Lab 5

Aim : Implement the following Algorithms :To understand the difference between Dynamic Programming and Greedy Programming.

(1) String Edit Distance (Dynamic Programming)

(2) Prim's Algorithm for Minimum Spanning Tree (Greedy Algorithms)

---

2. Objective:

String Edit Distance (Dynamic Programming):

The edit distance (also known as Levenshtein distance) between two strings is the minimum number of single-character edits (insertions, deletions, or substitutions) required to transform one string into the other. The dynamic programming approach to solving the problem of edit distance is an efficient algorithm that uses a table to store the solutions to subproblems in order to avoid redundant work.

The objective of the dynamic programming approach to edit distance is to find the minimum number of edits required to transform one string into another by breaking the problem down into smaller subproblems and solving them in a bottom-up manner. The algorithm uses a table to store the solutions to subproblems, and uses these solutions to build up the solution for the original problem.

The dynamic programming approach to edit distance is typically faster than the naive recursive approach, which recalculates the same subproblems over and over again, because it avoids redundant work by caching the solutions to subproblems in a table. The time complexity of the dynamic programming algorithm is O(nm) where n is the length of the first string and m is the length of the second string. The space complexity is also O(nm) as it uses a table to store the solutions.

The algorithm is used in many fields such as natural language processing, bioinformatics, spell-checking and many more.

Prim's Algorithm for Minimum Spanning Tree (Greedy Algorithms):

Prim's algorithm is a greedy algorithm that is used to find the minimum spanning tree (MST) of a connected, undirected graph. The objective of the

algorithm is to find a set of edges that connects all the vertices in the graph, such that the total weight of the edges is as small as possible.

The basic idea behind Prim's algorithm is to start with an arbitrary vertex and grow the MST one vertex at a time, by adding the cheapest edge that connects a vertex in the MST to a vertex not yet in the MST. The algorithm uses a priority queue to keep track of the vertices not yet in the MST and their associated edge costs, and repeatedly selects the edge with the lowest cost that connects a vertex in the MST to a vertex not yet in the MST.

The time complexity of Prim's algorithm using a heap is O(ElogV) where E is the number of edges and V is the number of vertices. The space complexity is O(V) as it uses a priority queue to store the vertices not yet in the MST.

Prim's algorithm can be used to solve various problems such as building a network, designing a circuit and many more.

3. Description:

String Edit Distance (Dynamic Programming):

The algorithm begins by creating a table with rows and columns, where the rows represent the characters of the first string and the columns represent the characters of the second string. Each cell in the table represents the minimum number of edits required to transform the substring of the first string up to that row into the substring of the second string up to that column.

The algorithm then proceeds by filling in the cells of the table row by row, using the following rules:

1. If the characters in the corresponding positions of the two strings are the same, the edit distance is the same as the edit distance between the two substrings without that character.
2. If the characters in the corresponding positions of the two strings are different, the edit distance is the minimum of the three possible operations: a substitution, a deletion, or an insertion.
3. The substitution operation is represented by the cell diagonally up and left of the current cell, plus 1.
4. The deletion operation is represented by the cell above the current cell, plus 1.

5. The insertion operation is represented by the cell to the left of the current cell, plus 1.

The final edit distance is stored in the bottom-right corner of the table, and the algorithm can also be used to trace the optimal sequence of edits to transform the first string into the second string.

The time complexity of the dynamic programming algorithm is O(nm) where n is the length of the first string and m is the length of the second string. The space complexity is also O(nm) as it uses a table to store the solutions.

Prim's Algorithm for Minimum Spanning Tree (Greedy Algorithms):

Prim's Algorithm is a greedy algorithm that is used to find the minimum spanning tree from a graph. Prim's algorithm finds the subset of edges that includes every vertex of the graph such that the sum of the weights of the edges can be minimized.

Prim's algorithm starts with the single node and explores all the adjacent nodes with all the connecting edges at every step. The edges with the minimal weights causing no cycles in the graph got selected.

How does the prim's algorithm work?

Prim's algorithm is a greedy algorithm that starts from one vertex and continues to add the edges with the smallest weight until the goal is reached. The steps to implement the prim's algorithm are given as follows -

- First, we have to initialize an MST with the randomly chosen vertex.
- Now, we have to find all the edges that connect the tree in the above step with the new vertices. From the edges found, select the minimum edge and add it to the tree.
- Repeat step 2 until the minimum spanning tree is formed.

The applications of prim's algorithm are -

Prim's algorithm can be used in network designing.

- It can be used to make network cycles.
- It can also be used to lay down electrical wiring cables.

4. Algorithm:

String Edit Distance (Dynamic Programming):

Algorithm :

Begin

if initLen = 0, then

return finalLen

if finalLen := 0, then

return initLen

if initStr[initLen - 1] = finalStr[finalLen - 1], then

return editCount(initStr, finalStr, initLen – 1, finalLen - 1)

answer := 1 + min of (editCount(initStr, finalStr, initLen , finalLen - 1)),

(editCount(initStr, finalStr, initLen – 1, finalLen),

(editCount(initStr, finalStr, initLen – 1, finalLen - 1)

return answer

End

Time complexity : O (m*n)

Prim's Algorithm for Minimum Spanning Tree (Greedy Algorithms):

Algorithm :

1. Create a MST set that keeps track of vertices already included in MST.
2. Assign key values to all vertices in the input graph. Initialize all key values as INFINITE (∞). Assign key values like 0 for the first vertex so that it is picked first.
3. While the MST set doesn't include all vertices.
4. Pick vertex u which is not MST set and has minimum key value. Include 'u'to MST set.
5. Update the key value of all adjacent vertices of u. To update, iterate through all adjacent vertices. For every adjacent vertex v, if the weight of edge u.v is less than the previous key value of v, update key value as a weight of u.v.

MST-PRIM (G, w, r)

1. for each u ∈ V [G]
2. do key [u] ← ∞
3. π [u] ← NIL
4. key [r] ← 0
5. Q ← V [G]
6. While Q ? ⌀
7. do u ← EXTRACT - MIN (Q)
8. for each v ∈ Adj [u]
9. do if v ∈ Q and w (u, v) < key [v]
10. then π [v] ← u
11. key [v] ← w (u, v)

Time Complexity : O((V + E)log V)

5. Example:

String Edit Distance (Dynamic Programming):

Here is an example of using the dynamic programming approach to calculate the edit distance between two strings "kitten" and "sitting":

1. Create a table with rows and columns, where the rows represent the characters of the first string "kitten" and the columns represent the characters of the second string "sitting".

		s	i	t	t	i	n	g
	0	1	2	3	4	5	6	7
k	1							
i	2							
t	3							
t	4							
e	5							
n	6							

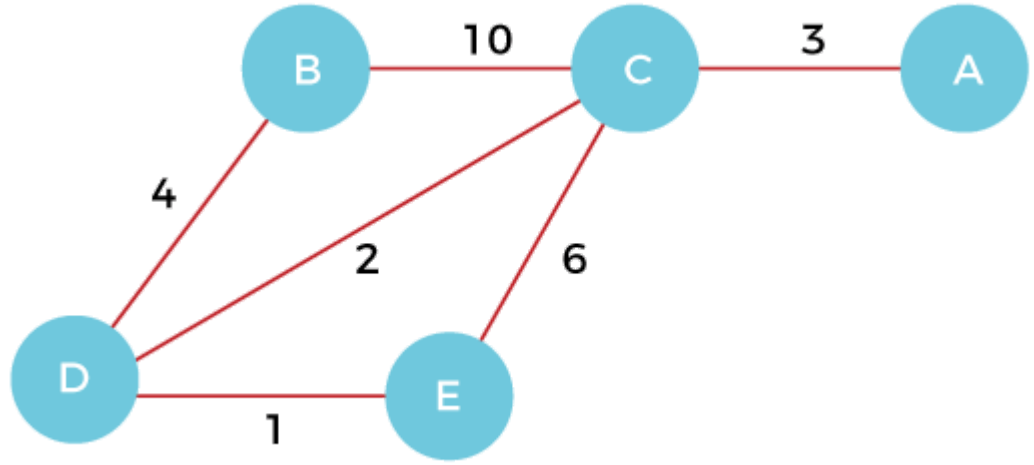
2. Fill in the first row and first column with the integers from 0 to the length of the second string and the first string respectively, representing the minimum number of operations needed to transform an empty string into the second string.

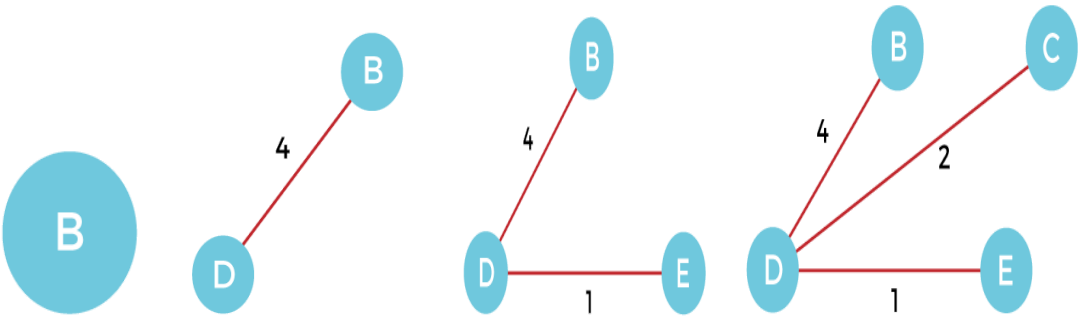
		s	i	t	t	i	n	g
	0	1	2	3	4	5	6	7
k	1	1	2	3	4	5	6	7
i	2	2	1	2	3	4	5	6
t	3	3	2	2	3	4	5	6
t	4	4	3	3	2	3	4	5
e	5	5	4	4	3	3	4	5
n	6	6	5	5	4	4	3	4

3. Fill in the rest of the cells using the following rules:
    - If the characters in the corresponding positions of the two strings are the same, the edit distance is the same as the edit distance between the two substrings without that character.
        - For example: the edit distance between "ki" and "si" is the same as the edit distance between "k" and "s" which is 1.
    - If the characters in the corresponding positions of the two strings are different, the edit distance is the minimum of the three possible operations: a substitution, a deletion, or an insertion.
        - For example: the edit distance between "kit" and "sit" is the minimum of the edit distance between "kit" and "sit" after a substitution operation, "kt" and "sit" after a deletion operation, and "kit" and "it" after an insertion operation.
4. The final edit distance is stored in the bottom-right corner of the table which is 4, meaning 4 operations are needed to transform "kitten" to "sitting"
5. To trace the optimal sequence of edits to transform the first string into the second string, one can retrace the path from the bottom-right corner of the table back to the top-left corner by checking from which cell the value in the current cell was derived.

Prim's Algorithm for Minimum Spanning Tree (Greedy Algorithms):

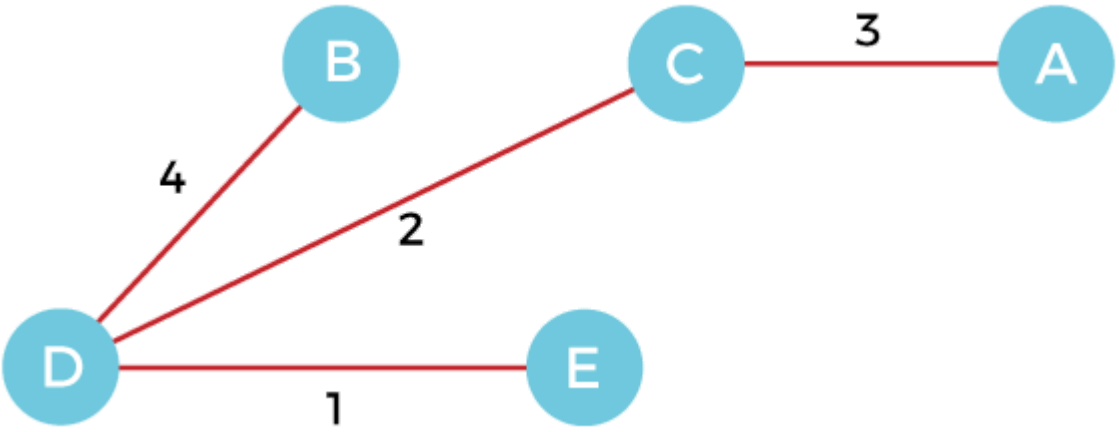
1. First, we have to choose a vertex from the above graph. Let's choose B.
2. Now, we have to choose and add the shortest edge from vertex B. There are two edges from vertex B that are B to C with weight 10 and edge B to D with weight 4. Among the edges, the edge BD has the minimum weight. So, add it to the MST.
3. Now, again, choose the edge with the minimum weight among all the other edges. In this case, the edges DE and CD are such edges. Add them to MST and explore the adjacent of C, i.e., E and A. So, select the edge DE and add it to the MST.
4. Now, select the edge CD, and add it to the MST.
5. Now, choose the edge CA. Here, we cannot select the edge CE as it would create a cycle to the graph. So, choose the edge CA and add it to the MST.





So, the graph produced in step 5 is the minimum spanning tree of the given graph. The cost of the MST is given below -

Cost of MST = 4 + 2 + 1 + 3 = 10 units



1.5. Implementation Notes:

String Edit Distance (Dynamic Programming):

Here are some implementation notes for the dynamic programming approach to the edit distance problem:

- Create the table: The first step is to create a 2D table with rows and columns, where the rows represent the characters of the first string and the columns represent the characters of the second string. Each cell in the table represents the minimum number of edits required to transform the substring of the first string up to that row into the substring of the second string up to that column.
- Initialize the table: Fill in the first row and first column with the integers from 0 to the length of the second string and the first string respectively, representing the minimum number of operations needed to transform an empty string into the second string.
- Fill in the rest of the cells: Use the previously described rules to fill in the rest of the cells using the values in the cells above, to the left and diagonally above it.
- The final edit distance is stored in the bottom-right corner of the table.
- Traceback: To trace the optimal sequence of edits to transform the first string into the second string, one can retrace the path from the bottom-right corner

of the table back to the top-left corner by checking from which cell the value in the current cell was derived.

- Time complexity: The time complexity of the dynamic programming algorithm is O(n*m) where n is the length of the first string and m is the length of the second string.
- Space complexity: The space complexity is also O(n*m) as it uses a table to store the solutions.
- Handling case sensitivity: Depending on the use case, it may be necessary to handle case sensitivity, by converting all characters to lowercase or uppercase before starting the algorithm.
- Handling non-letter characters: Depending on the use case, it may be necessary to handle non-letter characters, by removing them or treating them as special cases during the calculation of the edit distance.

Prim's Algorithm for Minimum Spanning Tree (Greedy Algorithms):

Here are some implementation notes for the Prim's algorithm using the greedy approach to find the minimum spanning tree (MST) of a connected, undirected graph:

- Choose a starting vertex: Begin by choosing an arbitrary vertex as the starting point for the MST.
- Initialize priority queue: Create a priority queue and insert all the vertices of the graph into it, along with their associated edge costs to the starting vertex.
- Select the edge with the lowest cost: Repeatedly select the edge with the lowest cost from the priority queue that connects a vertex in the MST to a vertex not yet in the MST.
- Add vertex to MST: Add the new vertex to the MST, and update the priority queue with the new edge costs to the vertices not yet in the MST.
- Repeat steps 3 and 4 until all vertices are in the MST.
- Time Complexity: The time complexity of Prim's algorithm using a heap is O(ElogV) where E is the number of edges and V is the number of vertices. The space complexity is O(V) as it uses a priority queue to store the vertices not yet in the MST.

- Optimizing the algorithm: One way to optimize the algorithm is to use a Fibonacci heap instead of a binary heap, which can reduce the time complexity to O(E+VlogV)
- Handling negative weights: The original Prim's algorithm can't handle negative weights. To handle negative weights, we can use Bellman-Ford algorithm or Dijkstra algorithm with a min-priority queue.
- Handling disconnected graph: If the graph is not connected the algorithm will not work properly, in this case, we should run Prim's algorithm on each connected component.
- Handling disconnected graph with negative weights: If the graph has negative weight edges, it is better to use Kruskal's algorithm instead of Prim's algorithm

1.6. Exercise:

a. Write Program to implement the String Edit Distance using Dynamic approach for given input and print out the out-put at each and every stage.
b. Write Program to implement the Prim's Algorithm for Minimum Spanning Tree (Greedy Algorithms) for given input and print out the out-put at each and every stage.

1.7. References:

https://www.geeksforgeeks.org/
https://www.javatpoint.com/
https://www.tutorialspoint.com/
https://www.w3schools.com/c