Design and Analysis of Algorithm

Greedy Methods
(Single Source shortest path, Knapsack problem)

Lecture – 45 - 53

Overview

- A greedy algorithm always makes the choice that looks best at the moment. (i.e. it makes a locally optimal choice in the hope that this choice will lead to a globally optimal solution).
- The objective of this section is to explores optimization problems that are solvable by greedy algorithms.

Greedy Algorithm

- In mathematics, computer science and economics, an optimization problem is the problem of finding the best solution from all feasible solutions.
- Algorithms for optimization problems typically go through a sequence of steps, with a set of choices at each step.
- Many optimization problems can be solved using a greedy approach.
- Greedy algorithms are simple and straightforward.

Greedy Algorithm

- A greedy algorithm always makes the choice that looks best at the moment.
- That is, it makes a locally optimal choice in the hope that this choice will lead to a globally optimal solution.
- Greedy algorithms do not always yield optimal solutions, but for many problems they do.
- This algorithms are easy to invent, easy to implement and most of the time provides best and optimized solution.

Greedy Algorithm

- Application of Greedy Algorithm:
  - A simple but nontrivial problem, the activity-selection problem, for which a greedy algorithm efficiently computes a solution.
  - In combinatorics,(a branch of mathematics), a 'matroid' is a structure that abstracts and generalizes the notion of linear independence in vector spaces. Greedy algorithm always produces an optimal solution for such problems. Scheduling unit-time tasks with deadlines and penalties is an example of such problem.

Greedy Algorithm

- Application of Greedy Algorithm:
  - An important application of greedy techniques is the design of data-compression codes (i.e. Huffman code) .
  - The greedy method is quite powerful and works well for a wide range of problems. They are:
    - Minimum-spanning-tree algorithms

      (Example: Prims and Kruskal algorithm)
    - Single Source Shortest Path.

      (Example: Dijkstra's and Bellman ford algorithm)

Greedy Algorithm

- Application of Greedy Algorithm:
  - A problem exhibits optimal substructure if an optimal solution to the problem contains within it optimal solutions to subproblems.
  - This property is a key ingredient of assessing the applicability of dynamic programming as well as greedy algorithms.
  - The subtleties between the above two techniques are illustrated with the help of two variants of a classical optimization problem known as knapsack problem. These variants are:
    - 0-1 knapsack problem (Dynamic Programming)
    - Fractional knapsack problem (Greedy Algorithm)

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - It is a shortest path problem where the shortest path from a given source vertex to all other remaining vertices is computed.
  - **Dijkstra's Algorithm and Bellman Ford Algorithm** are the famous algorithms used for solving single-source shortest path problem.

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Dijkstra's Algorithm**
    - Dijkstra Algorithm is a very famous greedy algorithm.
    - It is used for solving the single source shortest path problem.
    - It computes the shortest path from one particular source node to all other remaining nodes of the graph.

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Dijkstra’s Algorithm (Feasible Condition)**
    - Dijkstra algorithm works
      - for connected graphs.
      - for those graphs that do not contain any negative weight edge.
      - To provides the value or cost of the shortest paths.
      - for directed as well as undirected graphs.

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Dijkstra's Algorithm (Implementation)**

    The implementation of Dijkstra Algorithm is executed in the following steps-
  - **Step-01**:
    - In the first step. two sets are defined-
    - One set contains all those vertices which have been included in the shortest path tree.
    - In the beginning, this set is empty.
    - Other set contains all those vertices which are still left to be included in the shortest path tree.
    - In the beginning, this set contains all the vertices of the given graph.

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Dijkstra's Algorithm (Implementation)**

    The implementation of Dijkstra Algorithm is executed in the following steps-
  - **Step-02**:

    For each vertex of the given graph, two variables are defined as-
    - Π[v] which denotes the predecessor of vertex 'v'
    - d[v] which denotes the shortest path estimate of vertex 'v' from the source vertex.

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Dijkstra's Algorithm (Implementation)**

    The implementation of Dijkstra Algorithm is executed in the following steps-
  - **Step-02**:

    Initially, the value of these variables is set as-
    - The value of variable 'Π' for each vertex is set to NIL i.e. Π[v] = NIL
    - The value of variable 'd' for source vertex is set to 0 i.e. d[S] = 0
    - The value of variable 'd' for remaining vertices is set to ∞ i.e. d[v] = ∞

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Dijkstra's Algorithm (Implementation)**

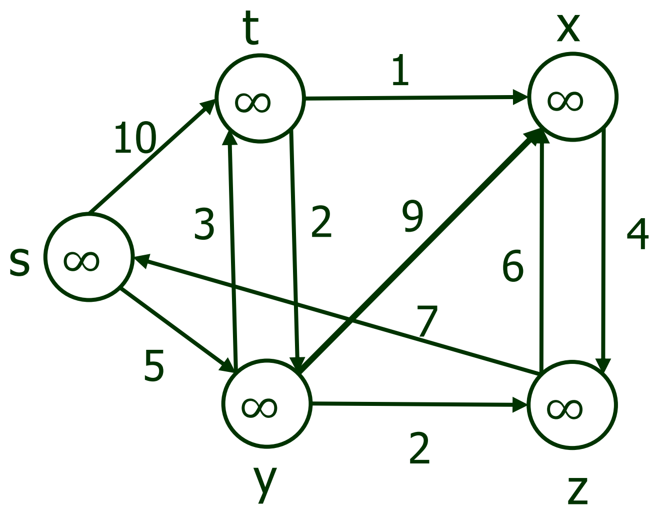
    The implementation of Dijkstra Algorithm is executed in the following steps-
  - **Step-03**:

    The following procedure is repeated until all the vertices of the graph are processed-
    - Among unprocessed vertices, a vertex with minimum value of variable 'd' is chosen.
    - Its outgoing edges are relaxed.
    - After relaxing the edges for that vertex, the sets created in step-01 are updated.

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Dijkstra's Algorithm**

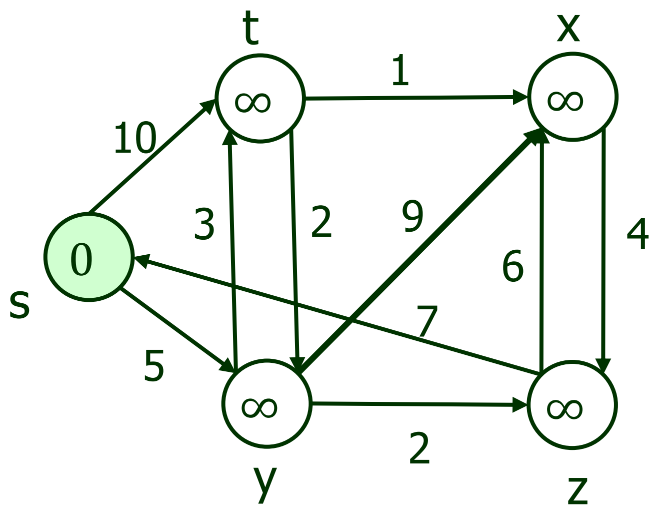
  Example 1: Construct the Single source shortest path for the given graph using Dijkstra's Algorithm-



Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Dijkstra's Algorithm**

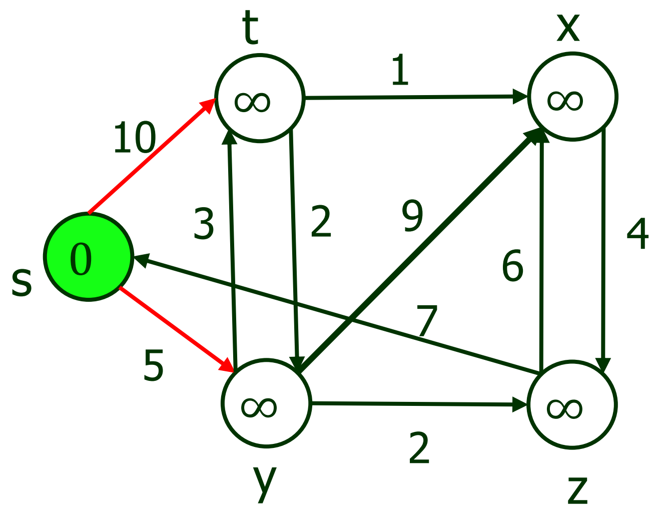
  Example 1: Construct the Single source shortest path for the given graph using Dijkstra's Algorithm-



Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Dijkstra's Algorithm**

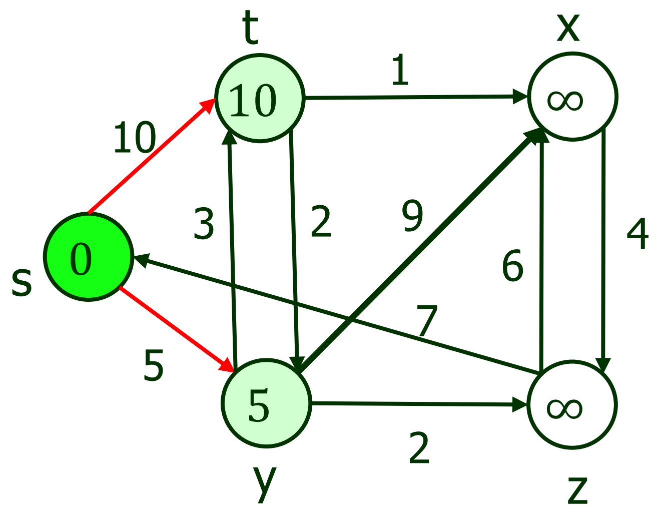
  Example 1: Construct the Single source shortest path for the given graph using Dijkstra's Algorithm-



Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Dijkstra's Algorithm**

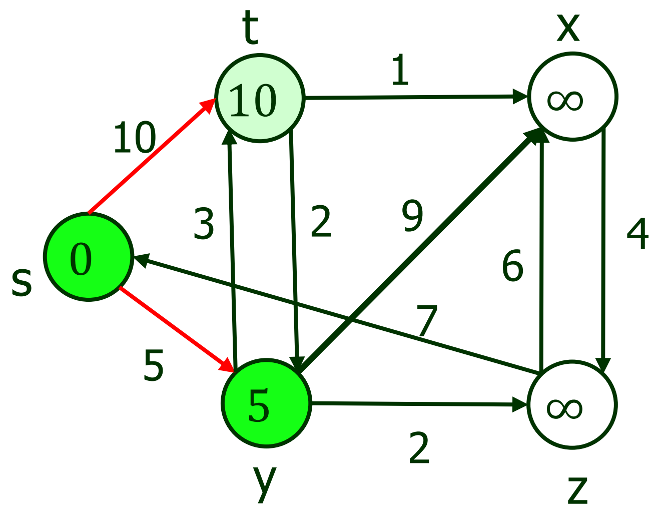
  Example 1: Construct the Single source shortest path for the given graph using Dijkstra's Algorithm-



Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Dijkstra's Algorithm**

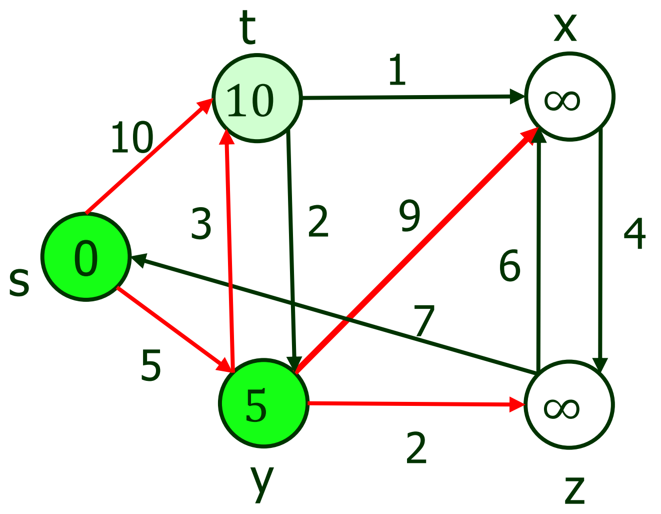
Example 1: Construct the Single source shortest path for the given graph using Dijkstra's Algorithm-



Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Dijkstra's Algorithm**

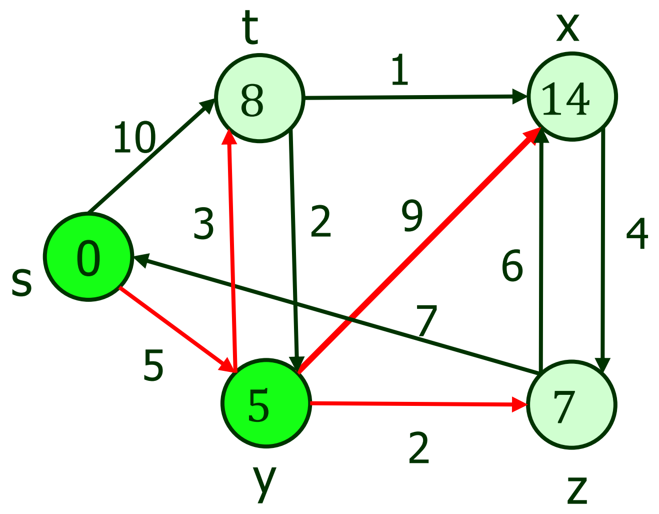
Example 1: Construct the Single source shortest path for the given graph using Dijkstra's Algorithm-



Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Dijkstra’s Algorithm**

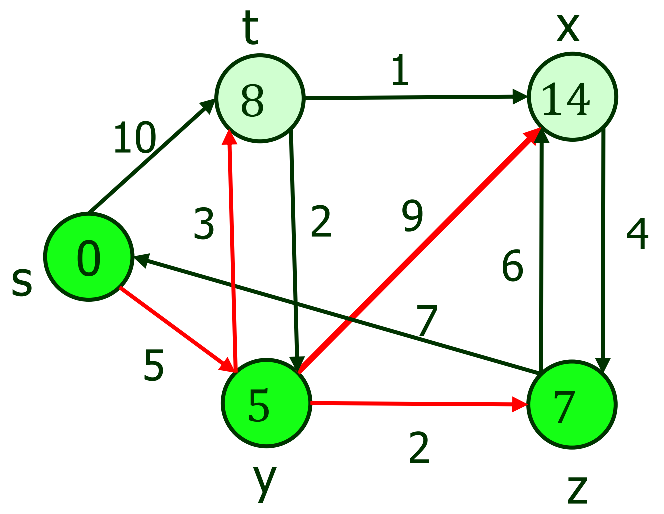
Example 1: Construct the Single source shortest path for the given graph using Dijkstra’s Algorithm-



Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Dijkstra's Algorithm**

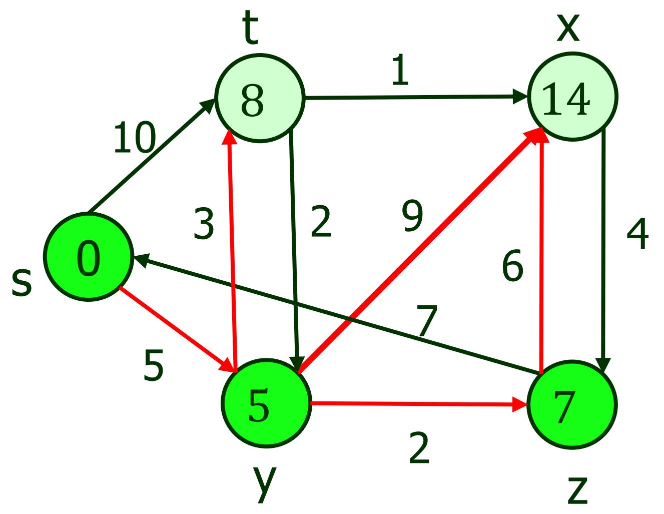
Example 1: Construct the Single source shortest path for the given graph using Dijkstra's Algorithm-



Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Dijkstra's Algorithm**

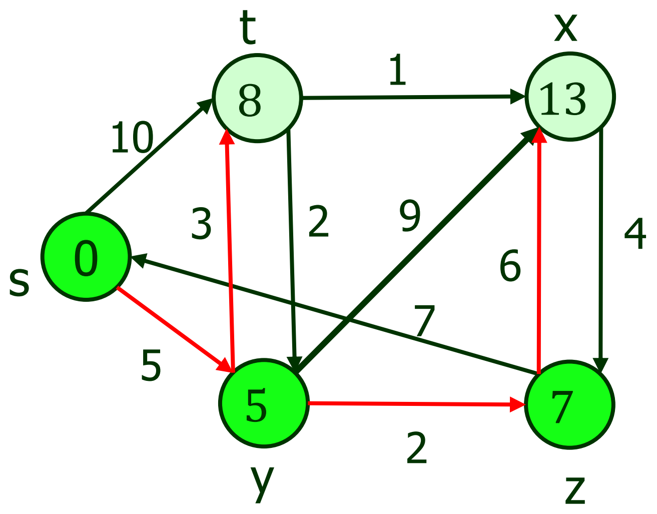
Example 1: Construct the Single source shortest path for the given graph using Dijkstra's Algorithm-



Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Dijkstra's Algorithm**

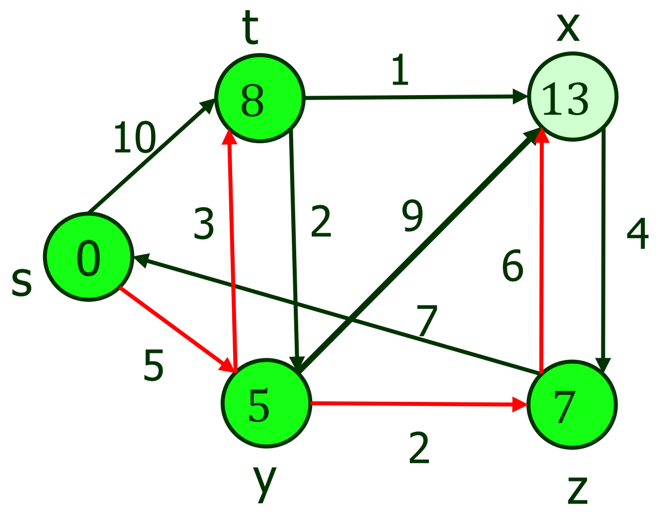
Example 1: Construct the Single source shortest path for the given graph using Dijkstra's Algorithm-



Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Dijkstra's Algorithm**

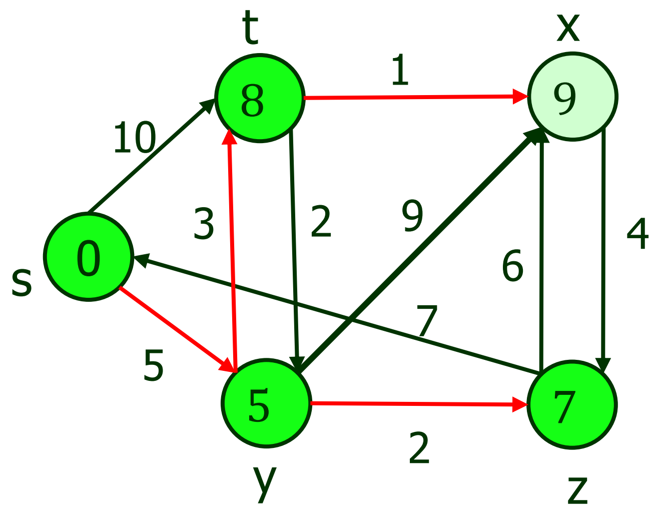
Example 1: Construct the Single source shortest path for the given graph using Dijkstra's Algorithm-



Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Dijkstra's Algorithm**

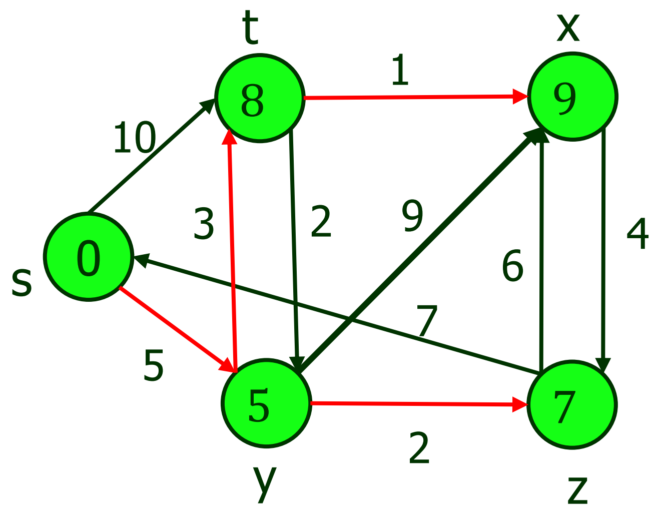
Example 1: Construct the Single source shortest path for the given graph using Dijkstra's Algorithm-



Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Dijkstra's Algorithm**

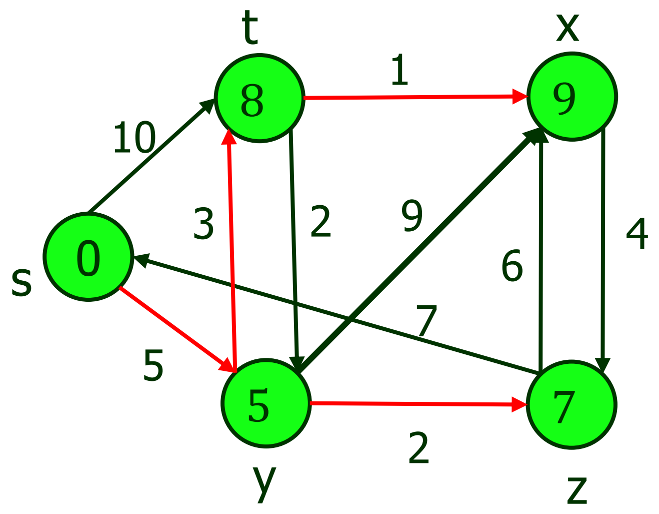
Example 1: Construct the Single source shortest path for the given graph using Dijkstra's Algorithm-



Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Dijkstra's Algorithm**

Example 1: Construct the Single source shortest path for the given graph using Dijkstra's Algorithm-



Hence the shortest path to all the vertex from s are:

$$s \rightarrow t = 8$$
$$s \rightarrow x = 9$$
$$s \rightarrow y = 5$$
$$s \rightarrow z = 7$$

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Dijkstra's Algorithm (Algorithm)**

```
DIJKSTRA(G, w, s)
1 INITIALIZE-SINGLE-SOURCE(G, s)
2 S ← Ø
3 Q ← V[G]
4 while Q ≠ Ø
5     do u ← EXTRACT-MIN(Q)
6            S ← $S \in \{u\}$
7            for each vertex $v \in Adj[u]$
8                 do RELAX(u, v, w)
```

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Dijkstra’s Algorithm (Algorithm)**

```
INITIALIZE-SINGLE-SOURCE(G, s)
1     for each vertex v  V[G]
2             do d[v] ← ∞
3             π[v] ← NIL
4     d[s] ← 0
RELAX(u, v, w)
1 if d[v] > d[u] + w(u, v)
2     then d[v] ← d[u] + w(u, v)
3             π[v] ← u
```

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Dijkstra's Algorithm (Complexity)**

CASE-01:

- $A[i, j]$ stores the information about edge (i, j).
- Time taken for selecting i with the smallest $dist$ is $O(V)$.
- For each neighbor of i, time taken for updating $dist[j]$ is $O(1)$ and there will be maximum V neighbors.
- Time taken for each iteration of the loop is O(V) and one vertex is deleted from Q.
- Thus, total time complexity becomes O(V^2).

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Dijkstra's Algorithm (Complexity)**

CASE-02:

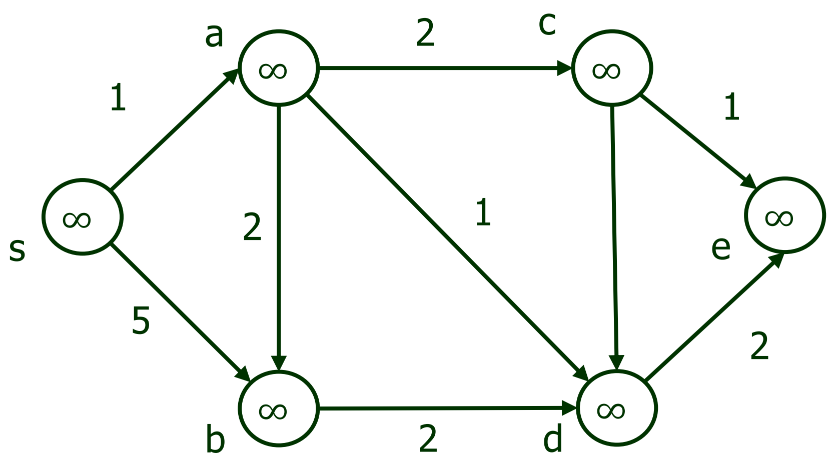
- With adjacency list representation, all vertices of the graph can be traversed using BFS in $O(V+E)$ time.
- In min heap, operations like extract-min and decrease-key value takes $O(\log V)$ time.
- So, overall time complexity becomes

  $O(E+V)\ x\ O(\log V)$ which is $O((E\ +\ V)\ x \log V)\ =\ O(E \log V)$
- This time complexity is reduced to $O(E+V\log V)$ using Fibonacci heap.

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Dijkstra's Algorithm (Self Practice)**

Example 2: Construct the Single source shortest path for the given graph using Dijkstra's Algorithm-



Greedy Algorithm

- **Problem 5: Single source shortest path**
- **Bellman Ford Algorithm**
  - Bellman-Ford algorithm solves the single-source shortest-path problem in the general case in which edges of a given digraph can have negative weight as long as G contains no negative cycles.
  - Like Dijkstra's algorithm, this algorithm, uses the notion of edge relaxation but does not use with greedy method. Again, it uses d[u] as an upper bound on the distance d[u, v] from u to v.

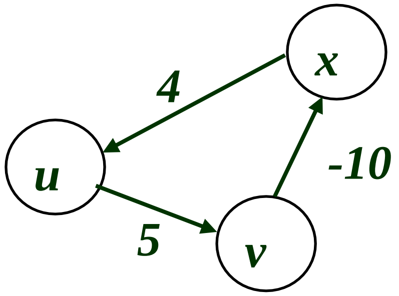
Greedy Algorithm

- **Problem 5: Single source shortest path**
- **Bellman Ford Algorithm**
  - The algorithm progressively decreases an estimate d[v] on the weight of the shortest path from the source vertex s to each vertex v ∊ V until it achieve the actual shortest-path.
  - The algorithm returns Boolean TRUE if the given digraph contains no negative cycles that are reachable from source vertex s otherwise it returns Boolean FALSE.

Greedy Algorithm

- **Problem 5: Single source shortest path**

Bellman Ford Algorithm (Negative Cycle Detection)



Assume:

$$\begin{aligned} d[u] &\le d[x] + 4 \\ d[v] &\le d[u] + 5 \\ d[x] &\le d[v] - 10 \end{aligned}$$

Adding:

$$d[u] + d[v] + d[x] \le d[x] + d[u] + d[v] - 1$$

Because it's a cycle, vertices on left are same as those on right. Thus we get $0 \le -1$; a contradiction.

So for at least one edge (u, v),

$$d[v] > d[u] + w(u, v)$$

This is exactly what Bellman-Ford checks for.

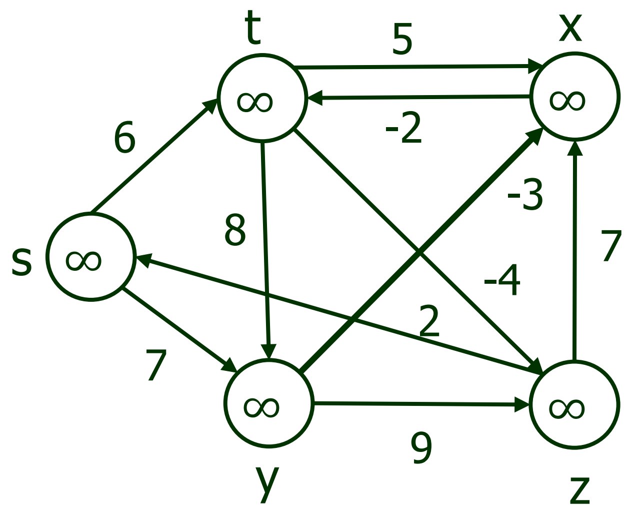
Greedy Algorithm

- **Problem 5: Single source shortest path**
- **Bellman Ford Algorithm (Implementation)**
  - Step 1: Start with the weighted graph.
  - Step 2: Choose the starting vertex by making the path value zero and assign infinity path values to all other vertices.
  - Step 3: Visit each edge and relax the path distances if they are inaccurate.
  - Step 4: Do step 3 V times because in the worst case a vertex's path length might need to be readjusted V times.
  - Step 5: After all vertices have their path lengths, check if a negative cycle is present or not.

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

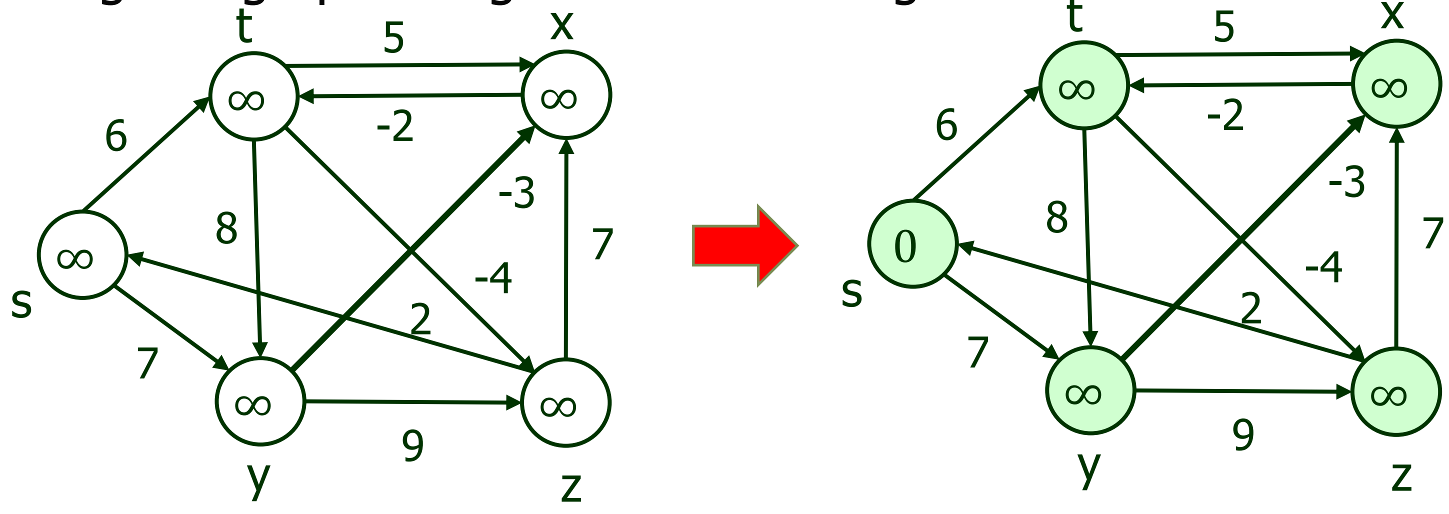
  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-



Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-

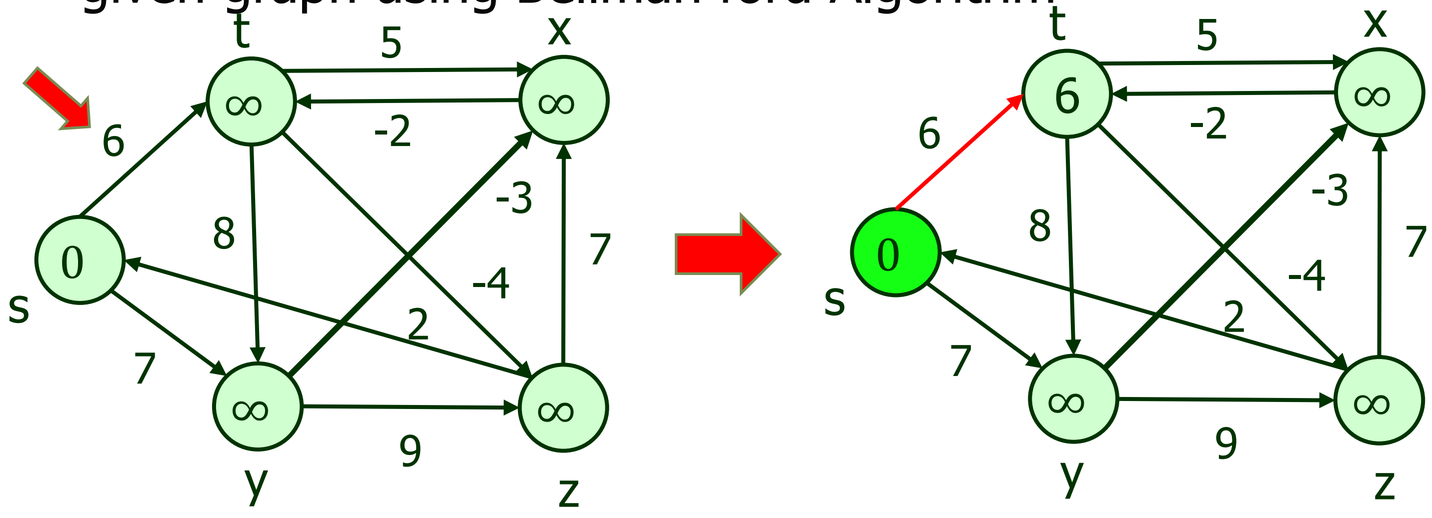


Iteration - 1

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-

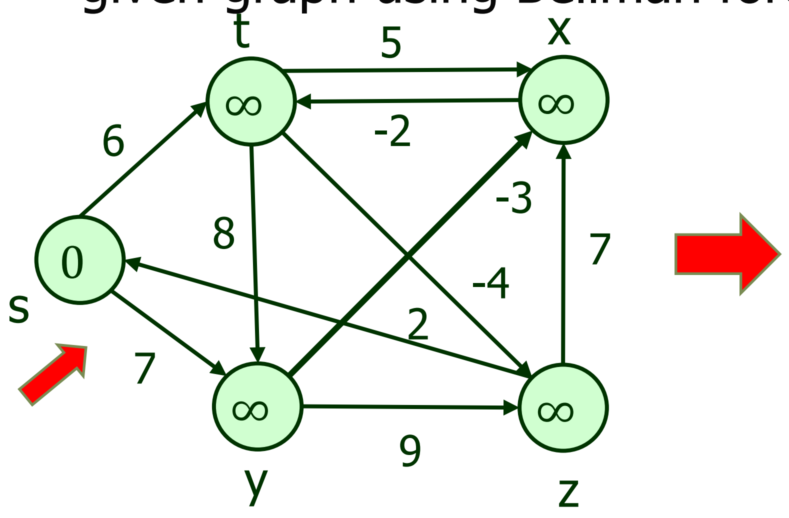


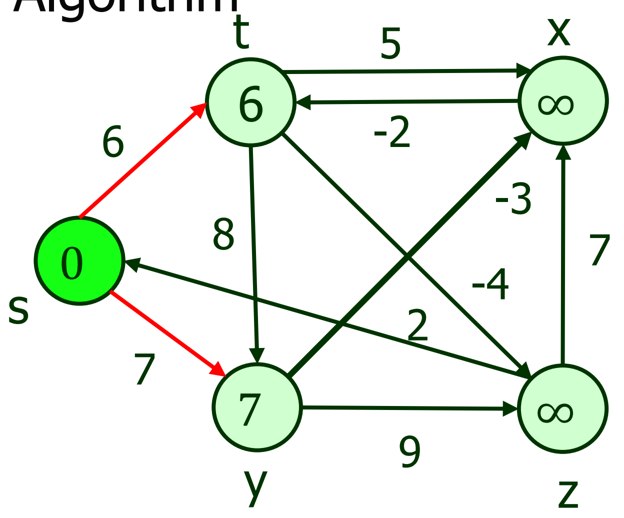
Iteration - 1 Edge no - 1

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-



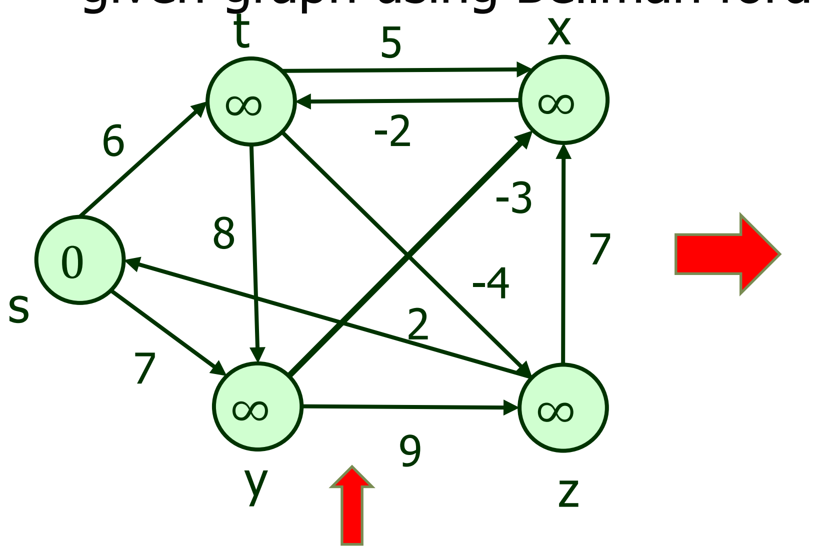


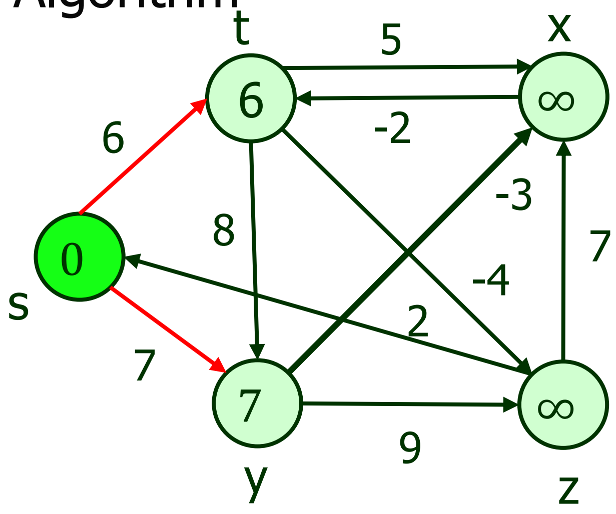
Iteration - 1 Edge no - 2

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-





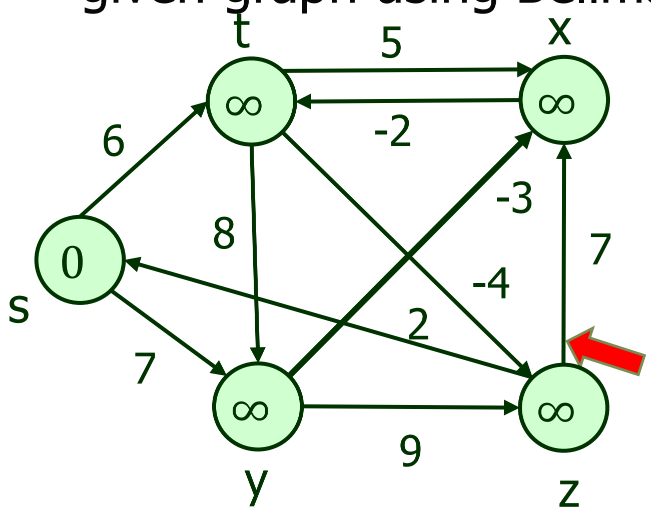
Iteration - 1

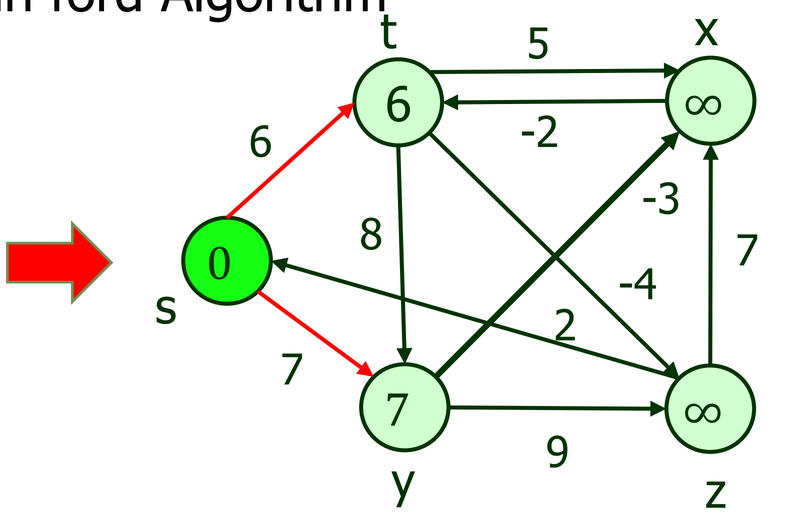
Edge no - 3

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-



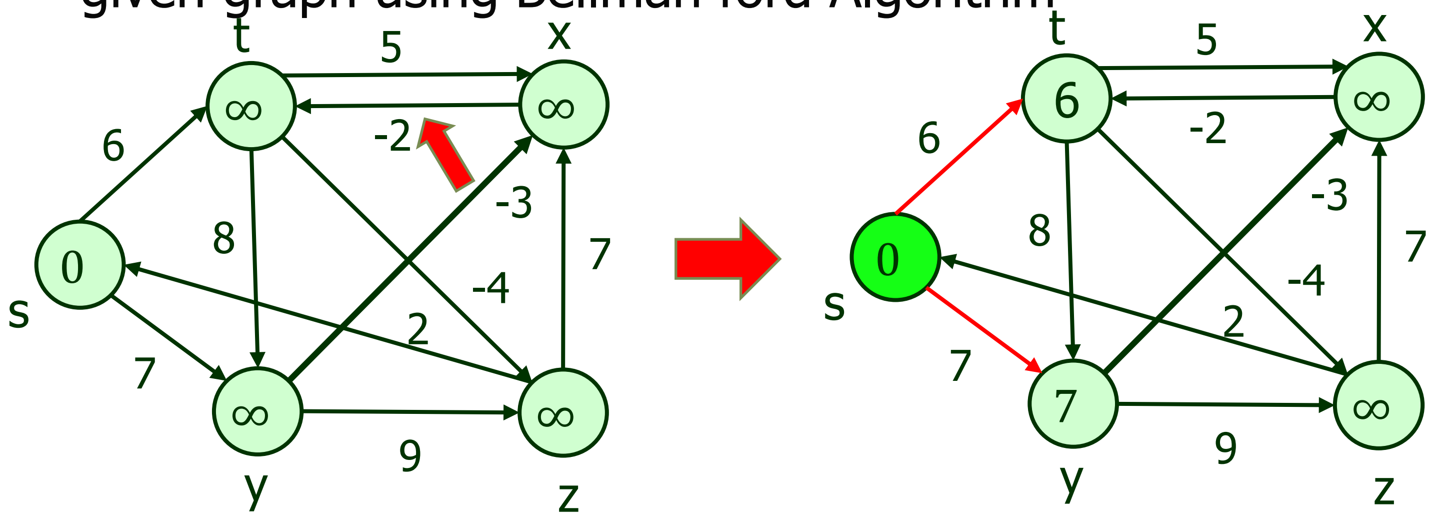


Iteration - 1 Edge no - 4

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-

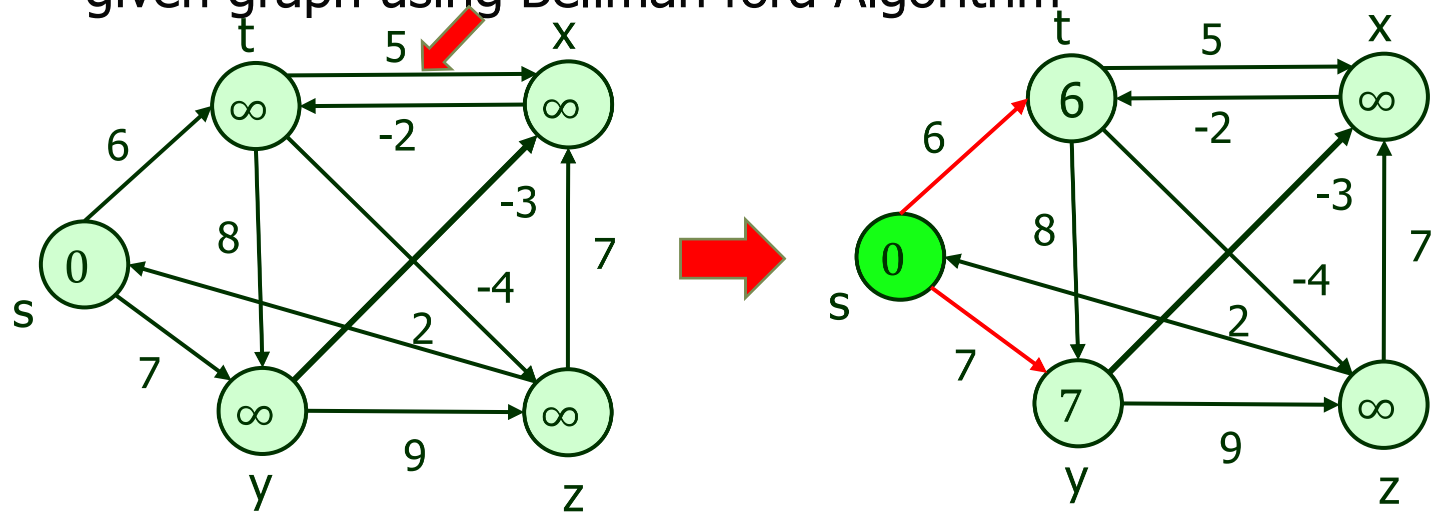


Iteration - 1 Edge no - 5

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-

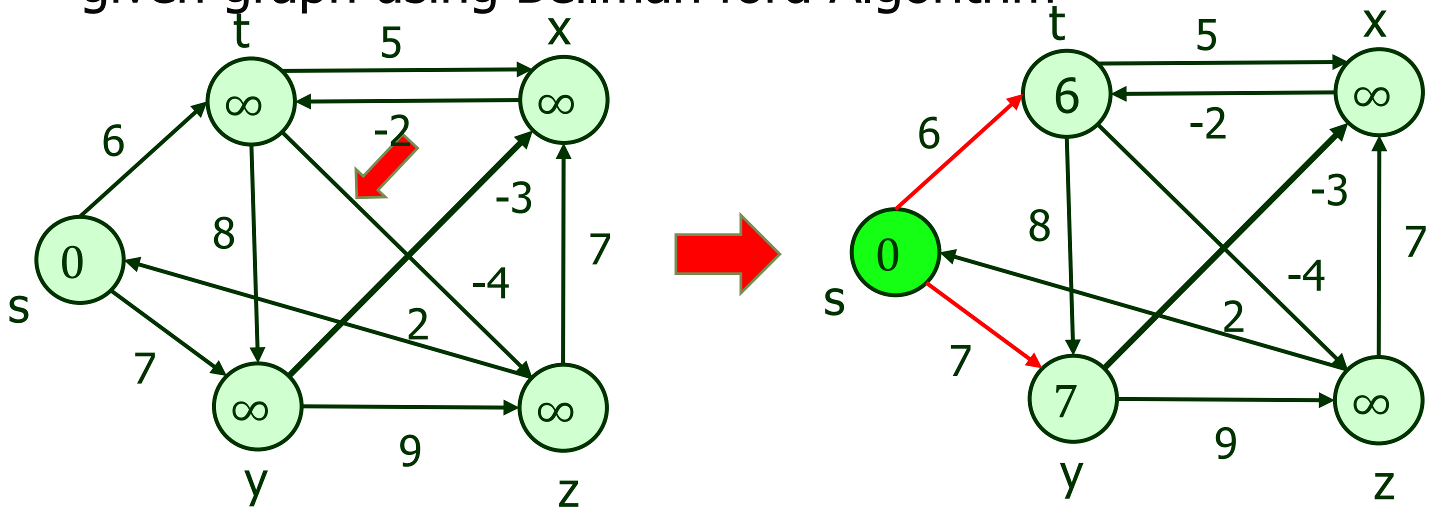


Iteration - 1 Edge no - 6

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-

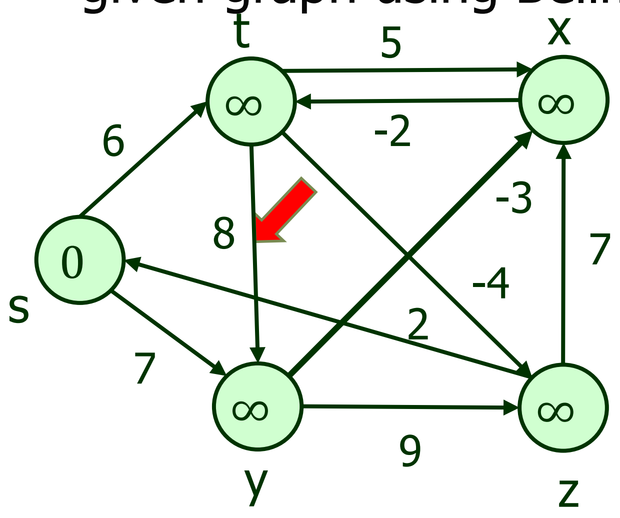


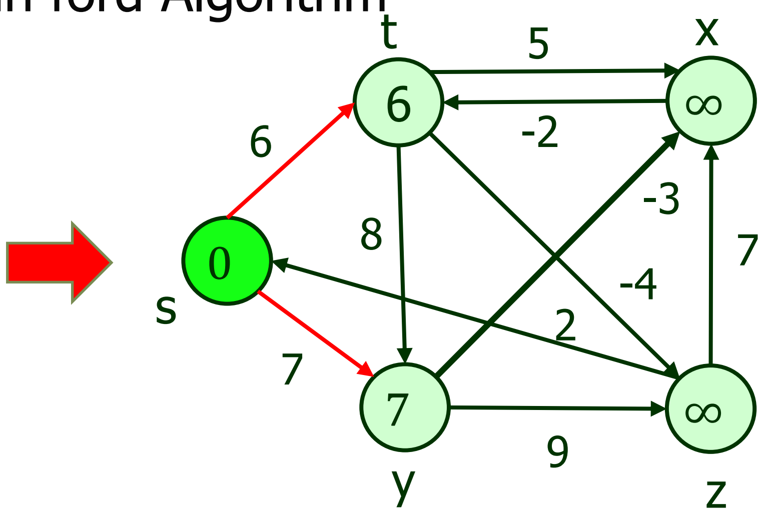
Iteration - 1 Edge no - 7

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-



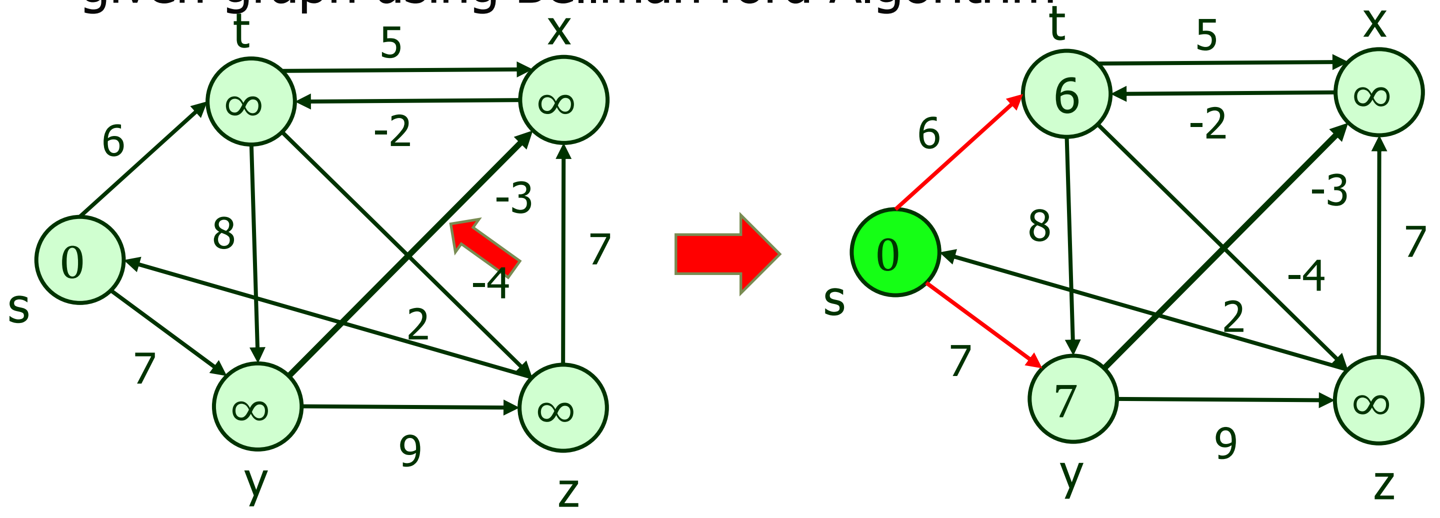


Iteration - 1 Edge no - 8

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-

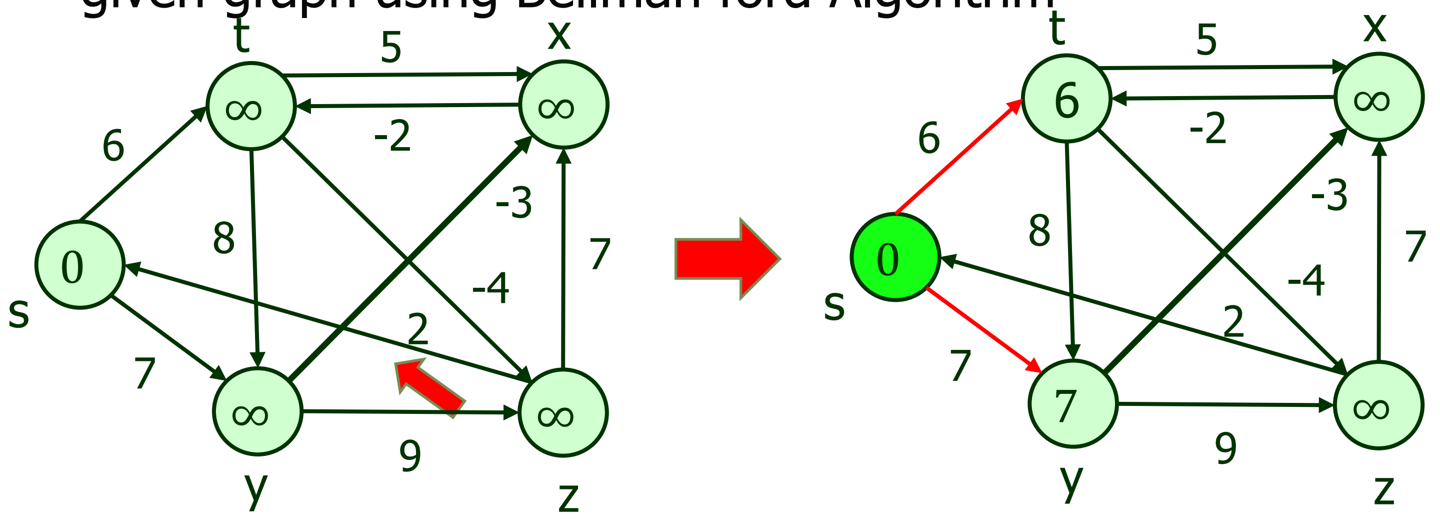


Iteration - 1 Edge no - 9

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-

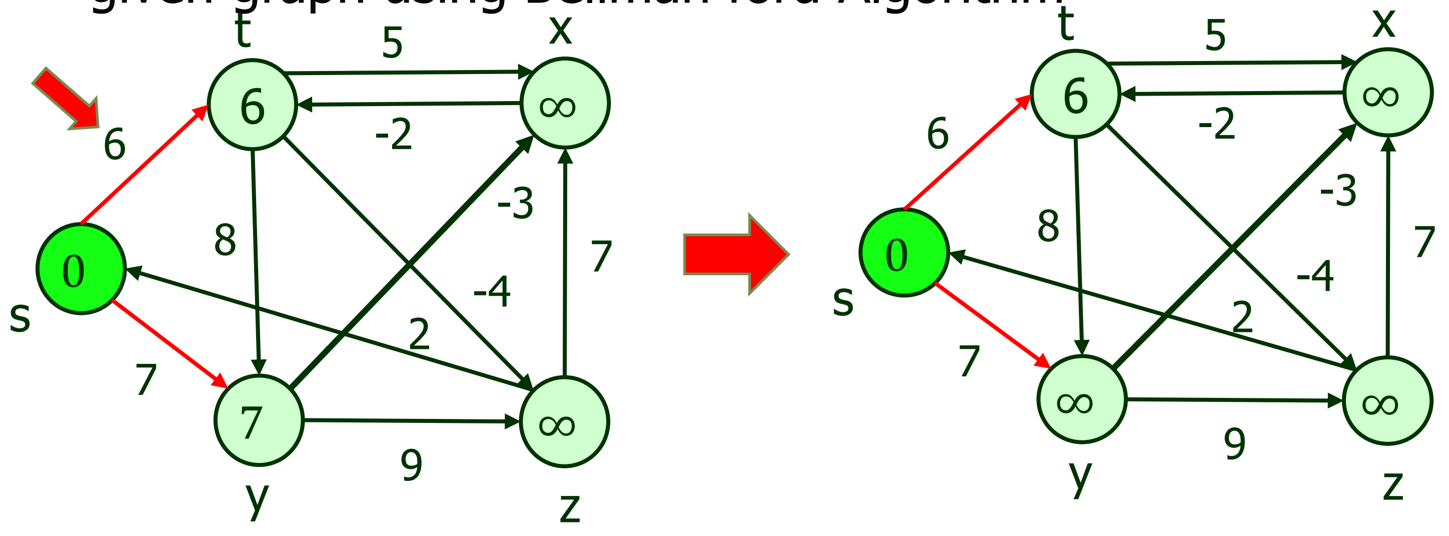


Iteration - 1 Edge no – 10

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-

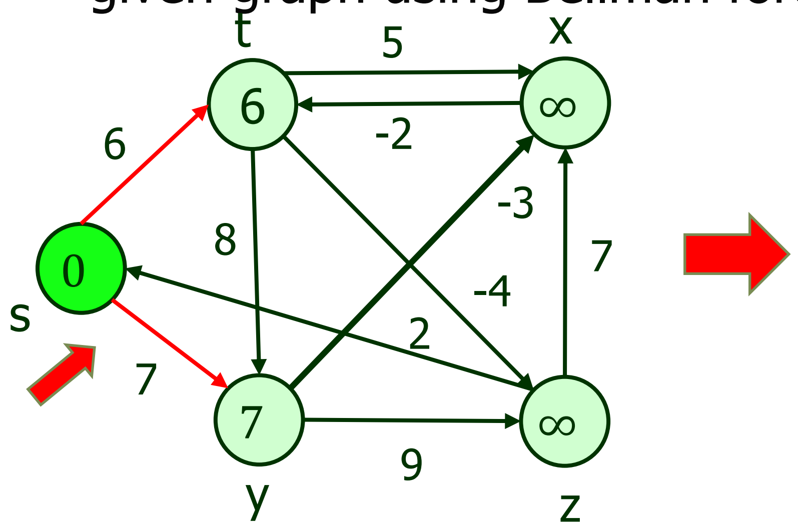


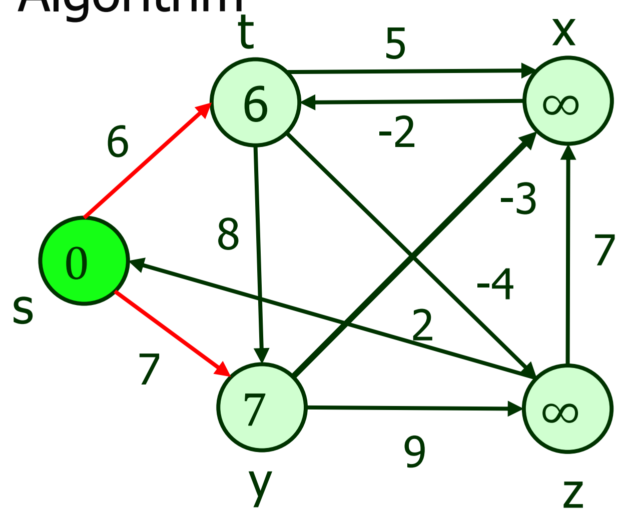
Iteration - 2 Edge no - 1

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-



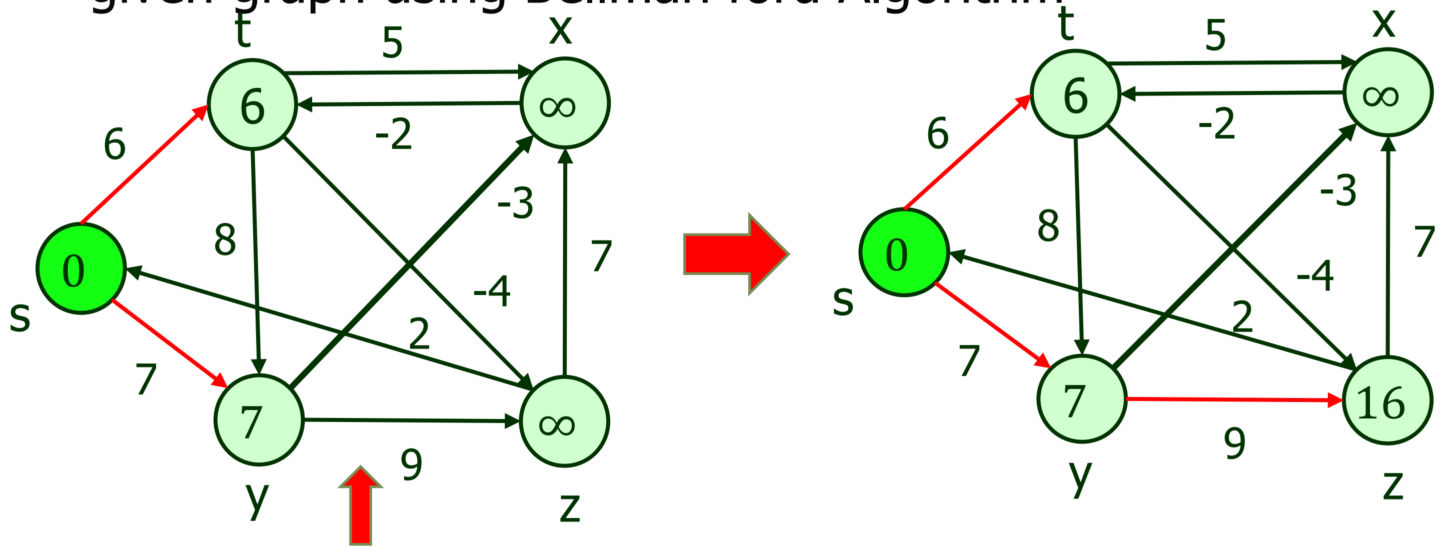


Iteration - 2 Edge no - 2

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-

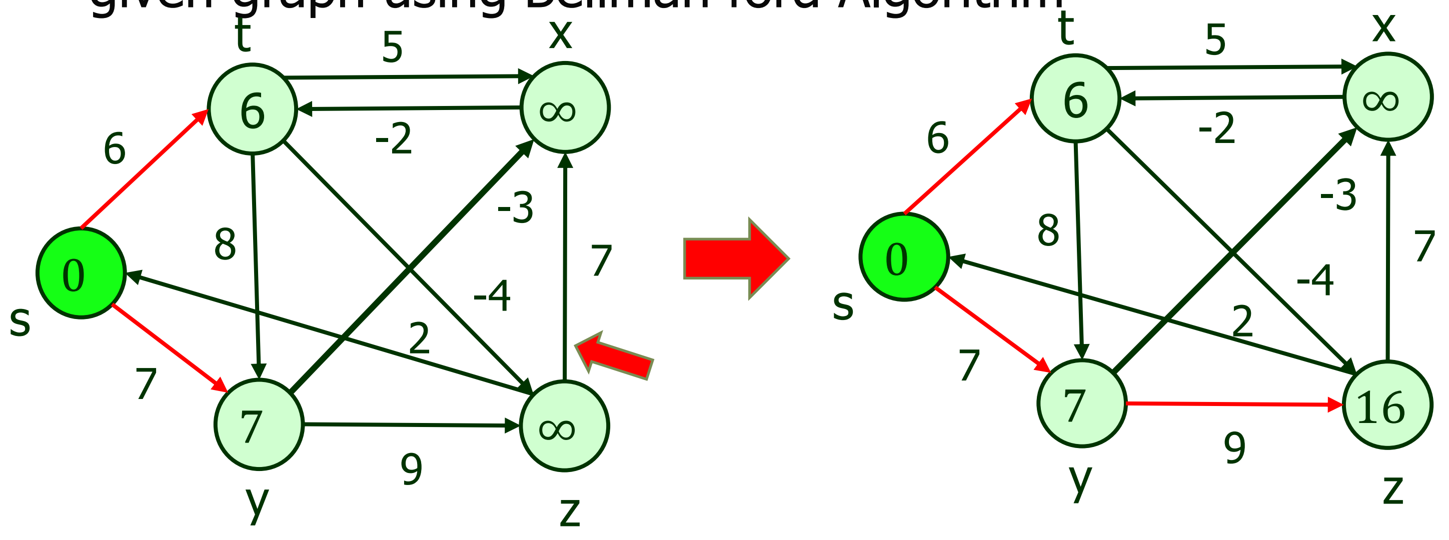


Iteration - 2 Edge no - 3

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-

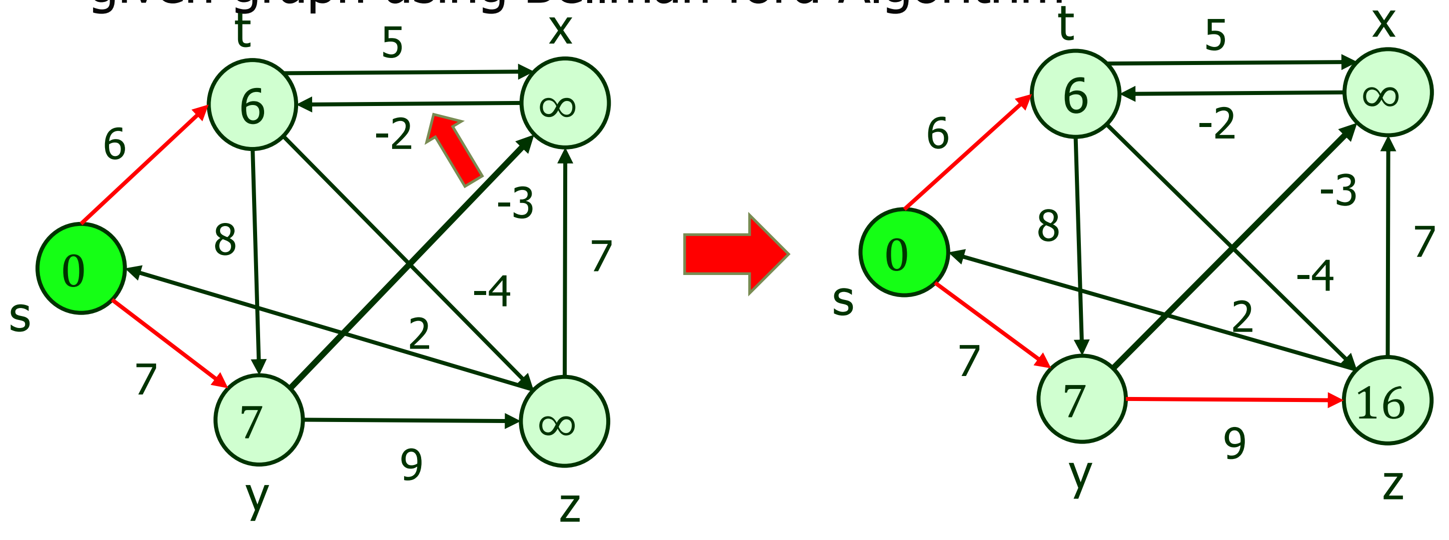


Iteration - 2 Edge no - 4

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-

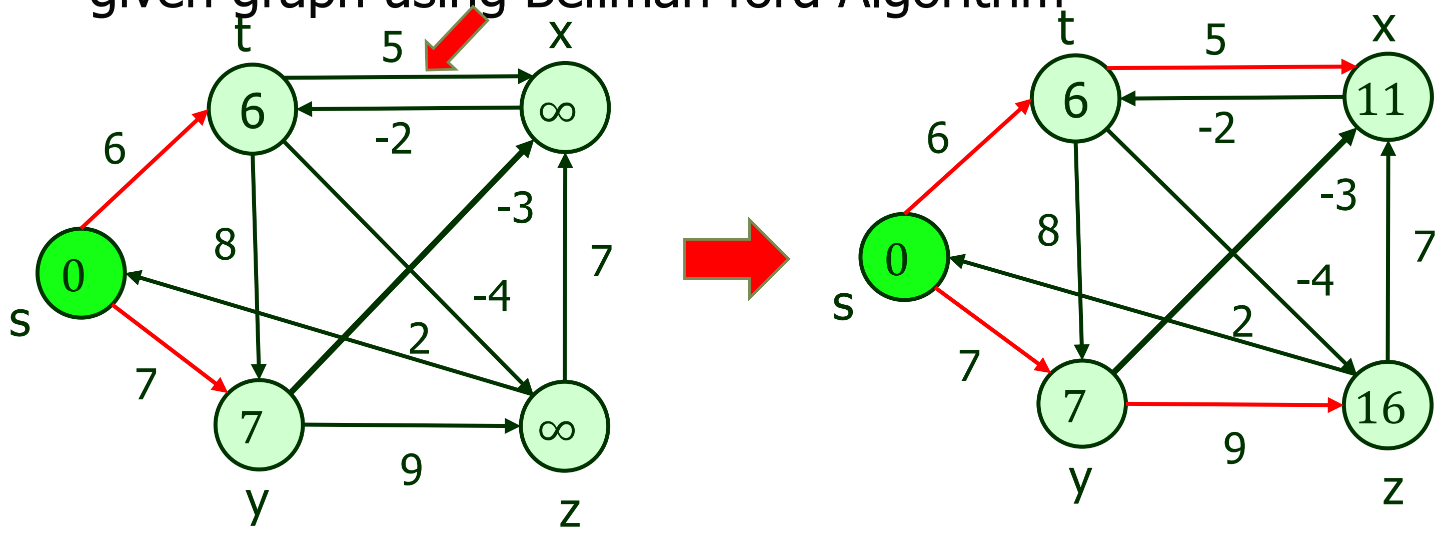


Iteration - 2   Edge no - 5

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-

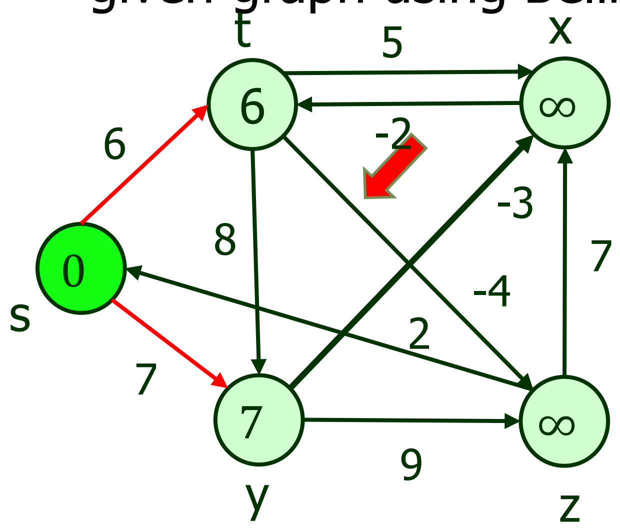


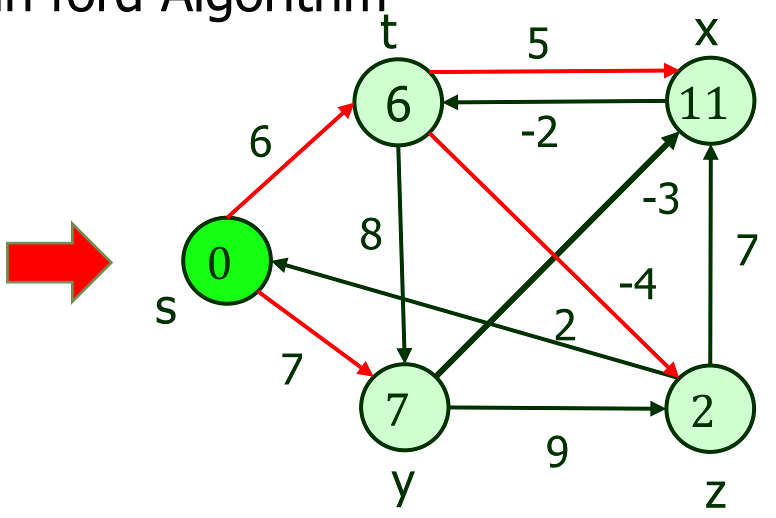
Iteration - 2   Edge no - 6

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-



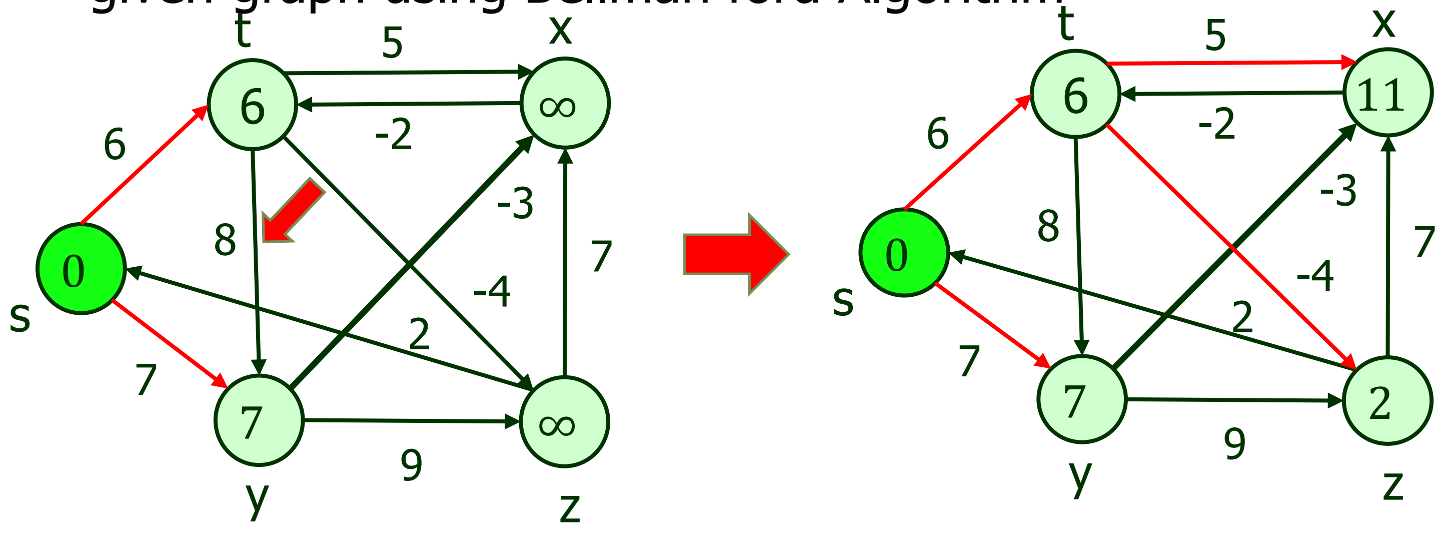


Iteration - 2   Edge no - 7

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-

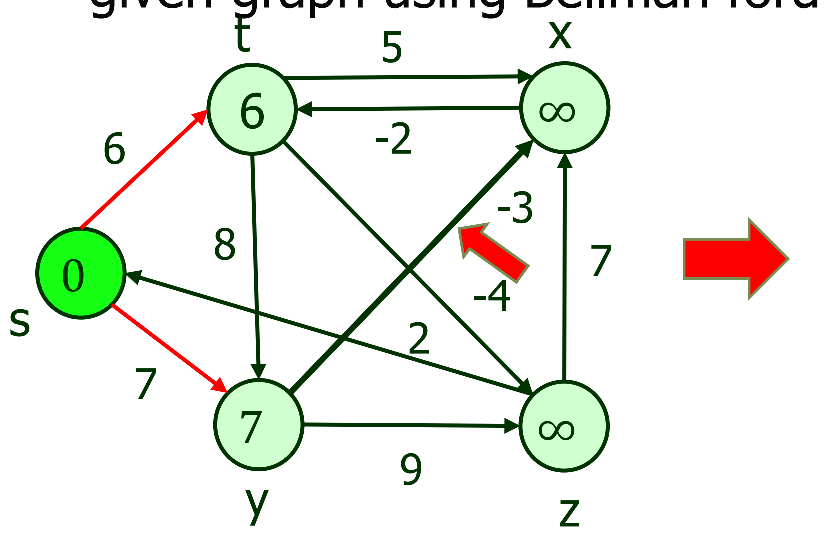


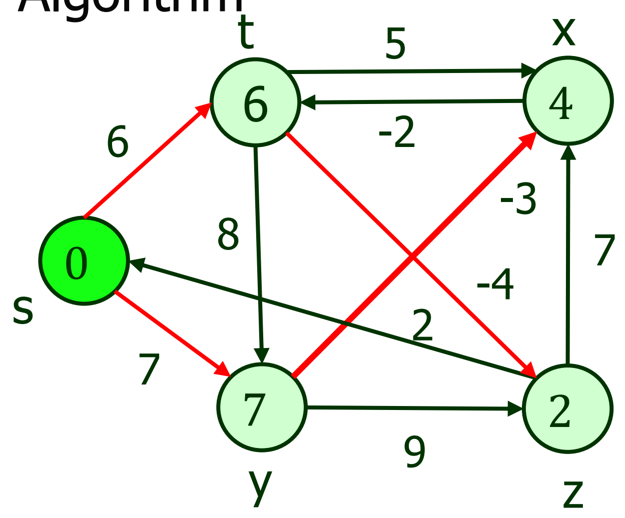
Iteration - 2 Edge no - 8

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-



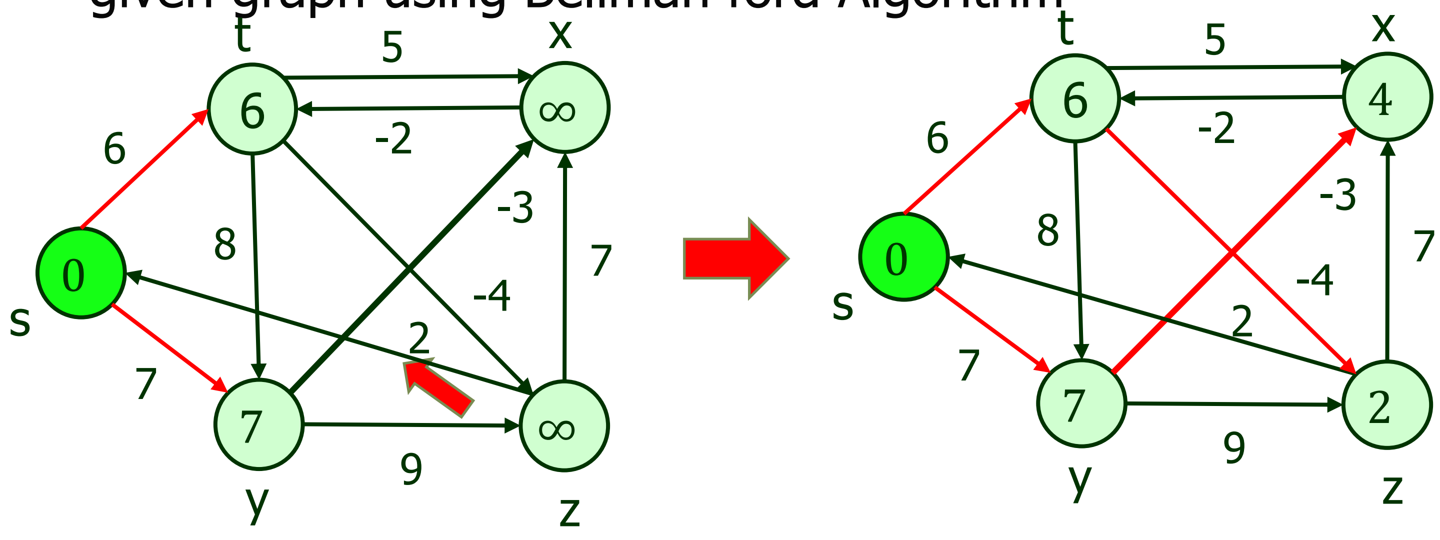


Iteration - 2 Edge no - 9

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-

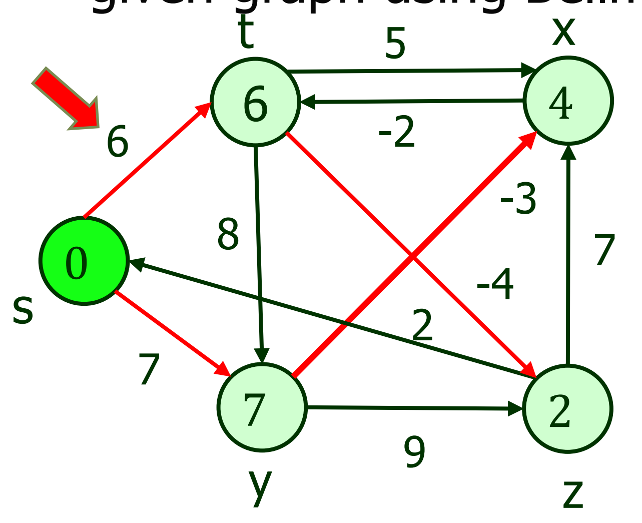


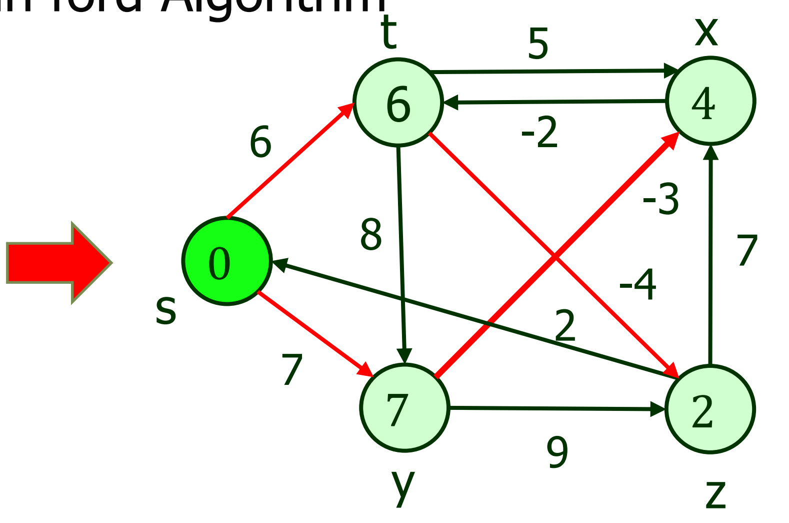
Iteration - 2 Edge no – 10

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-



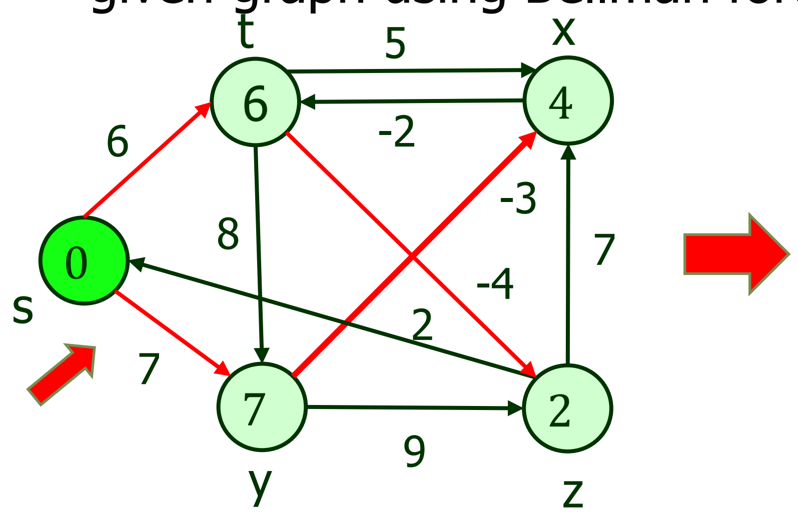


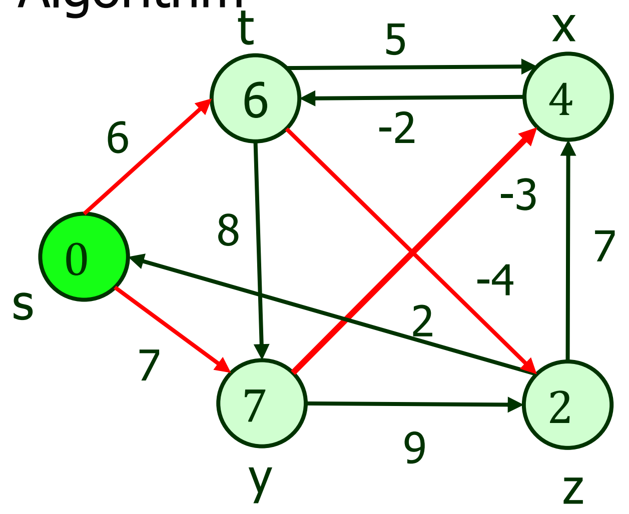
Iteration - 3 Edge no - 1

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-



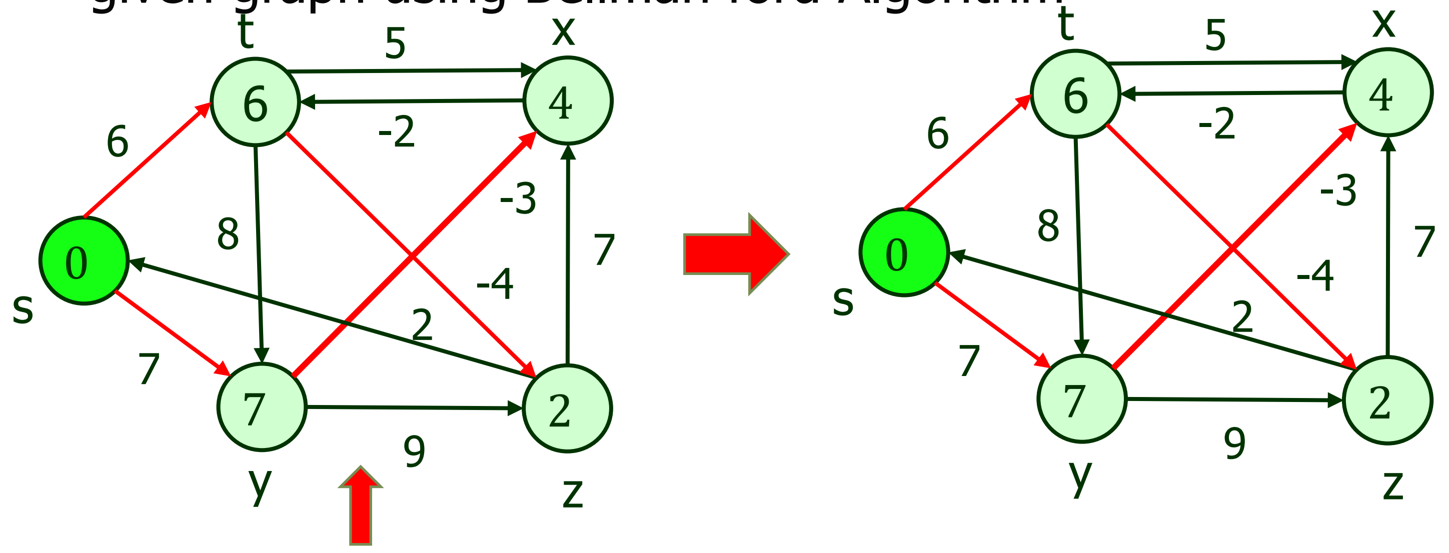


Iteration - 3 Edge no - 2

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-

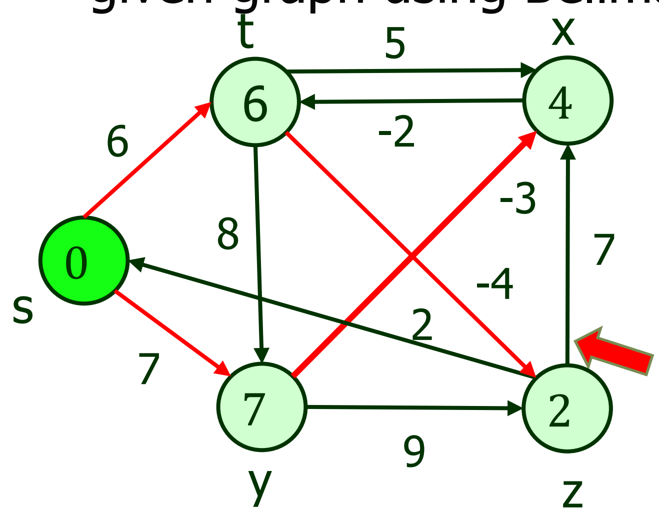


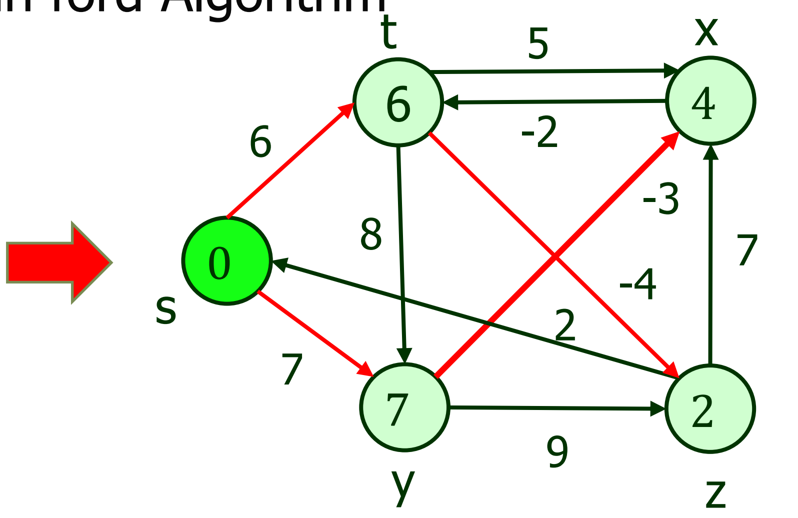
Iteration - 3 Edge no - 3

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-



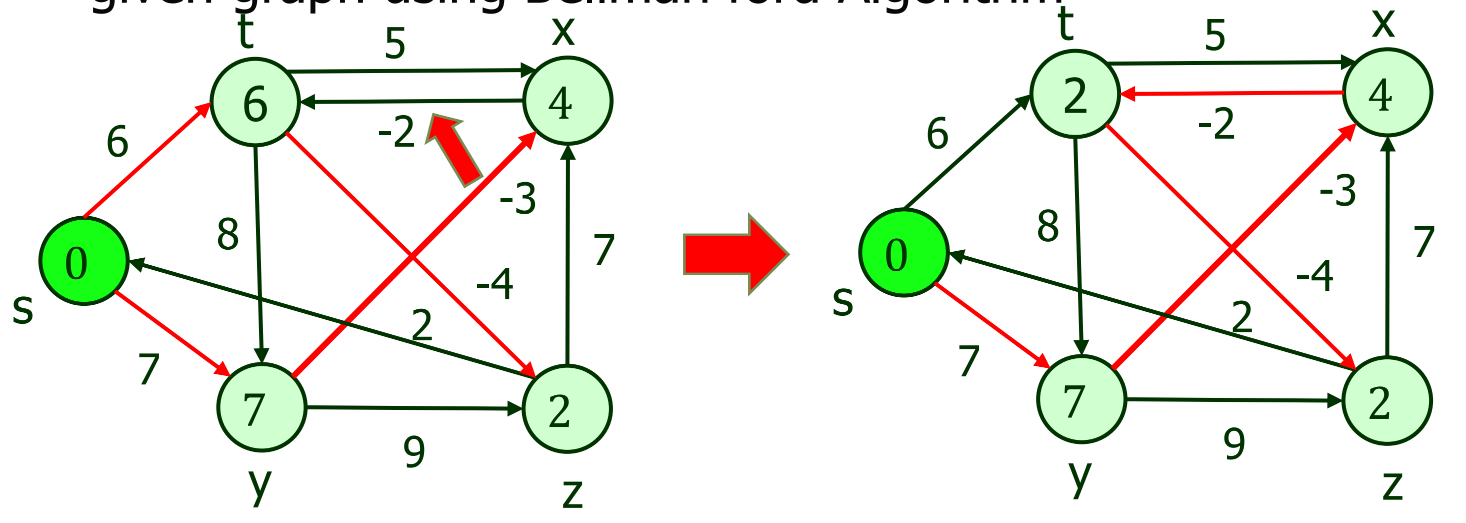


Iteration - 3 Edge no - 4

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-

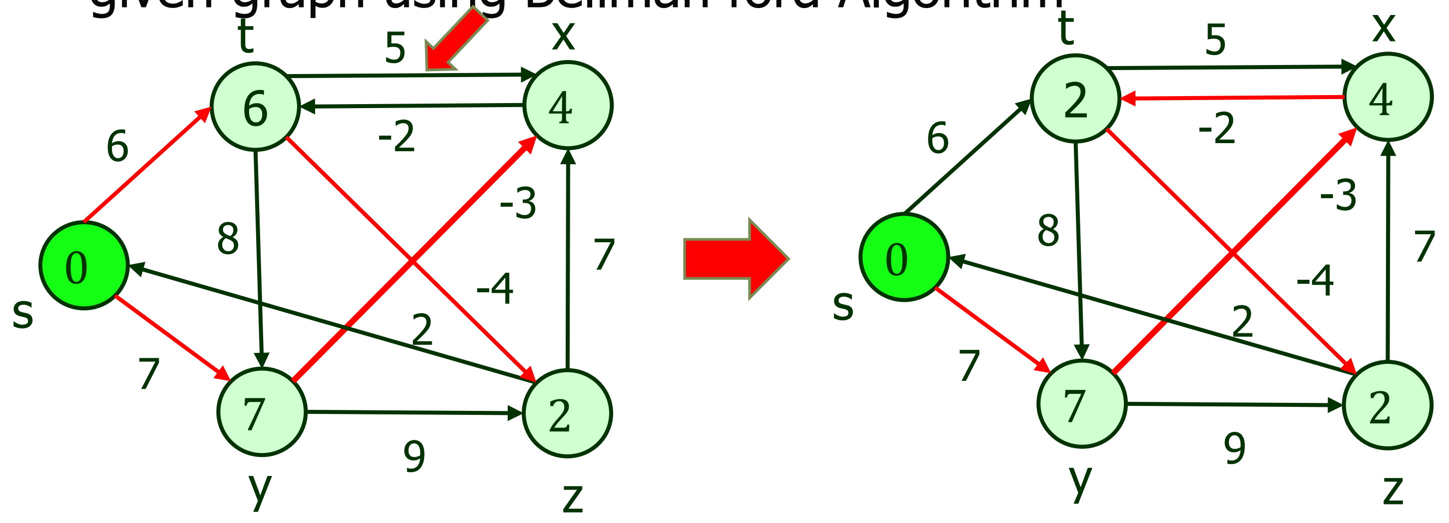


Iteration - 3 Edge no - 5

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-

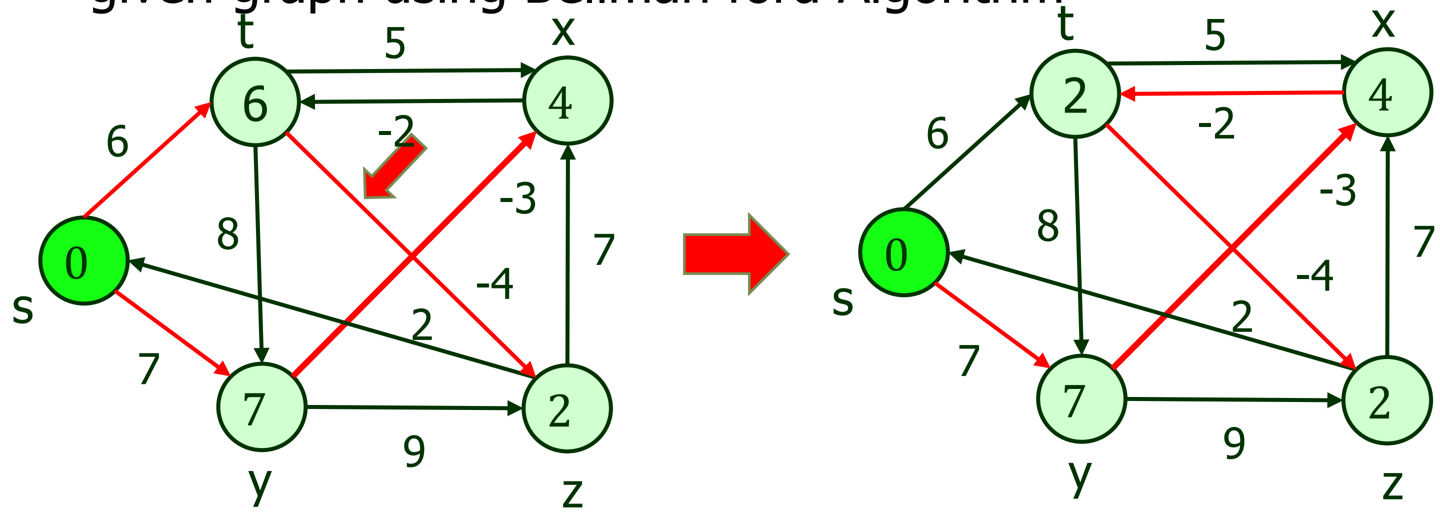


Iteration - 3 Edge no - 6

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-

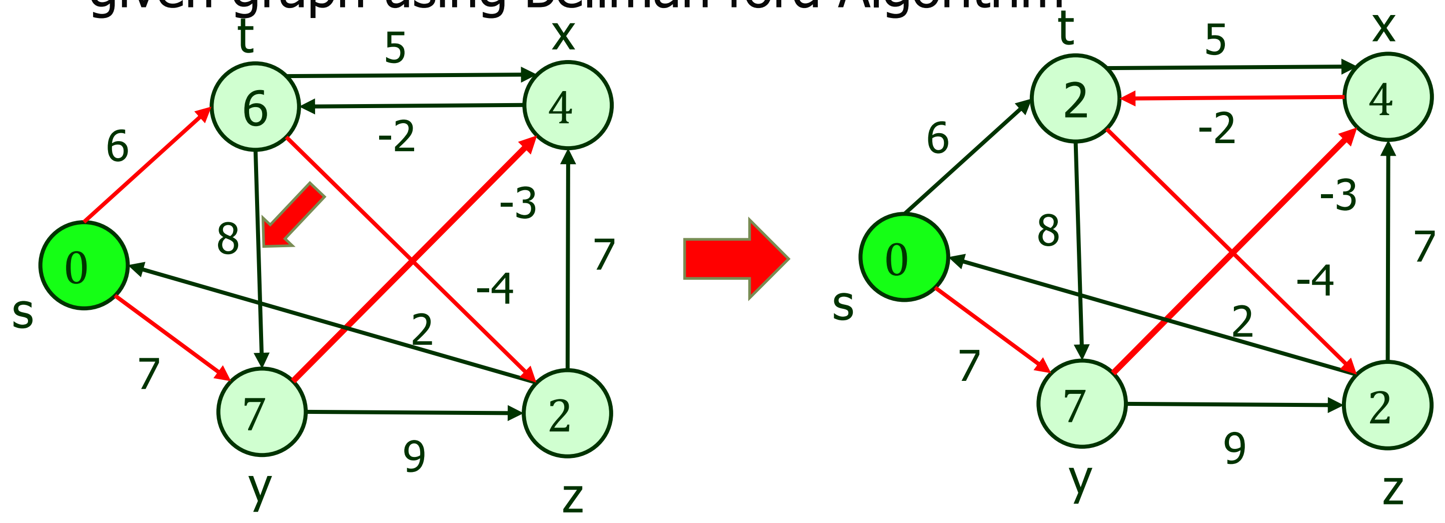


Iteration - 3 Edge no - 7

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-

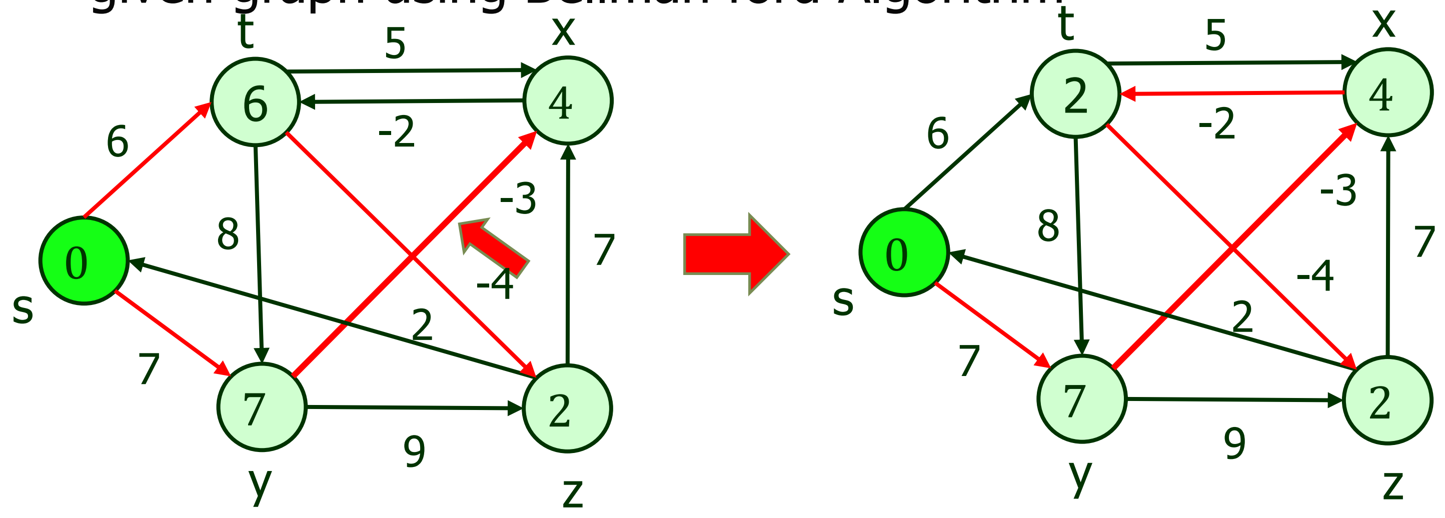


Iteration - 3 Edge no - 8

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-

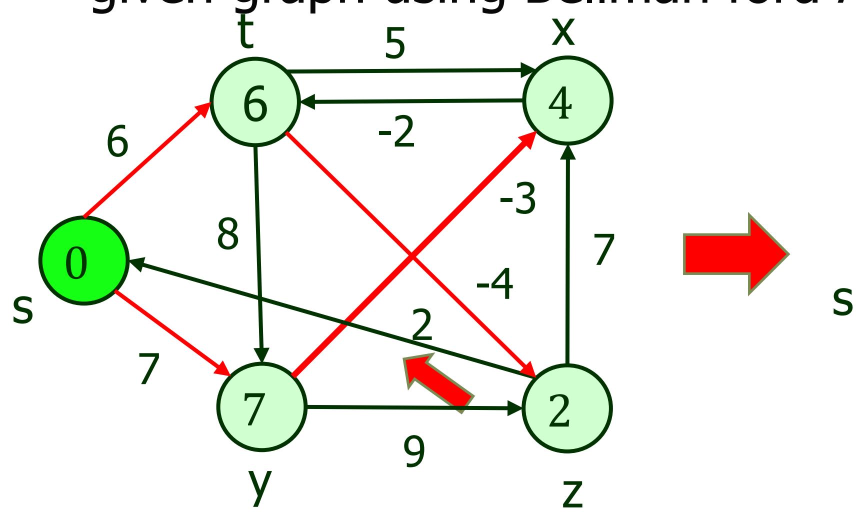


Iteration - 3 Edge no - 9

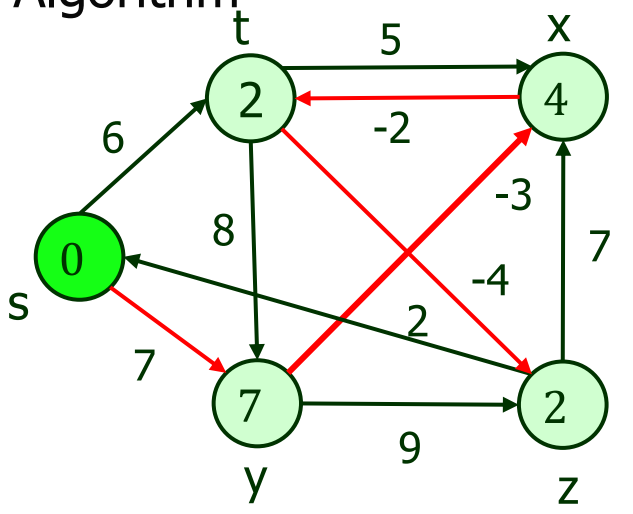
Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-



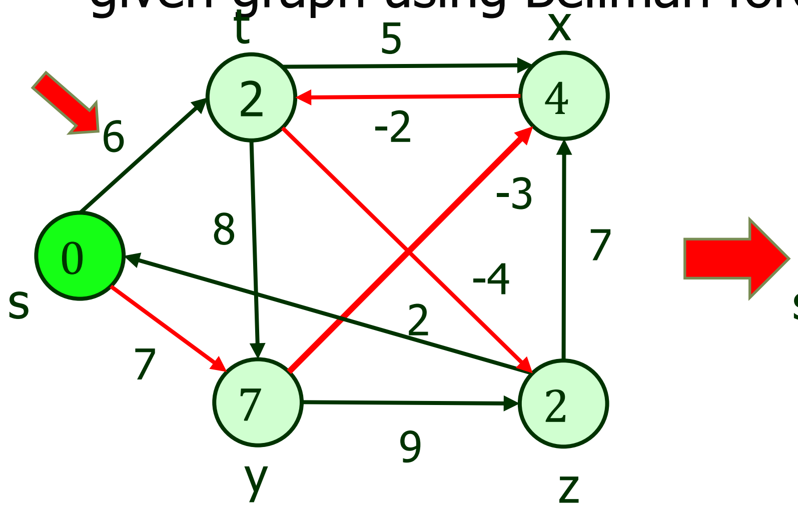
Iteration - 3　　　　Edge no – 10

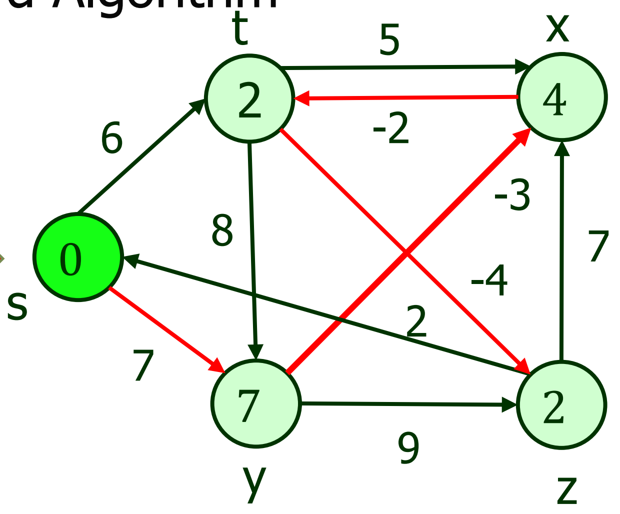


Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-



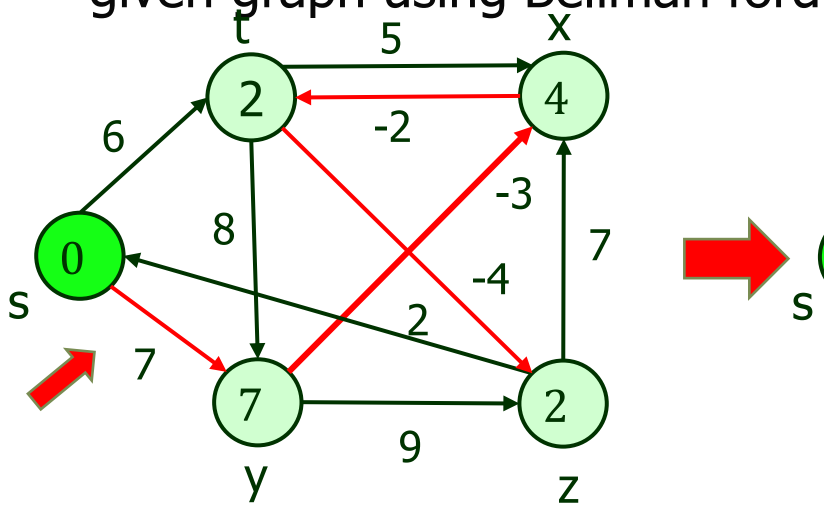


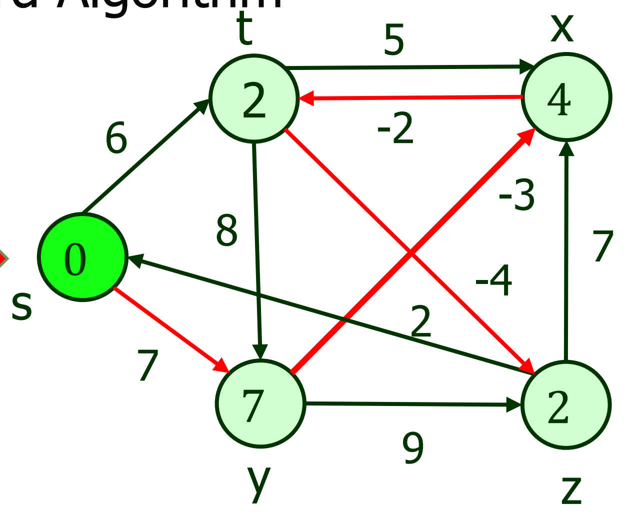
Iteration - 4 Edge no - 1

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-



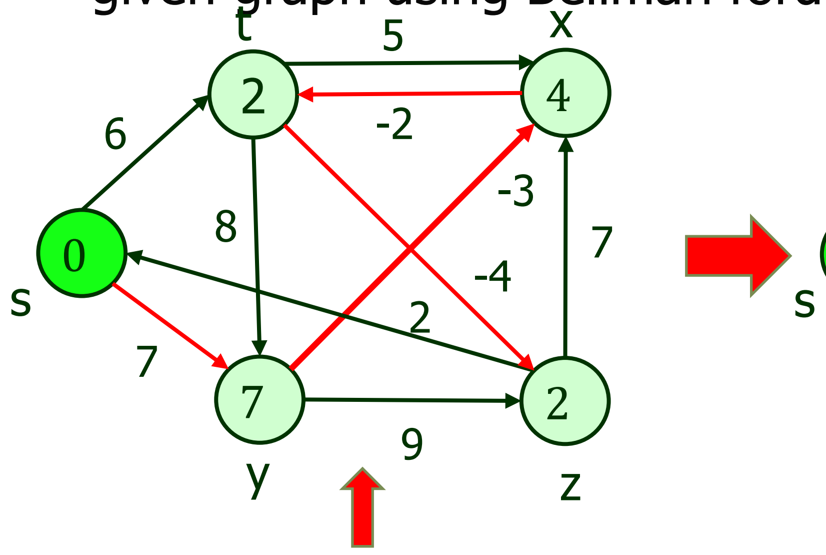


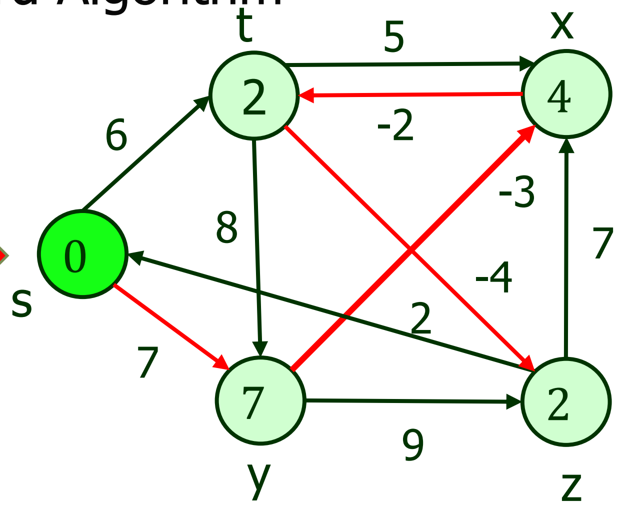
Iteration - 4 Edge no - 2

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-



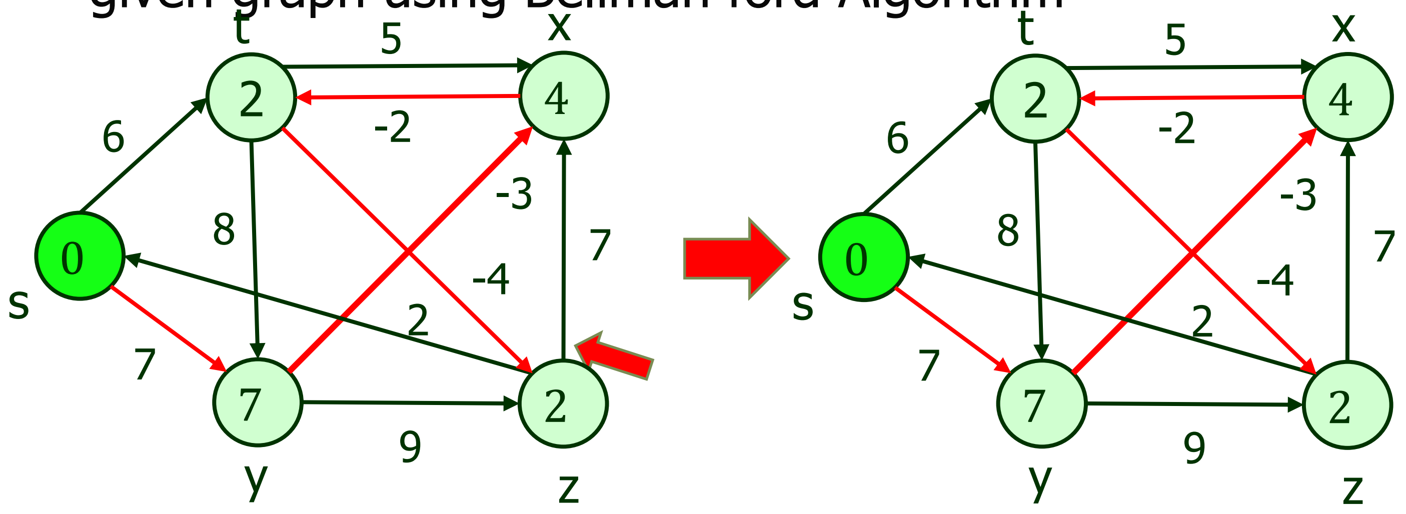


Iteration - 4　　　　Edge no - 3

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-

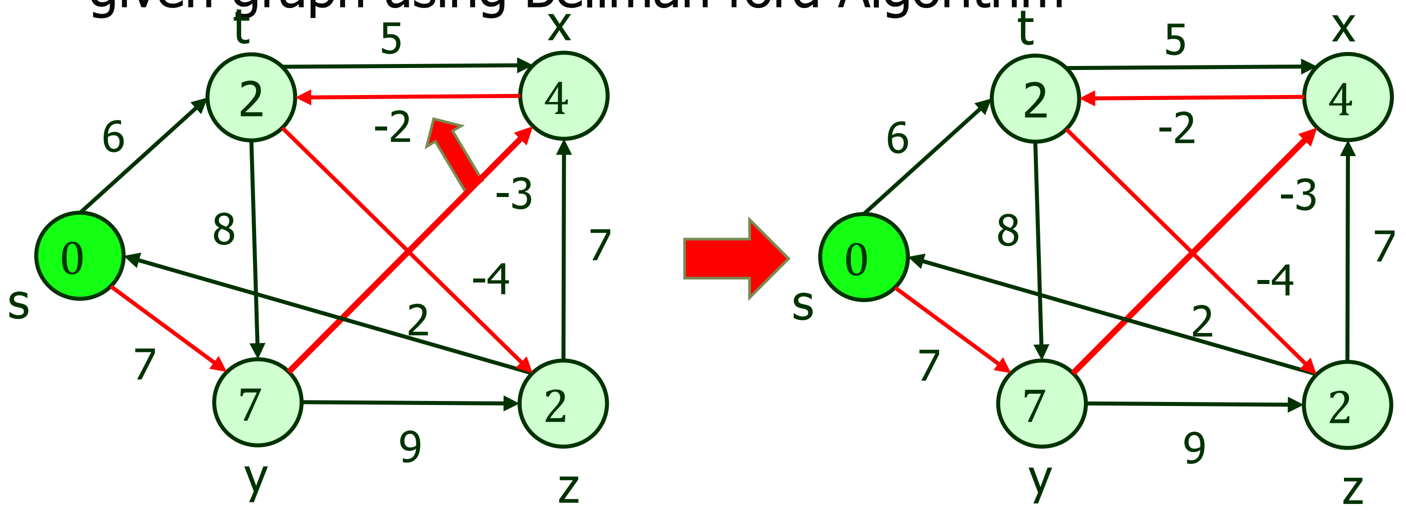


Iteration - 4 Edge no - 4

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-

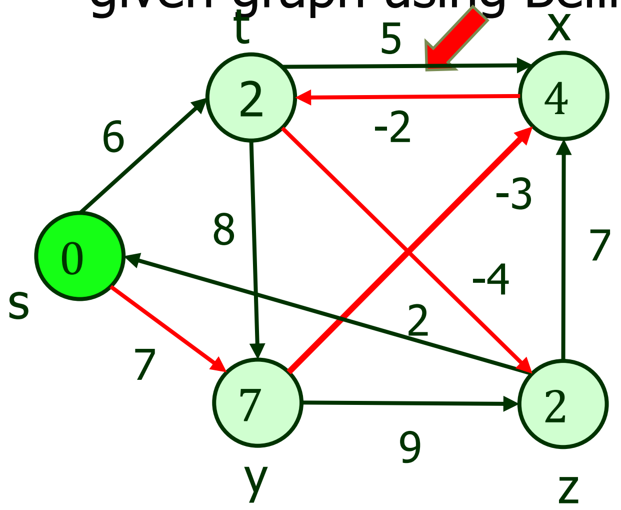


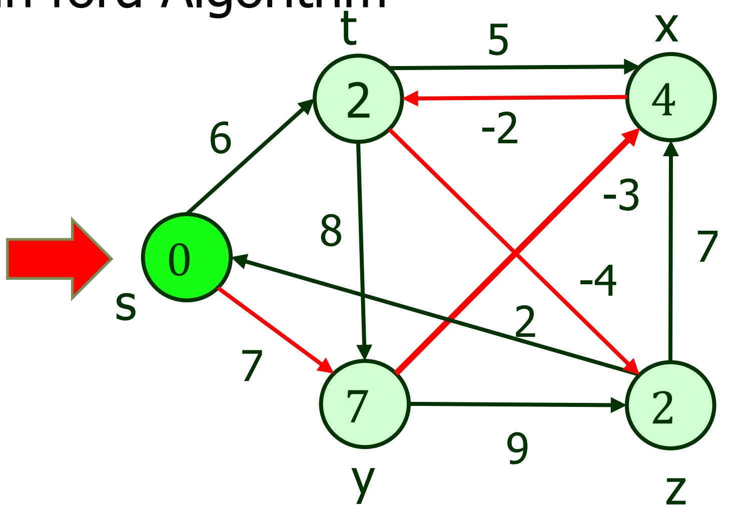
Iteration - 4 Edge no - 5

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-



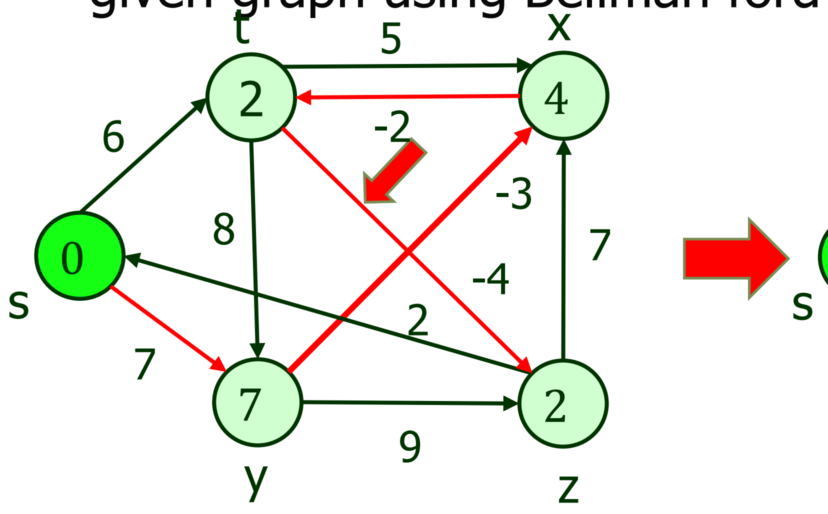


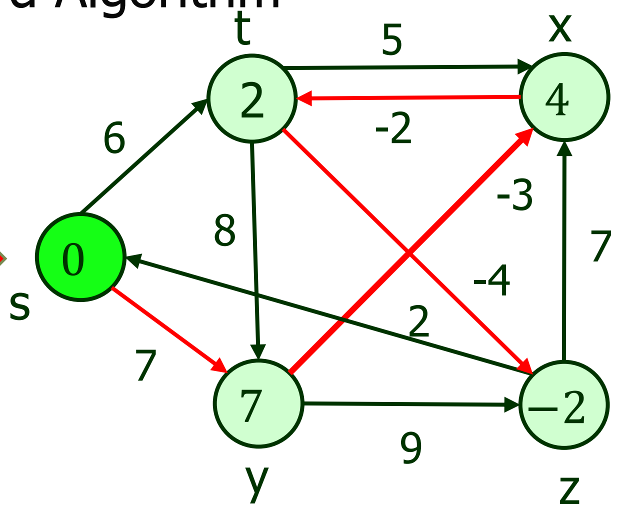
Iteration - 4 Edge no - 6

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-



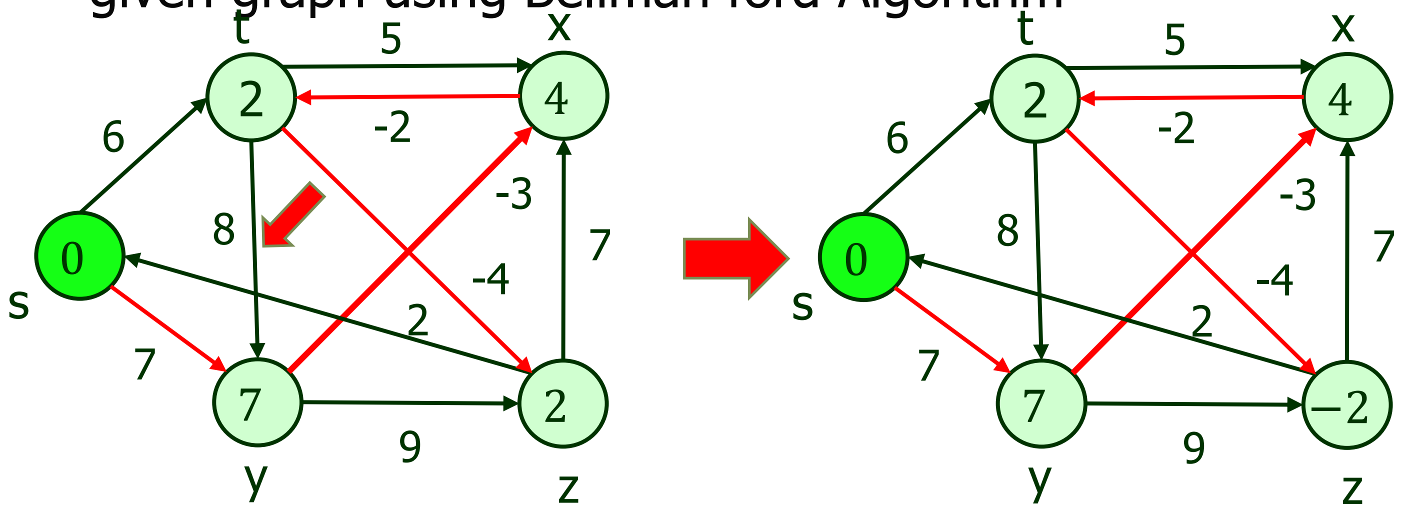


Iteration - 4 Edge no - 7

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-

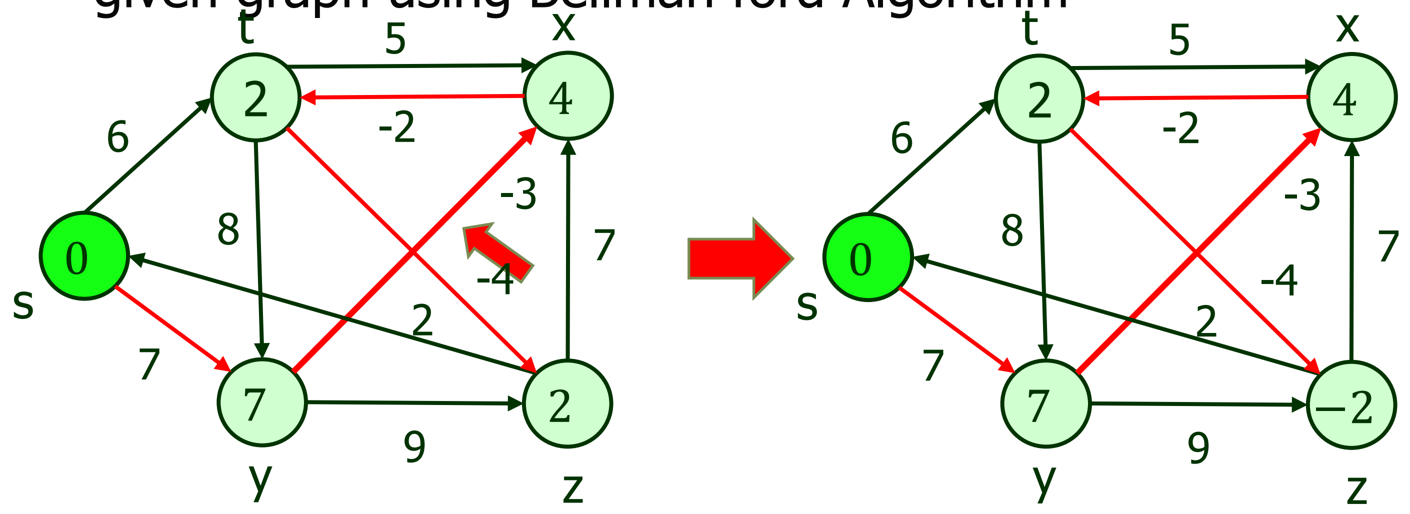


Iteration - 4 Edge no - 8

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-

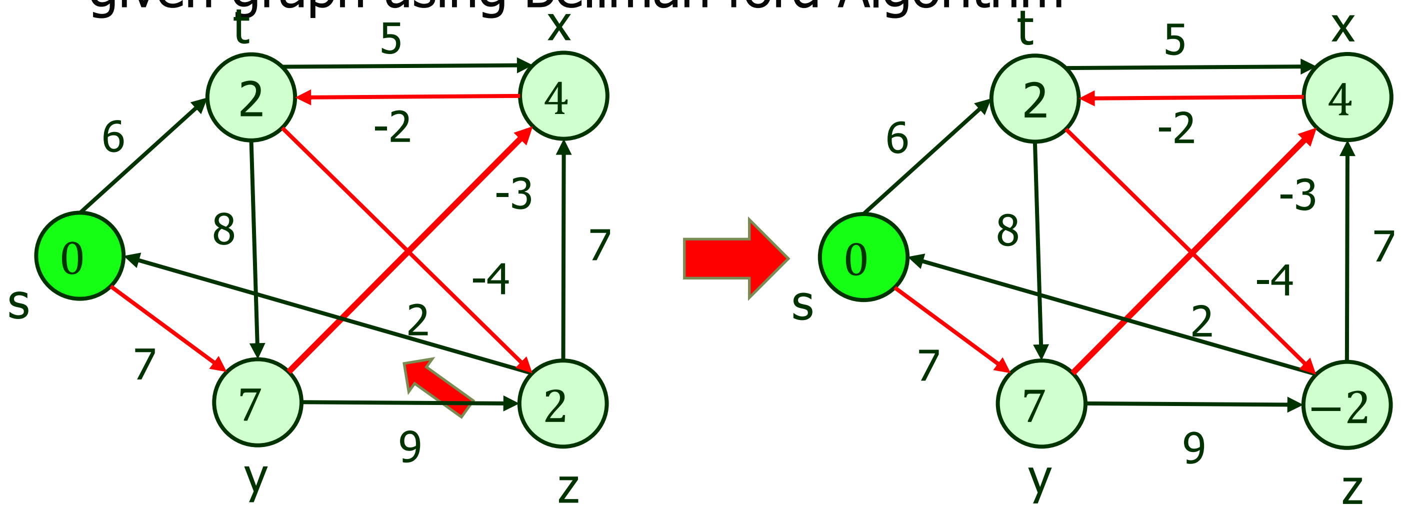


Iteration - 4 Edge no - 9

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

  Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-

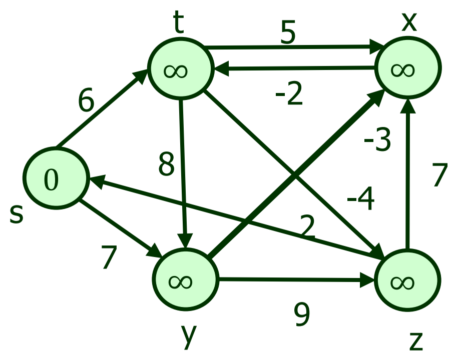


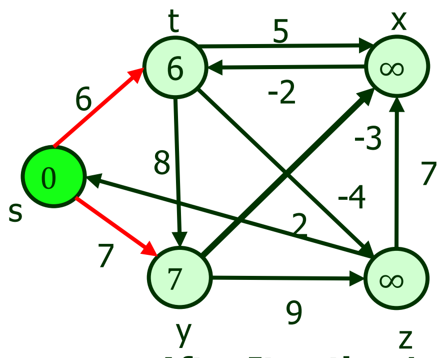
Iteration - 4 Edge no – 10

Greedy Algorithm

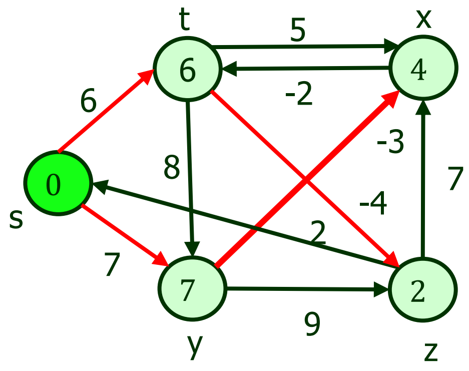
- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm**

Example 1: Construct the Single source shortest path for the given graph using Bellman ford Algorithm-

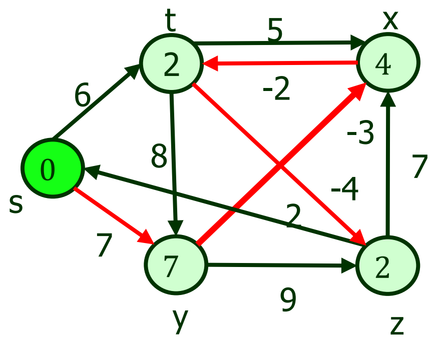




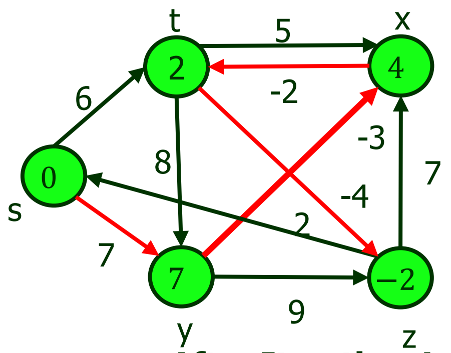
After Iteration 1



After Iteration 2



After Iteration 3



After Iteration 4

Greedy Algorithm

- **Problem 5: Single source shortest path**
- **Bellman Ford Algorithm (Algorithm)**

```
BELLMAN-FORD(G, w, s)
1   INITIALIZE-SINGLE-SOURCE(G, s)
2   for i ← 1 to |V[G]| - 1
3           do for each edge (u, v) ∈ E[G]
4                   do RELAX(u, v, w)
5   for each edge (u, v) ∈ E[G]
6           do if d[v] > d[u] + w(u, v)
7                   then return FALSE
8 return TRUE
```

Greedy Algorithm

- **Problem 5: Single source shortest path**
- **Bellman Ford Algorithm (Algorithm)**

```
INITIALIZE-SINGLE-SOURCE(G, s)
1   for each vertex v  V[G]
2           do d[v] ← ∞
3              π[v] ← NIL
4   d[s] ← 0
RELAX(u, v, w)
1 if d[v] > d[u] + w(u, v)
2   then d[v] ← d[u] + w(u, v)
3           π[v] ← u
```

Greedy Algorithm

- **Problem 5: Single source shortest path**
- **Bellman Ford Algorithm (Analysis)**

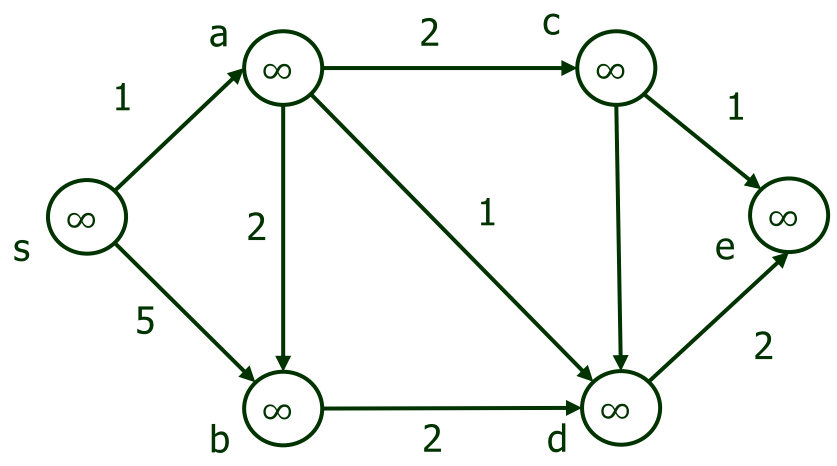
```
BELLMAN-FORD(G, w, s)
1   INITIALIZE-SINGLE-SOURCE(G, s)  → $\Theta(V)$
2   for i ← 1 to |V[G]| - 1
3           do for each edge (u, v) ∈ E[G]       $O(VE)$
4                   do RELAX(u, v, w)
5   for each edge (u, v) ∈ E[G]
6           do if d[v] > d[u] + w(u, v)           $O(E)$
7                   then return FALSE
8 return TRUE
```

$O(VE)$

Greedy Algorithm

- **Problem 5: Single source shortest path**
  - **Bellman Ford Algorithm(Self Practice)**

Example 2: Construct the Single source shortest path for the given graph using Bellman Ford Algorithm-



Greedy Algorithm

- **Problem 5: Knapsack Problem**

  Problem definition:

  - The Knapsack problem is an "combinatorial optimization problem", a topic in mathematics and computer science about finding the optimal object among a set of object .
  - Given a set of items, each with a weight and a profit, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total profit is as large as possible.

Greedy Algorithm

- **Problem 5: Knapsack Problem**

  Versions of Knapsack:

  - **Fractional Knapsack Problem:**

    Items are divisible; you can take any fraction of an item and it is solved by using **Greedy Algorithm.**

  - **0/1 Knapsack Problem:**

    Items are indivisible; you either take them or not and it is solved by using **Dynamic Programming(DP).**

Greedy Algorithm

- **Problem 5: Knapsack Problem**
  - **Fractional Knapsack Problem:**

    Given n objects and a knapsack with a capacity "M" (weight)

    - Each object i has weight w_i and profit p_i.
    - For each object i, suppose a fraction x_i , $0 \leq x_i \leq 1$, can be placed in the knapsack, then the profit earned is $p_i . x_i$.

Greedy Algorithm

- **Problem 5: Knapsack Problem**

Fractional Knapsack Problem:

The objective is to maximize profit subject to capacity constraints.
$i.e. maximize$

$\sum_{i=1}^{n} p_i x_i$ -------------------------------(1)

Subject to

$\sum_{i=1}^{n} w_i x_i \leq M$ ------------------------(2)

Where, $0 \leq x_i \leq 1$,

$p_i > 0$,

$w_i > 0$.

A feasible solution is any subset $\{x_1, x_2, x_3, \dots \dots, x_n\}$ satisfying (1) & (2).
An optimal solution is a feasible solution that maximize $\sum_{i=1}^{n} p_i x_i$.

Greedy Algorithm

- **Problem 5: Knapsack Problem**

Fractional Knapsack Problem(Implementation)

Fractional knapsack problem is solved using greedy method in the following steps-

Step-01:

- For each item, compute its (profit / weight) ratio.(i.e p_i/x_i)

Step-02:

- Arrange all the items in decreasing order of their (profit / weight) ratio.

Step-03:

- Start putting the items into the knapsack beginning from the item with the highest ratio.
- Put as many items as you can into the knapsack.

Greedy Algorithm

- **Problem 5: Knapsack Problem**

Fractional Knapsack Problem(Implementation)

Example 1: For the given set of items and knapsack capacity = 60 kg, find the optimal solution for the fractional knapsack problem making use of greedy approach.

Item	Weight	Profit
1	5	30
2	10	40
3	15	45
4	22	77
5	25	90

Greedy Algorithm

- **Problem 5: Knapsack Problem**

  Fractional Knapsack Problem(Implementation)

  Example 1: For the given set of items and knapsack capacity = 60 kg, find the optimal solution for the fractional knapsack problem making use of greedy approach.

  Step-01: Compute the (profit / weight) ratio for each item-

Greedy Algorithm

- **Problem 5: Knapsack Problem**

Fractional Knapsack Problem(Implementation)

Example 1: For the given set of items and knapsack capacity = 60 kg, find the optimal solution for the fractional knapsack problem making use of greedy approach.

Step-01: Compute the (profit / weight) ratio for each item-

Item	Weight	Profit	**Ratio**
1	5	30	6
2	10	40	4
3	15	45	3
4	22	77	3.5
5	25	90	3.6

Greedy Algorithm

- **Problem 5: Knapsack Problem**

Fractional Knapsack Problem(Implementation)

Example 1: For the given set of items and knapsack capacity = 60 kg, find the optimal solution for the fractional knapsack problem making use of greedy approach.

Step-02: Sort all the items in decreasing order of their value / weight ratio-

Item	Weight	Profit	**Ratio**
1	5	30	6
2	10	40	4
5	25	90	3.6
4	22	77	3.5
3	15	45	3

Greedy Algorithm

- **Problem 5: Knapsack Problem**

Fractional Knapsack Problem(Implementation)

Example 1: For the given set of items and knapsack capacity = 60 kg, find the optimal solution for the fractional knapsack problem making use of greedy approach.

Step-03: Start filling the knapsack by putting the items into it one by one.

Item	Weight	Profit	**Ratio**
1	5	30	6
2	10	40	4
5	25	90	3.6
4	22	77	3.5
3	15	45	3

Knapsack weight	Items in Knapsack	Cost
60	Ø	0

Greedy Algorithm

- **Problem 5: Knapsack Problem**

Fractional Knapsack Problem(Implementation)

Example 1: For the given set of items and knapsack capacity = 60 kg, find the optimal solution for the fractional knapsack problem making use of greedy approach.

Step-03: Start filling the knapsack by putting the items into it one by one.

Item	Weight	Profit	**Ratio**
1	5	30	6
2	10	40	4
5	25	90	3.6
4	22	77	3.5
3	15	45	3

Knapsack weight	Items in Knapsack	Cost
60	Ø	0

Greedy Algorithm

- **Problem 5: Knapsack Problem**

Fractional Knapsack Problem(Implementation)

Example 1: For the given set of items and knapsack capacity = 60 kg, find the optimal solution for the fractional knapsack problem making use of greedy approach.

Step-03: Start filling the knapsack by putting the items into it one by one.

Item	Weight	Profit	**Ratio**
1	5	30	6
2	10	40	4
5	25	90	3.6
4	22	77	3.5
3	15	45	3

Knapsack weight	Items in Knapsack	Cost
60	Ø	0
55	1	30

Greedy Algorithm

- **Problem 5: Knapsack Problem**

Fractional Knapsack Problem(Implementation)

Example 1: For the given set of items and knapsack capacity = 60 kg, find the optimal solution for the fractional knapsack problem making use of greedy approach.

Step-03: Start filling the knapsack by putting the items into it one by one.

Item	Weight	Profit	**Ratio**
1	5	30	6
2	10	40	4
5	25	90	3.6
4	22	77	3.5
3	15	45	3

Knapsack weight	Items in Knapsack	Cost
60	Ø	0
55	1	30

Greedy Algorithm

- **Problem 5: Knapsack Problem**

Fractional Knapsack Problem(Implementation)

Example 1: For the given set of items and knapsack capacity = 60 kg, find the optimal solution for the fractional knapsack problem making use of greedy approach.

Step-03: Start filling the knapsack by putting the items into it one by one.

Item	Weight	Profit	**Ratio**
1	5	30	6
2	10	40	4
5	25	90	3.6
4	22	77	3.5
3	15	45	3

Knapsack weight	Items in Knapsack	Cost
60	Ø	0
55	1	30
45	1, 2	70

Greedy Algorithm

- **Problem 5: Knapsack Problem**

Fractional Knapsack Problem(Implementation)

Example 1: For the given set of items and knapsack capacity = 60 kg, find the optimal solution for the fractional knapsack problem making use of greedy approach.

Step-03: Start filling the knapsack by putting the items into it one by one.

Item	Weight	Profit	**Ratio**
1	5	30	6
2	10	40	4
5	25	90	3.6
4	22	77	3.5
3	15	45	3

Knapsack weight	Items in Knapsack	Cost
60	Ø	0
55	1	30
45	1, 2	70

Greedy Algorithm

- **Problem 5: Knapsack Problem**

Fractional Knapsack Problem(Implementation)

Example 1: For the given set of items and knapsack capacity = 60 kg, find the optimal solution for the fractional knapsack problem making use of greedy approach.

Step-03: Start filling the knapsack by putting the items into it one by one.

Item	Weight	Profit	**Ratio**
1	5	30	6
2	10	40	4
5	25	90	3.6
4	22	77	3.5
3	15	45	3

Knapsack weight	Items in Knapsack	Cost
60	Ø	0
55	1	30
45	1,2	70
20	1,2,5	160

Greedy Algorithm

- **Problem 5: Knapsack Problem**

Fractional Knapsack Problem(Implementation)

Example 1: For the given set of items and knapsack capacity = 60 kg, find the optimal solution for the fractional knapsack problem making use of greedy approach.

Step-03: Start filling the knapsack by putting the items into it one by one.

Item	Weight	Profit	**Ratio**
1	5	30	6
2	10	40	4
5	25	90	3.6
4	22	77	3.5
3	15	45	3

Knapsack weight	Items in Knapsack	Cost
60	Ø	0
55	1	30
45	1,2	70
20	1,2,5	160

Greedy Algorithm

- **Problem 5: Knapsack Problem**

Fractional Knapsack Problem(Implementation)

Example 1:

Item	Weight	Profit	**Ratio**
1	5	30	6
2	10	40	4
5	25	90	3.6
4	22	77	3.5
3	15	45	3

Knapsack weight	Items in Knapsack	Cost
60	∅	0
55	1	30
45	1,2	70
20	1,2,5	160
0	1,2,5, frac(4)	230

Now, Knapsack weight left to be filled is 20 kg but item-4 has a weight of 22 kg.
Since in fractional knapsack problem, even the fraction of any item can be taken.
So, knapsack will contain the fractional part of item 4.(20 out of 22)
Total cost of the knapsack = 160 + (20/22) x 77 = 160 + 70 = 230 units

Greedy Algorithm

- **Problem 5: Knapsack Problem**

Fractional Knapsack Problem(Algorithm)

```
FRACTIONAL_KNAPSACK $(v, w, M)$
1. $load = 0$
2. $i = 1$
3. $While\ (load < M)\ and\ i \le n$
4.     $if\ w_i \le M - load$
5.         $take\ all\ of\ item\ i$
6.     $else\ take\ (M - load)/\ w_i\ of\ item\ i$
7.     $add\ what\ was\ taken\ to\ load(load = load + w_i)$
8.     i=i+1
```

(Note: Assume that the items are already sorted on the basis of ratio)

Greedy Algorithm

- **Problem 5: Knapsack Problem**

Fractional Knapsack Problem(Algorithm)

- The main time taking step is the sorting of all items in decreasing order of their (profit / weight) ratio.
- If the items are already arranged in the required order, then while loop takes $O(n)$ time.
- The average time complexity of Quick Sort is $O(n \log n)$.
- Therefore, total time taken including the sort is $O(n \log n)$.

Greedy Algorithm

- **Problem 5: Knapsack Problem**

Fractional Knapsack Problem(self practice)

Example 2: For the given set of items and knapsack capacity = 60 kg, find the optimal solution for the fractional knapsack problem making use of greedy approach.

Item	Weight	Profit
A	1	5
B	3	9
C	2	4
D	2	8

Greedy Algorithm

- **Problem 5: Knapsack Problem**

Fractional Knapsack Problem(self practice)

Example 3: Find the optimal solution for the fractional knapsack problem making use of greedy approach. Consider-

n = 3

M = 20 kg

(w1, w2, w3) = (18, 15, 10)

(p1, p2, p3) = (25, 24, 15)

Thank u