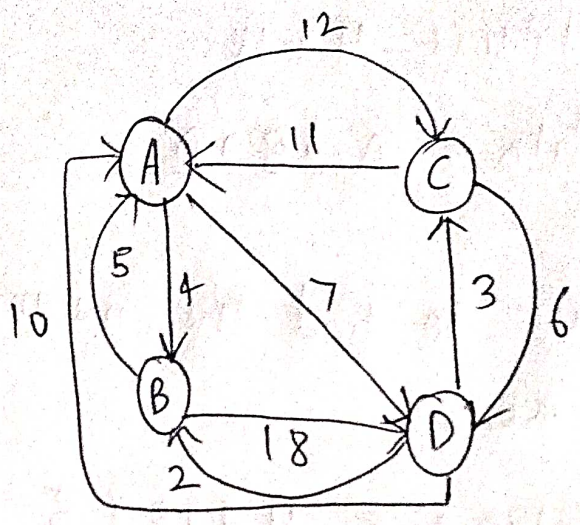
Assignment-1

Qu 1. Solve travelling salesman problem ~~problem~~ (TSP) using greedy method?



Sol. The TSP is an optimization problem that seeks to find the shortest possible route that a salesman can take to visit a given set of cities exactly once and then return to the starting city. The problem is called "travelling salesman" because it is often imagined as a salesman travelling to different cities to sell products.

	A	B	C	D
A	0	(4)	12	7
B	5	0	∞	(18)
C	(1)	∞	0	6
D	10	2	(3)	0

Let us take starting node as A. As the distance from A to B is minimum

So, Distance = 4

Path = A → B

Wisely we will move from B→D as the distance from B to D is minimum

Hence Distance = 4 + 18

Path = A → B → D

Now, we have to find the minimum distance from D to C

Distance = 4 + 18 + 3

Path = A → B → D → C

As all the nodes are covered, so we have to return to the root vertex.

Hence, Total distance travelled

= 4 + 8 + 13 + 11

= 36

So, the path followed is

$$[A \rightarrow B \rightarrow D \rightarrow C \rightarrow A]$$

Qu 2. Demonstrate how knapsack problem can be solved using dynamic programming?

Sol. The knapsack problem is a combinatorial optimization problem that asks, given a set of items, each with a weight and a value, and a knapsack with a maximum weight capacity, what is the maximum value subset of items that can be placed in the knapsack without exceeding its capacity.

One approach to solving the knapsack problem is to use dynamic programming. The idea behind this approach is to build up a table of subproblem and solutions, where each entry in the table represents the maximum value that can be obtained using a subset of the items up to a certain weight limit.

Starting with an empty subset and a weight limit of 0, we can fill out the table row by row, considering each item and its weight and value. For each item, we compare the value of including it in the subset with the value of excluding it. If the item fits within the weight limit, we add its value to the maximum value of the remaining weight limit.

Using this approach, we can find the maximum value subset of items that can be placed in the Knapsack without exceeding its capacity.

Example of 0/1 Knapsack problem using DP.

Weights: {3, 4, 6, 5}

Profits: {2, 3, 1, 4}

The weight of Knapsack is 8 Kg

wi	pi		0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
3	2	1	0	0	0	2	2	2	2	2	2
4	3	2	0	0	0	2	3	3	3	5	5
6	1	3	0	0	0	2	3	3	3	5	5
5	4	4	0	0	0	2	3	4	4	5	6

x_1 x_2 x_3 x_4

{ 1 0 0 1 }

Maximum Profit = 6