BCS 312/ CSC 312

Greedy Algorithms / Techniques.

- A greedy algorithm always make the choice that looks best at the moment.
- It deals with optimization-problems

Definition

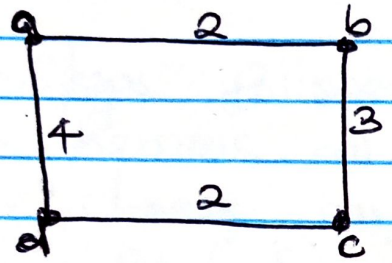
- Greedy Techniques involves constructing a solution through sequence of steps, expanding partially constructed solution obtained so far, until the complete solution for the problem is reached.
- Each step should have 3 properties

1. Feasible:
   The sub-problem should satisfy the Imposed constraints.
2. Locally optimal:-
   Among the Feasible solution for the problem choose the best solution i.e optimal solution.
3. Irrevocable:-
   Once the sub-problem is solved or a step is taken it should remain Unchanged.

Problems:-

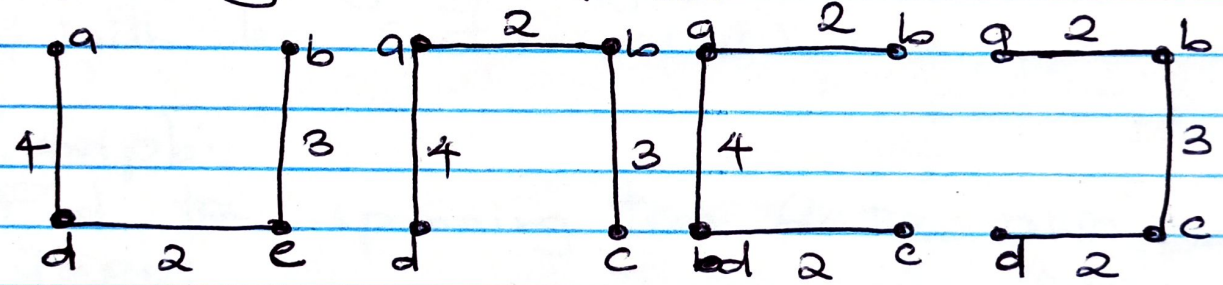
1) Minimum spanning Tree (MST)
2) Knapsack problem
3) Coin change problem
4) Single Source-shortest path (SSSP).

1. Spanning Tree(s)
- is acyclic graph which has span at all nodes
- Example Draw the spanning trees for the following Graph.



Graph (G)

Spanning Trees of (G)



(a) (b) (c) (d)

Cost for each weighted spanning trees

(a) $4 + 2 + 3 = 9$ units
(b) $4 + 2 + 3 = 9$ units
(c) $2 + 4 + 2 = 8$ units
(d) $2 + 3 + 2 = 7$ units.

Therefore the spanning Tree with minimum Cost (MST) for Graph(G) is spanning Tree (d) with the least cost
$(a - b) + (b - c) + (c - d) = 2 + 3 + 2 = \underline{7}$
To solve minimum spanning Tree (MST) we can use two Algorithms
1. Prim's Algorithm
2. Kruskal's Algorithm.

Prim's Algorithm (Nearest neighbour first)

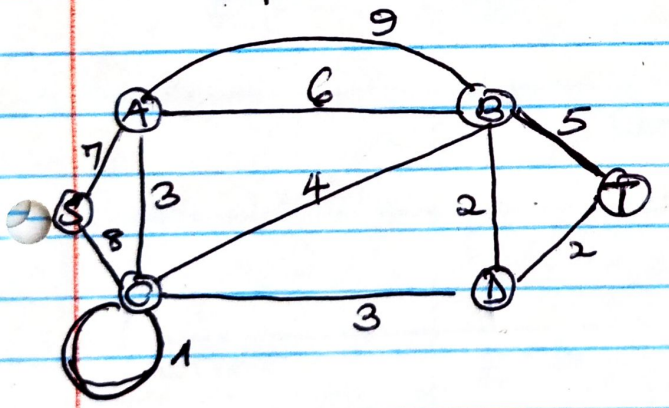
- It is a greedy algorithm that finds a MST for a weight undirected graph.
- In this method we start with reference node (arbitrarily selected).

Steps followed:

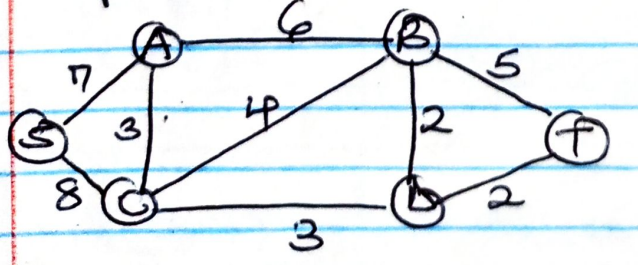
1. Remove all loops and parallel edges
2. Choose any arbitrary node as root node as (S)
3. Check outgoing edges and select the one with less cost (nearest).

Example:

Find the spanning tree for the following Graph.



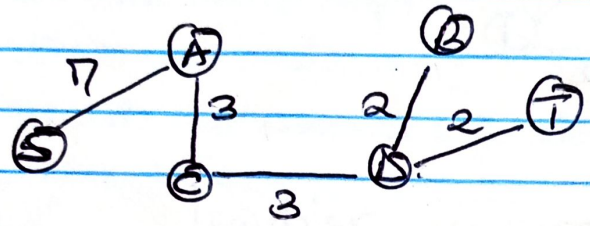
step1: Remove loops & parallel edges.



step2: arbitrary node root (S) node.

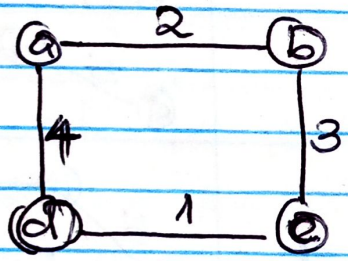
step3: S-A and S-C is (7, 8) - choose (S, A) = 7

Move to node A and check outgoing nodes A, B=(6) and (A, C) = 3 choose (A, C) = 3 move to C and continue to get.



$= 7 + 3 + 3 + 2 + 2 = \underline{17 \text{ units}}$

~~Find the~~ Construct a MST for the Graph.



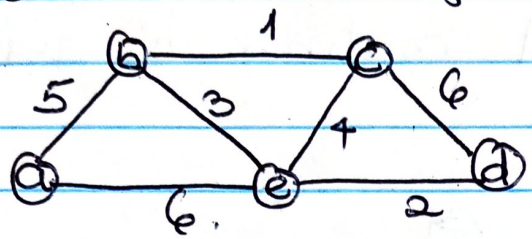
Soln:

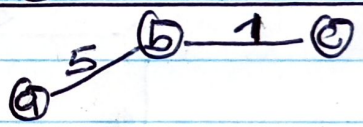
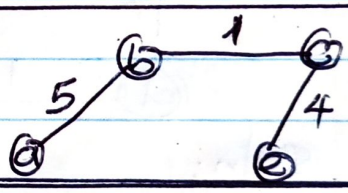
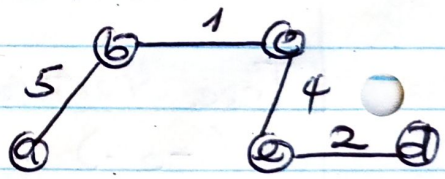
Tree vertices	Remaining vertices	Spanning Tree
a(-,-)	b(a, 2), c(a, ∞), d(a, 4)	a
shortest b(a, 2)	c(b, 3), d(b, ∞)	a 2 b
shortest c(b, 3)	d(c, 1)	a 2 b 3 c
d(c, 1)	NILL	a 2 b 3 c 1 d

Cost 2+3+1 = 6 units

The cost of the minimum edges = 6.

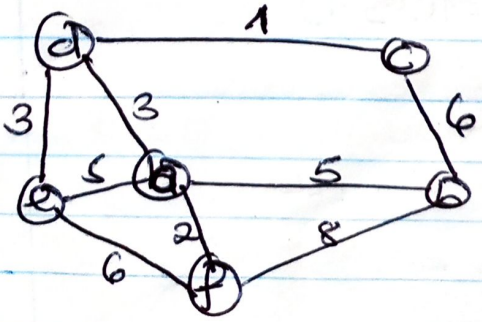
Example 2:
Construct MST for.



	Tree vertices	Remaining vertices	Spanning Tree
1.	Initial a(-,-)	b(a, 5) c(a, ∞), d(a, ∞) e(a, 6)	
2.	nearest b(a, 5)	c(b, 1), e(b, 3) d(b, ∞)	
3	nearest c(b, 1)	d(c, 6) e(c, 4)	
4	nearest e(c, 4)	d(e, 2)	
5	d(e, 2)	NIL	

Min (cost) = (a, b, c, e, d) = 5 + 1 + 4 + 2 = 12 unit

Find the MST for.



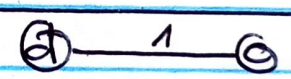
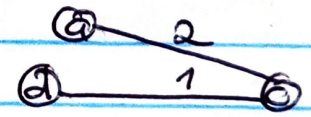
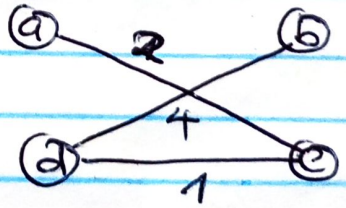
2. Kruskal's Algorithm:

- Find the min spanning Tree using the greedy approach.

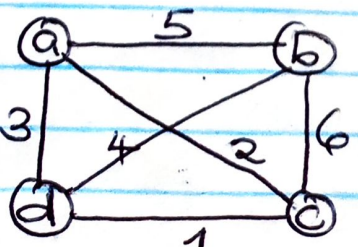
Steps

1. Remove all loops and parallel edges.
2. arrange all edges in order of weights
3. add the edge(s) which has the least weightage
   (in each case check wether it forms a loop (cyclic) or not (acyclic).
4. If acyclic skip the edge and repeat step 3. - stop when all nodes are represented.

Example;

Tree edge	Acyclic / cyclic	Spanning Tree.
dc	Acyclic	
ac	Acyclic	
ad	cyclic	*
bd	Acyclic	
Cost	MST ₹ = 7	

Graph.



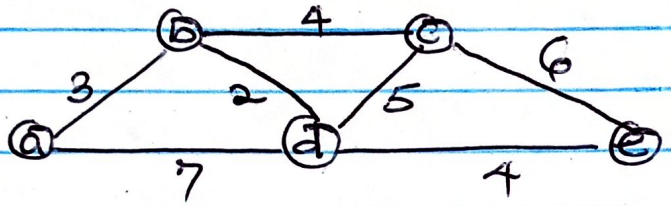
dc = 1
ac = 2
ad = 3
db = 4
ab = 5
bc = 6

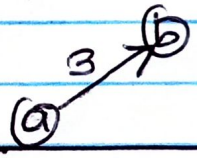
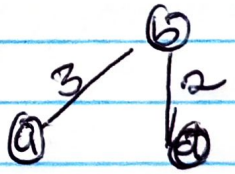
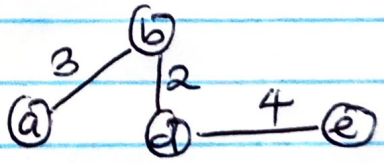
Single source shortest Path (SSSP)
- Dijkstra's Algorithm (Dij-K-stras)

- Used on link-starts routing algorithm
- It is based on nearest neighbours.

Example:-

Using node (a) as the source node conduct SSSP of the following Graph using Dijkstra's Algorithm.



Tree vertices	Remaining vertices From(a)	shortest Path.
Initial a(-, 0)	b(a, 3) c(a, ∞) d(a, 7) e(a, ∞)	(a)
nearest b(a, 3)	c(b, 3+4) d(b, 3+2) e(b, ∞)	
nearest d(b, 5)	c(d, 5+5) e(d, 5+4)	
e(d, 9)	NIL	

The shortest path is a, b, d, e = 3+2+4=9

Q. Explain the differences btn prim's and Kruskal's algorithms. Link this to efficiency analysis.

Fractional Knapsack problem.

- Let n be number of objects and each object is having a weight and contribution to profit.
- The Knapsack of capacity M or W is given. The objective is to fill the knapsack in a such a way that profit shall be maximum.
- We allow a fraction of item to be added to the Knapsack.

Mathematical we can write

$$\text{Maximum (profit)} \sum_{i=1}^{n} p_i x_i$$

$$\text{Subject to} \sum_{i=1}^{n} w_i x_i \leq M(W).$$

Where $1 \leq i \leq n$ and $0 \leq x_i \leq 1$

p_i and w_i are the profit and weight of i^{th} object and x_i is the fraction of j^{th} object to be selected or to be added.

Example:

Given $n = 3$

Profit $(P_1, P_2, P_3) = \{25, 24, 15\}$

Weight $(W_1, W_2, W_3) = \{18, 15, 10\}$ and

$M = 20$

Let us understand the problem:-

- we have 3 objects or item.
- Each object / item has profit and associated weights
- problem is to fill a bag or container

With capacity M = 20, with the objects.
- The filling should be done in a such a way that the profit is maximum or optimized.

Soln.

Some of the feasible solutions are.

Soln no	x_1	x_2	x_3	$\sum w_i x_i$	$\sum p_i x_i$
1	1	2/15	0	20	28.2
2	0	2/3	1	20	31.0
3	0	1	1/2	20	31.5

These solns are obtained by different greedy strategies

Greedy Strategy 1:

1. Items are arranged by their profit values
2. Item with maximum profit is selected first
3. If the weight of the object is less than the remaining capacity of the knapsack, then the object is selected full and the profit associated with the object is added to the total profit.
4. Otherwise - a fraction of the object is selected so that the knapsack can be filled exactly.
5. This process continues i.e selecting the object with highest profit to the lowest profitable object till the knapsack is exactly full.

Greedy II (2)

- Items are arranged by their weights
- Item with minimum weight is selected first.
- The process continues like greedy I till the Knapsack is exactly full.

Greedy III (3).

- Items are arranged by profit/weight ratio
- Item with maximum or highest ~~profit~~ profit/weight ration is selected first
- Process continues like greedy strategy I (1) till the Knapsack is exactly full.

NB

n- Items

Weight = $\{w_1, w_2, \ldots, w_n\}$

Price/profit/value $\{P_1, P_2, \ldots, P_n\}$

W = Capacity.

feasible condition : $w_1 + w_2 + \ldots \leq W$

optimal solution : maximum value or profit in Knapsack

Example 1:

n = 3, Profit = {25, 15, 24}, W.

Weight = {18, 10, 14}, W = 25

Using Greedy technique.

(1) Find the ratio of Profit/weight

(2) Arrange the item in order of Pi/wi ratio (highest to lowest).

Items	profit	weight	Pi/wi	Remaining capacity	Fractional of item	Profit in Knapsack
3	24	14	24/14 = 1.71	25 - 14 = 11	1	0+(1x24) = 24
2	15	10	15/10 = 1.5	11 - 10 = 1	1	24+(1x15) = 39
1	25	18	25/18 = 1.38	1 - 1 = 0	1/8	39+(1/8x25) = 40.38

The last item/object only contribute 1 weight hence 1/8 of its portion by weight is used and its corresponding profit is Calculated or Computed.

Example 2.

n = 4, weight = {7, 3, 4, 5}

profit = {42, 12, 40, 25} W = 12

Solution

Item	P	W	Pi/wi	
1	42	7	6	(ii)
2	12	3	4	(iv)
3	40	4	10	(i)
4	25	5	5	(iii)

Item	Profit	weight	Pi/wi	Rem Capacity	Fraction	Profit in knapsack
3	40	4	10	12-4 = 8	1	0+40 = 40
1	42	7	6	8-7 = 1	1	40+42 = 82
4	25	5	5	1-1	1/5	82+(1/5x25) = 82+5 = 87
2	12	3	4	0		

Profit(max) = 87