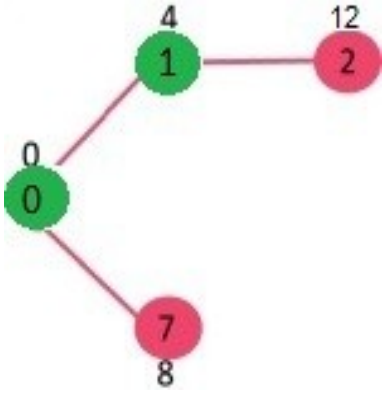


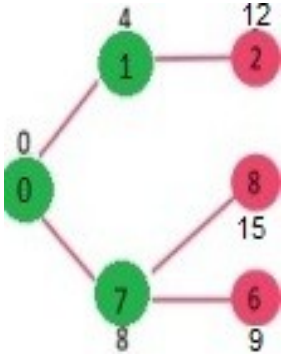
Greedy problem: Given a graph and a source vertex in the graph, find the shortest paths from the source to all vertices in the given graph.

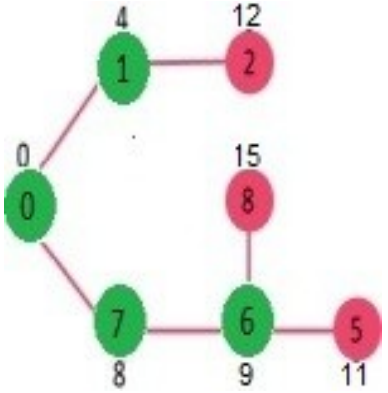
Problem: Dijkstra's Shortest Path





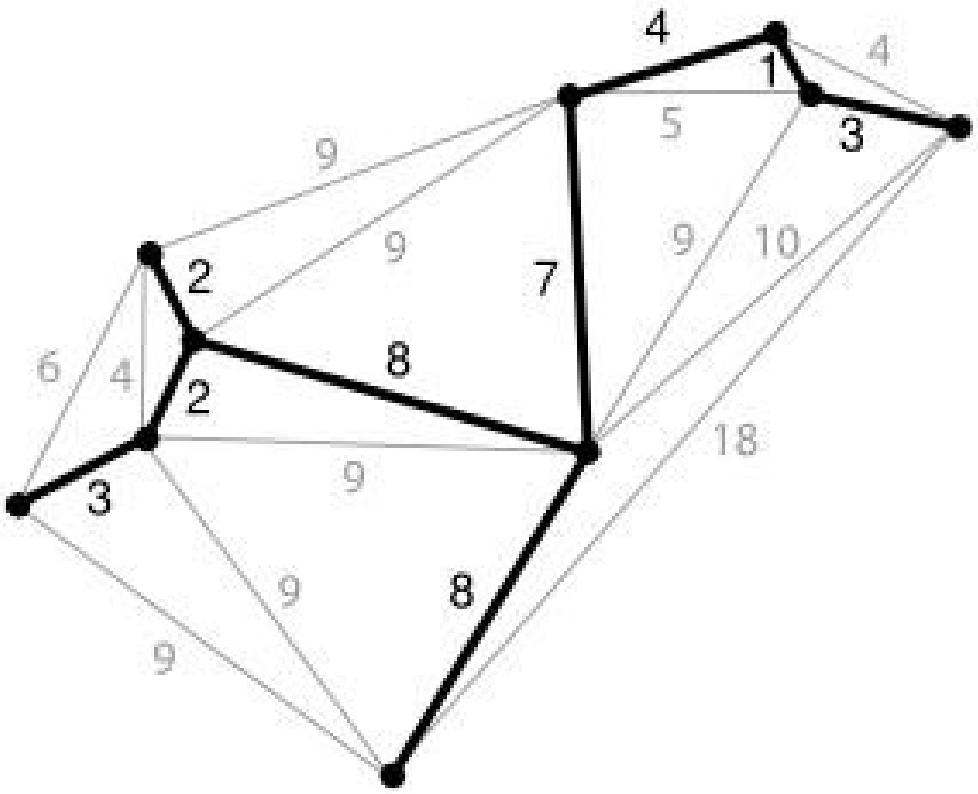






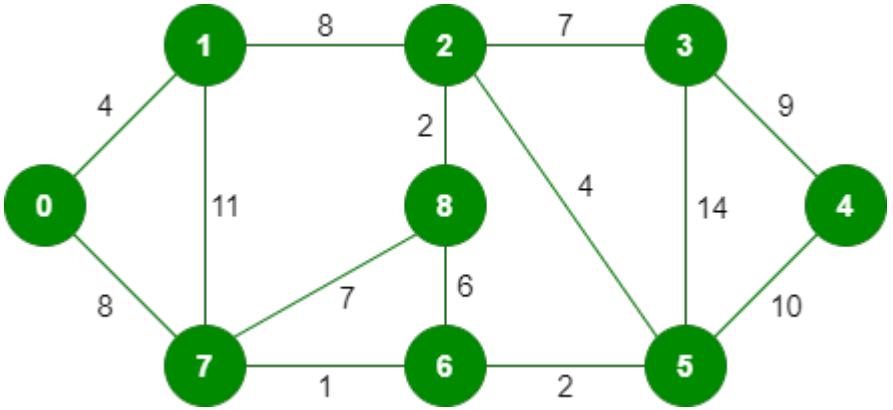
...

Problem: Minimum Spanning Tree



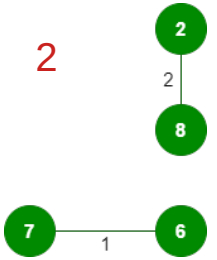
Greedy problem: A Spanning tree is a subset to a connected graph G, where all the edges are connected, i.e, one can traverse to any edge from a particular edge with or without intermediates. Also, a spanning tree must not have any cycle in it. Thus we can say that if there are N vertices in a connected graph then the no. of edges that a spanning tree may have is N-1.

Problem: Kruskal's Minimum Spanning Tree

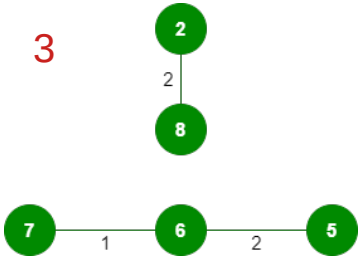


1

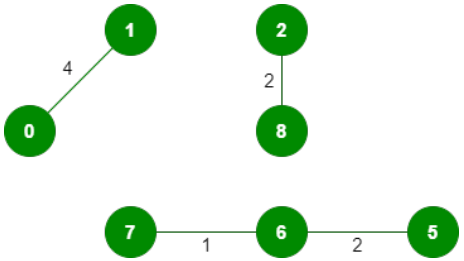
2



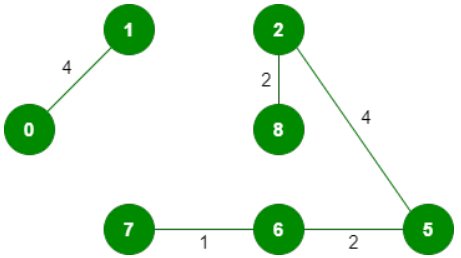
3



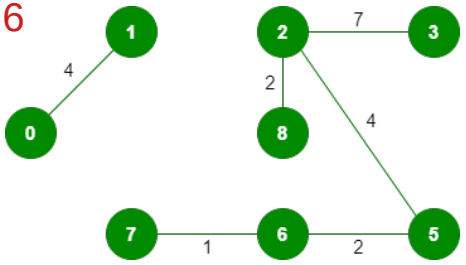
4



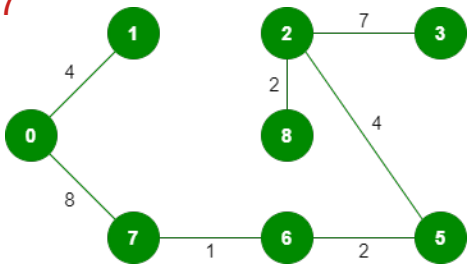
5



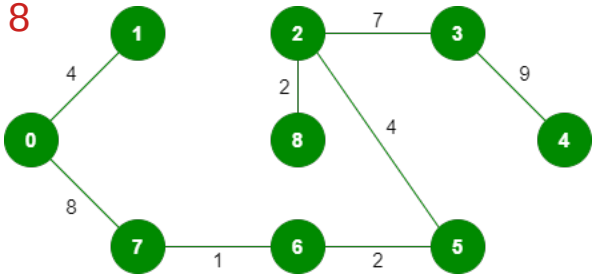
6



7



8



Dynamic Programming

Dynamic Programming is mainly an optimization over plain recursion.

Wherever we see a recursive solution that has repeated calls for same inputs, we can optimize it using Dynamic Programming.

The idea is to simply store the results of subproblems, so that we do not have to re-compute them when needed later.

This simple optimization reduces time complexities from exponential to polynomial.

Problem: 0-1 Knapsack Problem

Greedy problem: Given weights and values of *N* items, put these items in a knapsack of capacity *W* to get the maximum total value in the knapsack. In other words, given two integer arrays *val[0..N-1]* and *wt[0..N-1]* which represent values and weights associated with *N* items respectively. Also given an integer *W* which represents knapsack capacity, find out the maximum value subset of *val[]* such that the sum of the weights of this subset is smaller than or equal to *W*. You cannot break an item, either pick the complete item or don’t pick it (0-1 property)