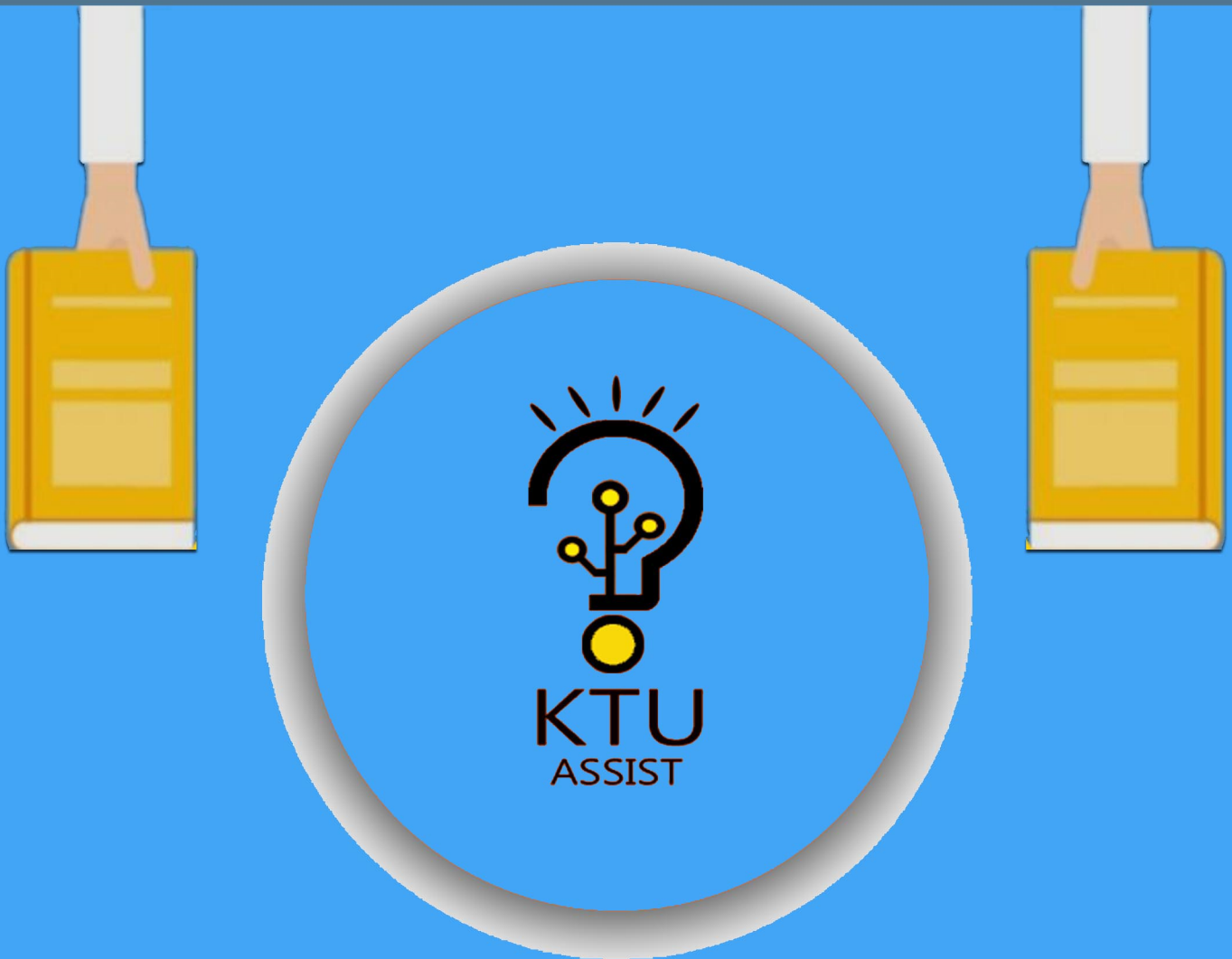


APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

STUDY MATERIALS



a complete app for ktu students

Get it on Google Play

www.ktuassist.in

Module 5

Greedy Strategy

GREEDY STRATEGY

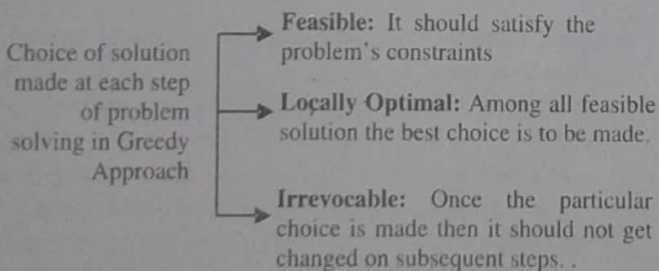
Ques 1) Discuss the greedy strategy with example.

Ans: Greedy Strategy

A greedy algorithm is any algorithm that follows the problem solving met heuristic of making the locally optimal choice at each stage with the hope of finding the global optimum.

Greedy method is a powerful technique used in the design of algorithms. Almost all problems that come under this category have n inputs. A subset of the given set of inputs that satisfies some given constraints is to be obtained. Such a subset is called a **feasible solution**. Now the problem is to find a feasible solution that maximizes or minimizes a given objective function. Such a function is called an **optimal solution**. The Greedy method works in stages.

For example, applying the greedy strategy to the traveling salesman problem yields the following algorithm: "At each stage visit the unvisited city nearest to the current city". It is similar to dynamic programming. Used for optimization problems.



Greedy algorithms can be characterized as being 'short sighted', and as 'non-recoverable'. They are ideal only for problems which have 'optimal substructure'. Despite this, greedy algorithms are best suited for simple problems (e.g. giving change). It is important, however, to note that the greedy algorithm can be used as a selection algorithm to prioritize options within a search, or branch and bound algorithm.

One way to construct a solution for such optimization problems is the greedy method, in which we construct the solution in stages. At each stage we make a decision that appears to be the best at that time, according to certain greedy criterion, and will not be changed in later stages.

Hence, each decision should assume the feasibility. Although the greedy method does not always lead to an optimal solution, e.g., in the traveling salesman problem, it does in some other cases.

For example, given a set S of n items, with each item i having

$$\begin{aligned} p_i &= \text{profit on } i^{\text{th}} \text{ item} \\ w_i &= \text{weight of } i^{\text{th}} \text{ item} \\ x_i &= \text{amount of } i^{\text{th}} \text{ item to maximize profit} \end{aligned}$$

Choose the items with maximum total profit but with weight $\leq W$.

Items:	1	2	3	4	5
Weight (gms):	4	8	2	6	1
Profit (₹):	12	32	40	30	50
Value (₹/gm):	3	4	20	5	50

This problem can be defined as follows:

Objective is to maximize $\sum_{i \in S} p_i x_i$

Subject to the constraint $\sum_{i \in S} x_i \leq W$

We need $W \leq 10$ gms with maximum profit. 5th item has maximum profit (₹ 50): select 5th item and its weight is 1gm ($W = 1$ gm). The subsequent phases are shown below, taking maximum profit with total weight restriction.

Table 5.1: Greedy Method Example

Item (i)	Profit (p _i)	Total Profit	Weight (w _i)	Total Weight (W)
5	50	50	1	1
3	40	90	2	3
4	30	120	6	9
2	32	152	1	10

The initial partial solution is item 5 with weight 1gm, having a profit of ₹ 50, subject to a total weight of 10gms. The remaining partial solutions with their sub-problems are shown in the above table.

Ques 2) What are the five pillars of greedy algorithms?

Ans: Five Pillars of Greedy Algorithms

- 1) **Candidate Set:** From which a solution is created.
- 2) **Selection Function:** It chooses the best candidate to be added to the solution.

- 3) **Feasibility Function:** It is used to determine if a candidate can be used to contribute to a solution.
- 4) **Objective Function:** It assigns a value to a solution, or a partial solution.
- 5) **Solution Function:** It will indicate when we have discovered a complete solution.

Ques 3) What is the control abstraction of greedy algorithm?

Ans: Control Abstraction of Greedy Algorithm

The greedy method suggests that one can devise an algorithm that works in stages, considering one input at a time. At each stage, a decision is made regarding whether a particular input is in an optimal solution. This is done by considering the inputs in an order determined by some selection procedure. If the inclusion of the next input into the partially constructed optimal solution will result in an infeasible solution, then this input is not added to the partial solution. Otherwise, it is added. The selection procedure itself is based on some optimization measure. This measure may be the objective function. In fact, several different optimization measures may be plausible for a given problem. Most of these, however, will result in algorithms that generate suboptimal solutions. This version of the greedy technique is called the **subset paradigm**.

One can describe the subset paradigm abstractly by considering the control abstraction in following algorithm.

Algorithm: Greedy Method Control Abstraction for the Subset Paradigm

Algorithm Greedy(a, n)

//a[1:n] contains the n inputs

```
{
  solution:  $\emptyset$ ; //Initialise the solution
  for i:= 1 to n do
  {
    x : Select(a);
    If Feasible (solution, x) then
      solution:= Union(solution, x);
  }
  return solution;}
```

The function 'Select' selects an input from a[] and removes it. The selected input's value is assigned to x. 'Feasible' is a Boolean-valued function that determines whether x can be included into the solution vector. The function 'Union' combines x with the solution and updates the objective function. The function 'Greedy' describes the essential way that a greedy algorithm will look, once a particular problem is chosen and the functions 'Select', 'Feasible' and 'Union' are properly implemented.

For problems that does not call for the selection of an optimal subset, in the greedy method one make decisions by considering the inputs in some order. Each decision is made using an optimization criterion that can be computed using decisions already made. This version of the greedy method is called the 'ordering paradigm'.

Ques 4) Write the steps of designing greedy algorithms?

Ans: Design of Greedy Algorithms

The simplified design of greedy algorithm is done through following steps:

- Step 1)** Cast the optimization problem as one in which we make a choice and are left with one sub-problem to solve.
- Step 2)** Prove that there's always an optimal solution that makes the greedy choice, so that the greedy choice is always safe.
- Step 3)** Show that greedy choice and optimal solution to sub-problem \Rightarrow optimal solution to the problem.

Ques 5) What are the main properties of greedy algorithms?

Ans: Properties of Greedy Algorithms

The greedy algorithm is said to be optimal because of two key ingredients given below. Greedy algorithms produce good solutions on some mathematical problems, but not on others. Most problems for which they work well have two properties:

- 1) **Greedy Choice Property:** The first key ingredient is the greedy-choice property: a globally optimal solution can be arrived at by making a locally optimal (greedy) choice. In other words, when we are considering which choice to make, we make the choice that looks best in the current problem, without considering results from sub-problems

One can make whatever choice seems best at the moment and then solve the subproblems that arise later. The choice made by a greedy algorithm may depend on choices made so far but not on future choices or all the solutions to the sub-problem. It iteratively makes one greedy choice after another, reducing each given problem into a smaller one.

- 2) **Optimal Substructure:** "A problem exhibits optimal substructure if an optimal solution to the problem contains optimal solutions to the sub-problems". A problem has optimal substructure if the best next move always leads to the optimal solution. An example of 'non-optimal substructure' would be a situation where capturing a queen in chess (good next move) will eventually leads to the loss of the game (bad overall move).

Ques 6) Write the algorithm of greedy method.

Ans: Algorithm of Greedy Method

The following is a generic pseudocode for the greedy method:

Algorithm: Greedy (C)

Step 1: $S = \phi$; /* S is the set in which we construct the solution */

Step 2: While not solution (S) and $C \neq \phi$ Do

Step 3: Select $x \in C$ based on the selection criterion;

Step 4: $C \leftarrow C - (x)$;

Step 5: If feasible $(S \cup \{x\})$ Then

Step 6: $S = S \cup \{x\}$;

Step 7: If solution $\{S\}$ Then
Return (S) ;

Step 8: Else
Return ("No solution");

Step 9: End Greedy

It is easy to see why such algorithms are called 'greedy'. At every step, the procedure chooses the best candidate from the remaining, without worrying about the future. Once a candidate is included in the solution, it is there for good; once a candidate is excluded from the solution, it is never re-considered. The selection function is usually based on the objective function; they may even be identical. At times, there may be several possible selection functions; so that one has to choose the right one if we want our algorithm to work properly.

For example, suppose we want to give change to a customer using the smallest possible number of coins. The elements of the problem are as follows: the candidates are a finite set of coins, representing, e.g., 1, 5, 10 and 25 units and containing atleast one coin of each type. The solution is a subset of the coins whose value is the amount we have to pay. A feasible set is the total value of the chosen subset whose value does not exceed the amount to be paid. The selection function is choosing the highest valued coin remaining in the set of candidates and the objective function is the number of coins used in the solution.

Ques 7) What are the main applications of greedy method?

Ans: Applications of Greedy Method

The greedy method technique has many applications such as:

- 1) Knapsack Problem,
- 2) Minimum-Cost Spanning Trees.
- 3) Single-Source Shortest Paths,
- 4) Container Loading,
- 5) Activity Selection Problem
- 6) Topological Sorting,
- 7) Bipartite Cover
- 8) Graph Coloring

Ques 8) Explain the knapsack problem.
Or

Discuss about the Fractional and 0-1 Knapsack Problem.

Ans: Knapsack Problem

A thief robbing a store can carry a maximal weight of w into his knapsack. There are n items and i^{th} item weigh w_i and is worth v_i dollars. What items should the thief take?

There are two versions of problem:

- 1) **Fractional Knapsack Problem:** The setup is same, but the thief can take fractions of items, meaning that

the items can be broken into smaller pieces so that thief may decide to carry only a fraction of x_i of item i , where $0 \leq x_i \leq 1$.

It is same as the 0-1 knapsack problem, but can take fraction of a time. Both have optimal substructure. The fractional knapsack problem has the greedy-choice property, whereas the 0-1 knapsack problem does not possess it.

Exhibit greedy choice property.

Greedy algorithm exists.

Exhibit optimal substructure property.

To solve the fractional problem, rank items by value/weight: v_i/w_i .

Let $v_i/w_i \geq v_{i+1}/w_{i+1}$ for all i .

Algorithm:
KNAPSACKGREEDY (v, w, W)

Step 1: load $\leftarrow 0$

Step 2: $i \leftarrow 1$

Step 3: while load $< W$ and $i \leq n$

Step 4: do if $w_i \leq W - \text{load}$

Step 5: then take all of item i

Step 6: else take $(W - \text{load})/w_i$ of item i

Step 7: add what was taken to load

Step 8: $i \leftarrow i + 1$

$O(n \lg n)$ to sort, $O(n)$ thereafter.

- 2) **0-1 Knapsack Problem:** The setup is the same, but the items may not be broken into smaller pieces, so thief may decide either to take an item or to leave it (binary choice), but may not take a fraction of an item.

The features of **0-1Knapsack Problem** are:

- i) n items.
- ii) Item i is worth \$ v_i , weights w_i pounds.
- iii) Find a most valuable subset of items with total weight $\leq W$.
- iv) Have to either take an item or not take it—can't take part of it.

Exhibit no greedy choice property.

No greedy algorithm exists.

Exhibit optimal substructure property.

Only dynamic programming algorithm exists.

Greedy algorithm doesn't work for the 0-1knapsack problem. Greedy algorithm gets empty space, which lowers the average value per pound of the items taken.

For example, consider the following example:

i	1	2	3
v_i	60	100	120
w_i	10	20	30
v_i/w_i	6	5	4

$W = 50$.

Greedy solution:

- 1) Take items 1 and 2.
- 2) Value = 160, weight = 30.

Have 20 pounds of capacity left over.

Optimal solution:

- 1) Take items 2 and 3.
- 2) Value = 220, weight = 50.

No leftover capacity.

Ques 9) For the following instance calculate knapsack solution where $j = 3$, $m = 30$ and weight are (10, 12, 15) and profit are (20, 28, 22).

Ans: Given, $(w_1, w_2, w_3) = (10, 12, 15)$ and profit $(p_1, p_2, p_3) = (20, 28, 22)$ then

Solution No.	(n_1, n_2, n_3)	$m \geq \sum w_i n_i$	$\sum p_i n_i$
1	$(1, 1, \frac{8}{15})$	30	59.73
2	$(\frac{3}{10}, 1, 1)$	30	56
3	$(1, \frac{1}{4}, 1)$	30	49
4	$(\frac{3}{5}, 1, \frac{2}{5})$	30	48.80

Hence optimal profit = 59.73 and n_i i.e., $(n_1, n_2, n_3) = (1, 1, \frac{8}{15})$.

Ques 10) Consider that there are three items. Weight and profit value of each item is as given below,

i	W_i	P_i
1	18	30
2	15	21
3	10	18

Also $W = 20$. Obtain the solution for the above given Knapsack problem.

Ans: The feasible solutions are as given below:

X_1	X_2	X_3
1/2	1/3	1/4
1	2/15	0
0	2/3	1
0	1	1/2

Let us compute $\sum W_i X_i$

$$1) \frac{1}{2} \times 18 + \frac{1}{3} \times 15 + \frac{1}{4} \times 10 = 16.5$$

$$2) 1 \times 18 + \frac{2}{15} \times 15 + 0 \times 8 = 20$$

$$3) 0 \times 18 + \frac{2}{3} \times 15 + 10 = 20$$

$$4) 0 \times 18 + 1 \times 15 + \frac{1}{2} \times 10 = 20$$

Let us compute $\sum P_i X_i$

$$1) \frac{1}{2} \times 30 + \frac{1}{3} \times 21 + \frac{1}{4} \times 18 = 26.5$$

$$2) 1 \times 30 + \frac{2}{15} \times 21 + 0 \times 18 = 32.8$$

$$3) 0 \times 30 + \frac{2}{3} \times 21 + 18 = 32$$

$$4) 0 \times 30 + 1 \times 21 + \frac{1}{2} \times 18 = 30$$

All this is shown in table below:

$\sum W_i X_i$	$\sum P_i X_i$
16.5	26.5
20	32.8
20	32
20	30

The solution 2 gives the maximum profit and hence it turns out to be optimum solution.

Ques 11) Find an optimal solution to the knapsack instance $n = 7$, $m = 15$

$(p_1, p_2, \dots, p_7) = (10, 5, 15, 7, 6, 18, 3)$ and $(w_1, w_2, \dots, w_7) = (2, 3, 5, 7, 1, 4, 1)$.

Ans: $n \rightarrow$ Number of object
 $m \rightarrow$ Capacity of bag
 $p_i \rightarrow$ Profit related to w_i .

S.No.	x_1	x_2	x_3	x_4	x_5	x_6	x_7	$\sum w_i x_i$	$\sum p_i x_i$
1)	1	1	0	1	1	0	1	$2+3+7+1+1=14$	$10+5+7+6+3=31$
2)	1	0	1	0	1	1	0	$2+5+1+4=12$	$10+15+6+18=49$
3)	1	1	1	0	1	1	0	$2+3+5+1+4=15$	$10+5+15+6+18=54$ This is optimal solution.
4)	1	1	1	0	0	1	1	$2+3+5+4+1=15$	$10+5+15+18+3=51$
5)	0	1	1	1	0	0	0	$3+5+7=15$	$5+15+7=27$
6)	1	0	0	1	1	1	1	$2+7+1+4+1=15$	$10+7+6+18+3=44$

Solution 3 gives the optimal solution.

Ques 12) Consider the following instance of the knapsack problem $n = 3$, $m = 20$, $(p_1, p_2, p_3) = (25, 24, 15)$ and $(w_1, w_2, w_3) = (18, 15, 10)$.

Ans: $n \rightarrow$ Number of object; $m \rightarrow$ capacity of bag; $p_i \rightarrow$ profit related to w_i .

It is fractional knapsack problem.

S.No.	x_1	x_2	x_3	$\Sigma w_i x_i$	$\Sigma p_i x_i$
1)	1	$\frac{2}{15}$	0	$18 + \left(\frac{2}{15} \times 15\right) = 18 + 2 = 20$	$25 + \left(\frac{2}{15} \times 24\right) = 25 + 3.2 = 28.2$
2)	0	1	$\frac{1}{2}$	$15 + \left(10 \times \frac{1}{2}\right) = 15 + 5 = 20$	$24 + 7.5 = 31.5$ This is optimal solution.
3)	0	$\frac{2}{3}$	1	$\left(15 \times \frac{2}{3}\right) + 10 = 10 + 10 = 20$	$16 + 15 = 31$

Ques 13) What is difference between greedy technique and dynamic programming?

Ans: Difference between Greedy Technique and Dynamic Programming

Table 5.2 describes the difference between Greedy method and dynamic programming:

Table 5.2

Greedy Technique	Dynamic Programming
Greedy method is used for obtaining optimum solution.	Dynamic programming is also for obtaining optimum solution.
In Greedy method, a set of feasible solutions and the picks-up the optimum solution.	There is no special set of feasible solutions in this method.
In Greedy method, the optimum selection is without revising previously generated solutions.	Dynamic programming considers all possible sequences in order to obtain the optimum solution.
In Greedy method, there is no as such guarantee of getting optimum solution.	It is guaranteed that the dynamic programming will generate optimal solution using principle of optimality.

MINIMUM COST SPANNING TREE COMPUTATION

Ques 14) What algorithms are used for finding minimal spanning trees? List them.

Ans: Algorithms for Finding Minimal Spanning Trees

There are basically two algorithms for solving the minimum spanning tree problem:

- 1) Prim's Algorithm
- 2) Kruskal's Algorithm

Ques 15) Explain the Prim's algorithm to find the minimum spanning tree with the help of an example.

Or

Discuss the Prim's algorithm.

Ans: PRIM's Algorithm

One of the classic algorithms for this problem is that found by **Robert C. Prim**. It's a greedy style algorithm and it's guaranteed to produce a correct result. At any point in the algorithm, one can see that he/she has a set of vertices that have already been included in the tree; the rest of the vertices have not.

The algorithm then finds, at each stage, a new vertex to add to the tree by choosing the edge (u, v) such that the cost of (u, v) is the smallest among all edges where u is in the tree and v is not.

Steps for Prim's Method

Step 1: Choose any vertex V_1 of G .

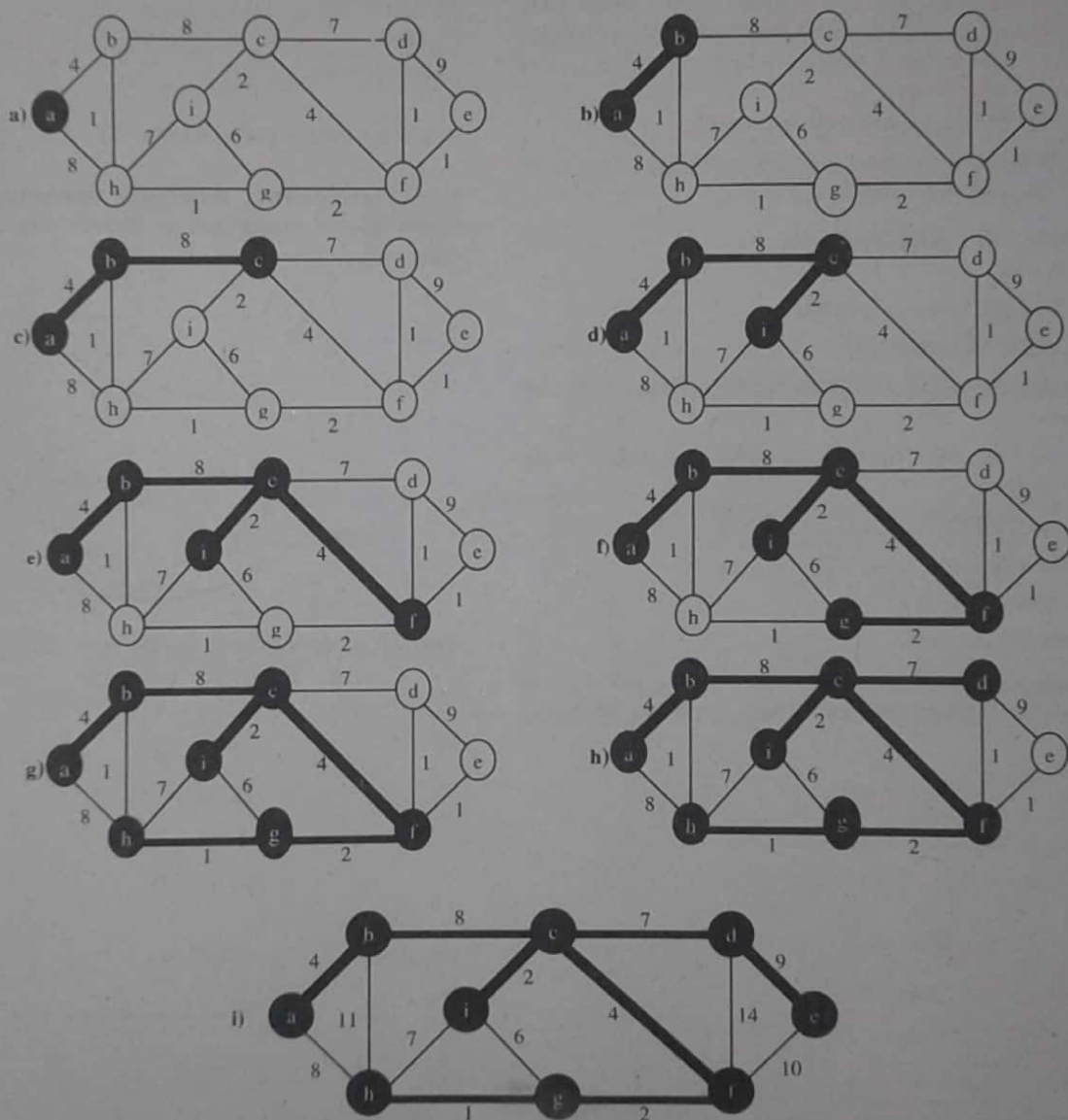
Step 2: Choose an edge $e_1 = V_1, V_2$ of G such that $V_2 \neq V_1$ and e_1 has the smallest weight among the edges of G incident with V_1 .

Step 3: If edges e_1, e_2, \dots, e_i have been chosen involving end points v_1, v_2, \dots, v_{i+1} , choose an edge $e_{i+1} = V_j V_k$ with $V_j \in \{V_1, \dots, v_{i+1}\}$, and $V_k \notin \{v_1, \dots, v_{i+1}\}$ such that e_{i+1} has smallest weight among the edges of G with precisely one end in $\{v_1, v_2, \dots, v_{i+1}\}$.

Step 4: Stop after $n-1$ edges have been chosen otherwise go to step 3.

Algorithm**MST-PRIM** (G, w, r)**Step 1:** for each $u \in V[G]$ **Step 2:** do $\text{key}[u] \leftarrow \infty$ **Step 3:** $\pi[u] \leftarrow \text{NIL}$ **Step 4:** $\text{key}[r] \leftarrow 0$ **Step 5:** $Q \leftarrow V[G]$ **Step 6:** while $Q \neq \emptyset$ **Step 7:** do $u \leftarrow \text{EXTRACT-MIN}(Q)$ **Step 8:** for each $v \in \text{adj}[u]$ **Step 9:** do if $v \in Q$ and $w(u, v) < \text{key}[v]$ **Step 10:** then $\pi[v] \leftarrow u$ **Step 11:** $\text{key}[v] \leftarrow w(u, v)$

For example, Prim's algorithm is described by the figures below:



Explanation: The execution of Prim's algorithm on the graph. The root vertex is a. Shaded edges are in the tree being grown, and the vertices in the tree are shown in black. At each step of the algorithm, the vertices in the tree determine a cut of the graph, and a light edge crossing the cut is added to the tree.

In the second step, for example, the algorithm has a choice of adding either edge (b, c) or edge (a, h) to the since both are light edges crossing the cut.

Ques 16) Describe the Kruskal's algorithm to find the minimum spanning tree.

Or

Discuss the Kruskal's algorithm.

Ans: Kruskal's Algorithm

In this approach Minimum cost spanning tree T is built edge by edge. Edges are considered for inclusion in T in ascending order. An edge is included in T if it doesn't form a cycle with edges already in T . Since G is connected and has $n > 0$ vertices exactly $(n - 1)$ edges will be selected for inclusion in T .

Steps for Kruskal's Method

Step 1) Choose an edge e_1 of the graph G , such that the weight of e_1 i.e. $w(e_1)$ is as small as possible and e_1 is not a loop.

Step 2) If edges e_1, e_2, \dots, e_i have been chosen then choose an edge e_{i+1} not already chosen, such that
i) the induced subgraph $G[[e_1, \dots, e_{i+1}]]$ is acyclic and
ii) $w(e_{i+1})$ is as small as possible.

Step 3) If G has n vertices, stop after $n-1$ edges have been chosen, otherwise repeat step 2.

Algorithm: MST-KRUSKAL (G, w)

Step 1: $A \leftarrow \emptyset$

Step 2: for each vertex $v \in V[G]$

Step 3: do MAKE-SET (v)

Step 4: sort the edges of E into non-decreasing order by weight w

Step 5: for each edge $(u, v) \in E$, taken in non-decreasing order by weight

Step 6: do if FIND-SET (u) \neq FIND-SET (v)

Step 7: then $A \leftarrow A \cup \{(u, v)\}$

Step 8: UNION (u, v)

Step 9: return A

For example, consider the following figure 5.1 to understand the procedure for finding minimum spanning tree:

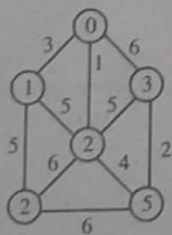


Figure 5.1

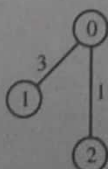
Step-1

No. of Nodes	0	1	2	3	4	5
Distance	0	3	1	6	∞	∞
Distance from		0	0	0		



Step-2

No. of Nodes	0	1	2	3	4	5
Distance	0	3	0	5	6	4



Distance from		0	2	2	2
---------------	--	---	---	---	---

Step-3

No. of Nodes	0	1	2	3	4	5
Distance	0	0	0	5	3	4
Distance from				2	1	2

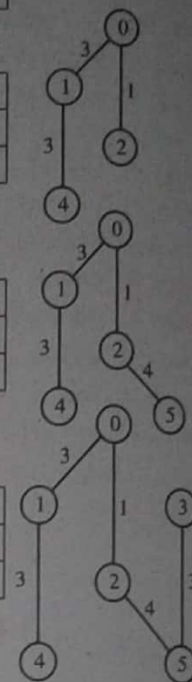
Step-4

No. of Nodes	0	1	2	3	4	5
Distance	0	0	0	5	0	4
Distance from				2		2

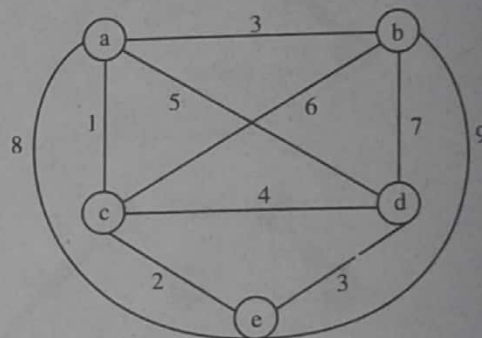
Step-5

No. of Nodes	0	1	2	3	4	5
Distance	0	3	0	5	6	4
Distance from		0		2	2	2

Minimum cost = $1+2+3+3+4 = 13$



Ques 17) Construct minimum spanning tree for the below given graph using Prim's algorithm (Source node = a).



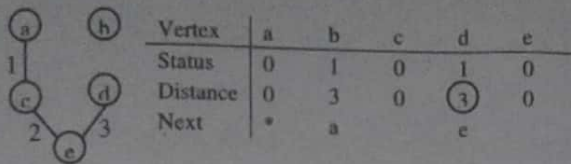
Ans: The cost adjacency matrix is,

	a	b	c	d	e
a	0	3	1	5	8
b	3	0	6	7	9
c	1	6	0	4	2
d	5	7	4	0	3
e	8	9	2	3	0

The prim's algo is as follows:

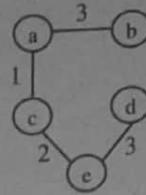
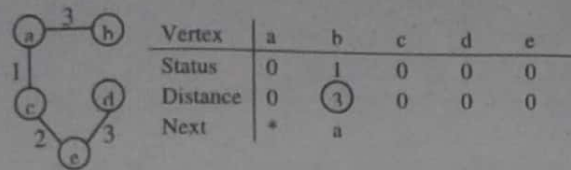
Vertex	a	b	c	d	e
Status	0	1	1	1	1
Distance	0	3	1	5	8
Next	*	a	a	a	a

Vertex	a	b	c	d	e
Status	0	1	0	1	1
Distance	0	3	0	4	2
Next	*	c	0	c	c



Minimum cost = $1 + 2 + 3 + 3 = 9$

Minimum spanning tree



Ques 18) Construct the minimal spanning tree for the graph shown below:

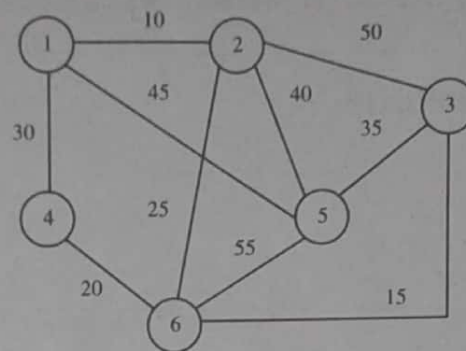


Figure 5.2

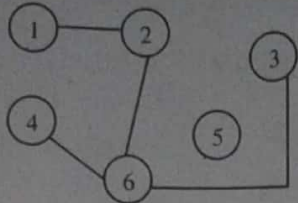
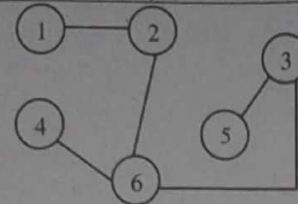
Ans: Arrange all the edges in the increasing order of their costs:

Cost	10	15	20	25	30	35	40	45	50	55
Edge	(1, 2)	(3, 6)	(4, 6)	(2, 6)	(1, 4)	(3, 5)	(2, 5)	(1, 5)	(2, 3)	(5, 6)

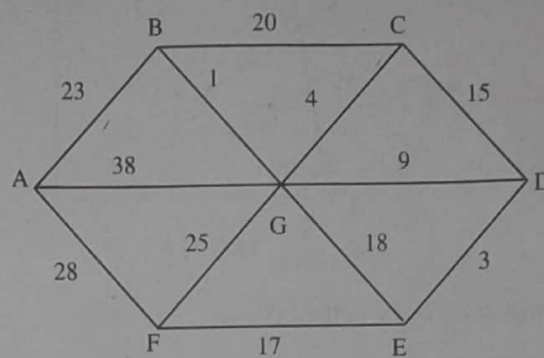
The stages in Kruskal's algorithm for minimal spanning tree are as follows:

Table 5.3

Edge	Cost	Stages in Kruskal's Algorithm	Remarks
(1, 2)	10		The edge between vertices 1 and 2 is the first edge selected. It is included in the spanning tree.
(3, 6)	15		Next, the edge between vertices 3 and 6 is selected and included in the tree.
(4, 6)	20		The edge between vertices 4 and 6 is next included in the tree.

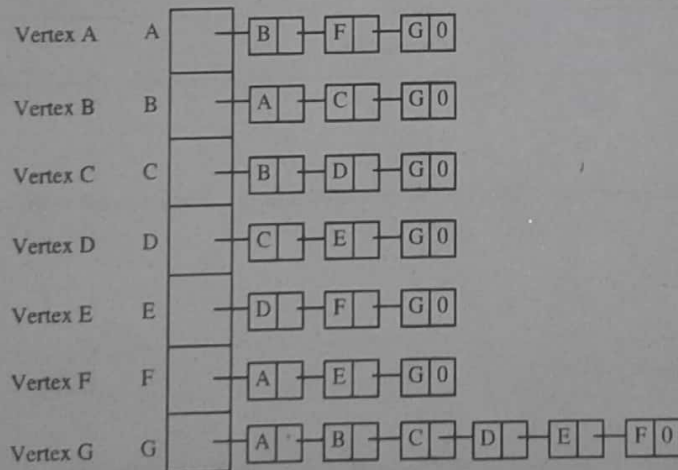
(2, 6)	25		The edge between vertices 2 and 6 considered next and included in the tree.
(1, 4)	30	Reject	The edge between the vertices 1 and 4 is discarded as its inclusion creates a cycle.
(3, 5)	35		Finally, the edge between vertices 3 and 5 is considered and included in the tree built. This completes the tree. The cost of the minimal spanning tree is 105.

Ques 19) Consider the following undirected graph:



- Find the adjacency list representation of the graph.
- Find a minimum cost spanning tree by Kruskal's algorithm.

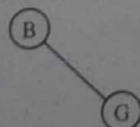
Ans: (a) Adjacency List Representation



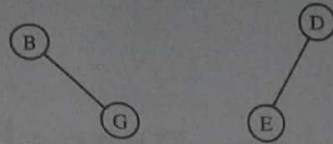
(b) Arrange all edges in the increasing order of their costs.

Cost	1	3	4	9	15	17	18	20	23	25	28	38
Edge	(B,G)	(D,E)	(C,G)	(D,G)	(C,D)	(E,F)	(E,G)	(B,C)	(A,B)	(F,G)	(A,F)	(A,G)

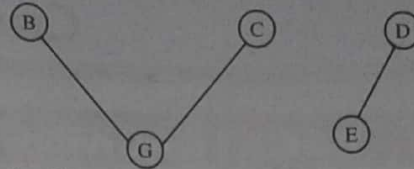
The edge between vertices B and G is first edge selected. It is included in spanning tree.



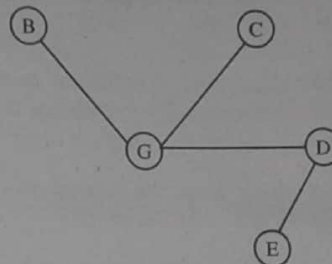
Next the edge between D, E is selected and add in spanning tree.



Next the edge between C and G is selected and added in spanning tree.

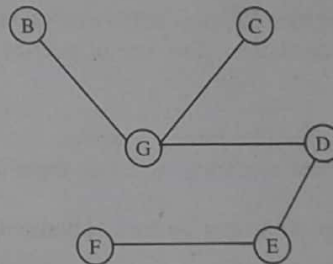


Next the edge between D and G is selected and add in spanning tree.

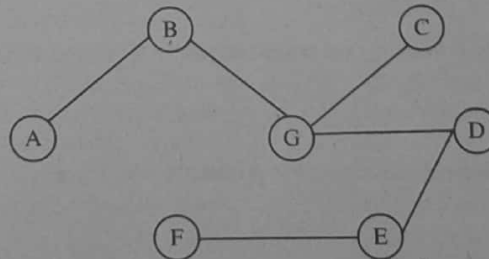


Next the edge between C and D is discarded as its inclusion creates a cycle.

Next the edge between E and F is selected and add in spanning tree.



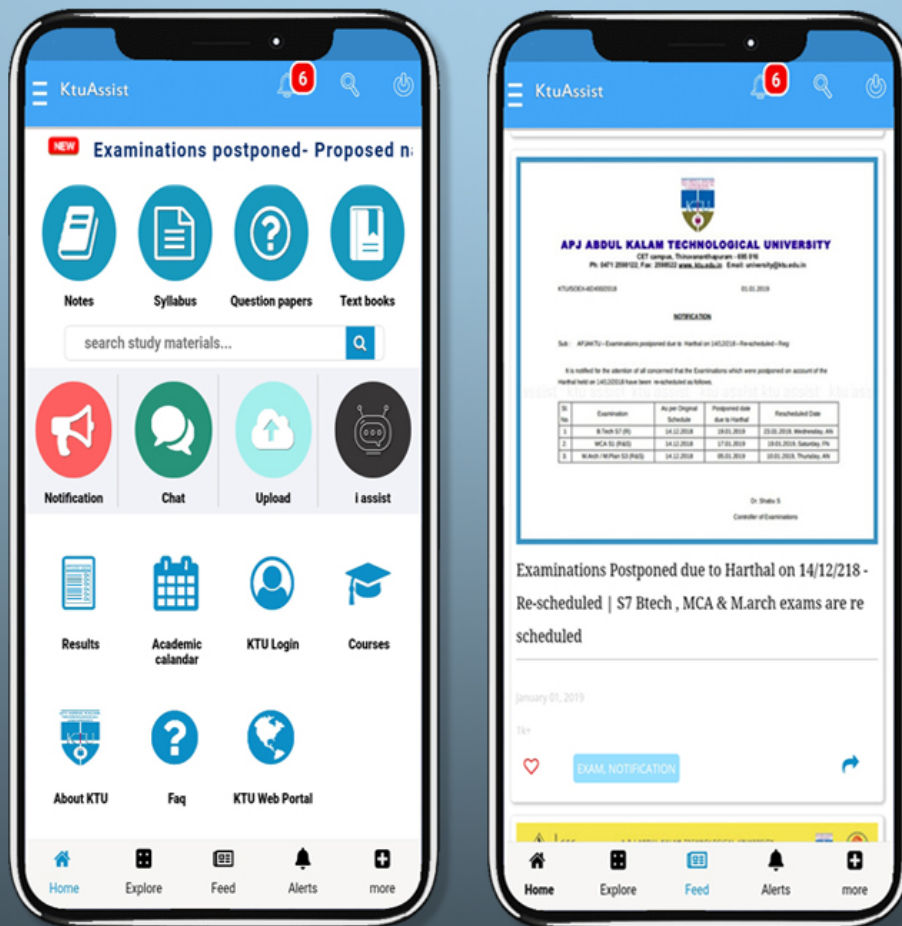
Next the edges between E and G, and B and C are discarded. Next the edge between A and B is selected and add in spanning tree.



Next the edges between, F and G, and A and F, and A and G are discarded.

Now the tree is complete and the minimum cost is:

$$1 + 3 + 4 + 9 + 17 + 23 = 57$$



KTU ASSIST
 GET IT ON GOOGLE PLAY

END