LA-II TE(B) 53

Pune District Education Association's
College Of Engineering
Manjari (Bk.), Hadapsar, Pune-412307.

Assignment NO :- 1

Title of Assignment :-
Implement depth first search algorithm & Breadth first search algorithm, use an Undirected graph & develop a recursive algorithm for Searching all the vertices of a graph or tree data structure.

PREREQUISITE :-
Basic knowledge of Array, list, Stack, Queue Graph, Tree etc.

Objective :-
In this ~~experiment~~, we will be able to do the following :
- To Understand Uninformed Search Strategies.
- To make use of Graph & Tree Data Structure for implementations of Uninformed Search Strategies.
- Study the various Graph traversal algorithms & the difference bet' them.
- Understand the BFS Graph traversal Using queues.
- Demonstrate knowledge of time Complexity & Space Complexity of performing a BFS on a given Graph.

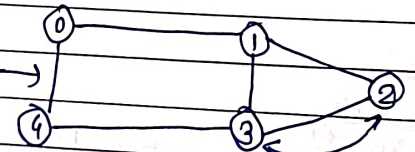
Outcome :-
Successfully able to find depth first search algorithm & Breadth first search algorithm

Software & Hardware Requirement :-
Open source C++ programming tool like GCC, Python, Java & Ubuntu.

Theory :-
Uninformed (Blind Search) Search :
- Uninformed search is a class of general purpose search algorithms which operates in brute force-way.
- It examines each node of the tree until achieves the goal node.
- Uninformed search algorithms do not have additional about state or search space other than how to traverse the tree & how to identify leaf & goal nodes, so it is also called blind search.
- Eg. DFS, BFS search algorithm.

Graphs defination :-
A graph is a pictorial representation of a set of objects where some pairs of objects are connected by links. The interconnected objects are represented by points termed as vertices & the links that connect the vertices are called edges. Pictorial Representation of Graph.

fig. Pictorial Representation of graph



Pune District Education Association's
College Of Engineering
Manjari (Bk.), Hadapsar, Pune-412307.

Types of graph :-
- Undirected Graph :- Directions are not given to edges.
- Directed Graph :- Directions are given to edges with

Theory of Graph traversal techniques
- In computer science, graph traversal (also known as graph search) refers to the process of visiting (checking and/or updating) each vertex in a graph. Such traversals are classified by the order in which the vertices are visited. Tree traversal is a special case of graph traversal.

Techniques of graph traversal
- DFS - A depth-first search (DFS) is an algorithm for traversing a finite graph. DFS visits the child vertices before visiting the sibling vertices; that is, it traverses the depth of any particular path before exploring its breadth. A stack (often the program's call stack via recursion) is generally used when implementing the algorithm.

- BFS - A breadth-first search (BFS) is another technique for traversing a finite graph. BFS visits the neighbor vertices before visiting the child vertices, & a queue is used in the search process. This algorithm is often used to

Find the Shortest Path from one vertex to

DFS

- Depth-first search is a recursive algorithm traversing a tree or graph data structure
- It is called depth-first search because it starts from the root node & follows each path to its greatest depth node before moving to the next path.
- DFS uses a stack data structure for its implementation

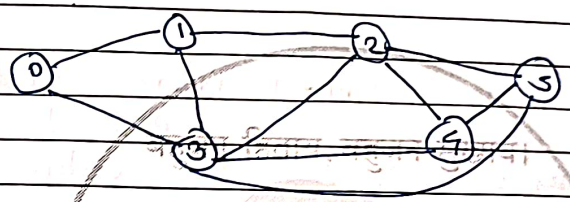
Depth first search (DFS) Algorithm

Depth first search (DFS) algorithm starts the initial node of the graph G, & then go deeper & deeper until we find the goal node or the node which has no children. The algorithm then backtracks from the dead end towards the most recent node that is yet to be completely explored. The data structure which is being used in DFS is Stack.

- Step 1 :- Start by putting any one of the graph vertices on top of a stack. (Take as source node of DFS).
- Step 2 : Take the top item of the stack & add its visited as 1.
- Step 3 : Create a list of the vertex's adjacent nodes. Add the ones whose visited is 0 to the top of stack.
- Step 4 : Keep repeating steps 2 & 3 until the stack is

Pune District Education Association's
College Of Engineering
Manjari (Bk.), Hadapsar, Pune-412307.

An example which explain DFS Algorithm



Initial Condition :-

Visited	0	0	0	0	0	0
	0	1	2	3	4	5

Current Stack

DFS Sequence

Step 1 :-

Start with node 0

Node 0 is pushed into stack & its visited value set to 1

Visited	1	0	0	0	0	0
	0	1	2	3	4	5

Current Stack | 0 |

DFS Sequence

Step 2 :

Node 0 is popped from stack & its adjacent nodes and 3 are pushed into stack & visited value set to

Visited	1	1	0	1	0	0
	0	1	2	3	4	5

Current Stack

3
1

DFS sequence : 0

Step 3 :
Node 3 is popped from stack & its nodes 2, 4 & 5 are pushed into stack because their visited value is 0, Set visited value to 1

Visited

1	1	1	1	1	1
0	1	2	3	4	5

Current Stack

5
4
2
1

DFS sequence : 0, 3

Step 4 :
Node 5 is popped from stack. No adjacent node to add in stack.

Visited

1	1	1	1	1	1
0	1	2	3	4	5

Current Stack

4
2
1

DFS Sequence : 0, 3, 5

Step 5 :-
Node 4 is popped from stack Adjacent Nodes visited value is already 1 So on need to push it into stack.

visited

1	1	1	1	1	1
0	1	2	3	4	5

Current Stack

2
1

DFS sequence : 0, 3, 5, 4

Step 6 :-
Node 2 is popped from stack, Adjacent Nodes visited value is Already 1, So no Need to push it into stack.

Visited

1	1	1	1	1	1
0	1	2	3	4	5

Current Stack

1

DFS Sequence : 0, 3, 5, 4, 2

Step 7 :
Node 1 is popped from stack & No adjacent nodes are remain unvisited
no ~~stack~~ nodes are pushed into stack
& stack became empty we can stop

visited

1	1	1	1	1	1
0	1	2	3	4	5

Current Stack

DFS Sequence : 0, 3, 5, 4, 2, 1

- DFS Advantages :-
  - It requires less memory as it only needs to store a stack of nodes on the path from root to current nodes.
  - It takes less time to reach goal.
- DFS disadvantages :-
  - There is possibility many states keep reoccuring.
  - Sometimes it may go to the infinite loop
- Application :-
  DFS Algorithm can be used implement the topological sorting.
- It can be used to find the paths between two vertices.
- DFS algorithm is also used for one solution puzzles.

BFS algorithm :-

The algorithm starts with examining the source node & all of its neighbour

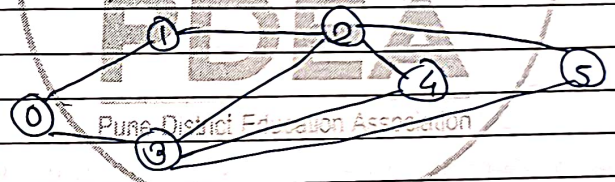
Queues Defination :-

A queue is a linear structure which follows a particular order in which the operations are performed

Pune District Education Association's
College Of Engineering
Manjari (Bk.), Hadapsar, Pune-412307.

The algorithm starts with the source node & of its neighbours. In the next step, the neighbours of the nearest node of the source node are explored. The algorithm then explores all neighbours of all the nodes & ensures that each node is visited exactly once & no node is visited twice.

An example which explains BFS Algorithm



Initial Condition

visited : | 0 | 0 | 0 | 0 | 0 | 0 |
0 1 2 3 4 5

Curr. Queue | - |

BFS Sequence | - |

Step 1 :

Node '0' is selected as source node of BFS & node '0' is enqueued to queue

visited | 1 | 0 | 0 | 0 | 0 | 0 |
0 1 2 3 4 5

Curr. Queue | 0 |

BFS. Sequence | - |

Step 2:
Node '0' is dequeued from Queue & its adjacent nodes are enqueued into queue i.e. 1, 3, by changing visited value to '1'.

Visited | 1 | 1 | 0 | 1 | 0 | 0 |
0 1 2 3 4 5

Curr. Queue | 1 | 3 |

BFS Sequence : | 0 |

Step 3:
Node "1" is dequeued from Queue & adjacent nodes are enqueued into queue i.e. 2, 3 but only node 2 is having visited value 0 but node 3 is already having visited value 1.

Visited | 1 | 1 | 1 | 1 | 0 | 0 |
0 1 2 3 4 5

Curr. Queue | 3 | 2 |

BFS Sequence | 0 | 1 |

Step 4 & 5 :-
Node 3 is dequeued from queue & adjacent nodes 4, 5 enqueued into queue & set visited & Also 2 is dequeued from queue its adjacent nodes already having visited value 1. So No enqueued node 3, 4, 5 is queue

Visited | 1 | 1 | 1 | 1 | 1 | 1 |
0 1 2 3 4 5

Curr. Queue |

BFS Sequence | 0 | 1 | 3 | 2 | 4 | 5 |

As queue is empty, BFS is done on node '0'
Nodes visited in BFS Sequence : 0, 1, 3, 2, 4, 5

Pune District Education Association's
College Of Engineering
Manjari (Bk.), Hadapsar, Pune-412307.

BFS Application :
- Path & minimum Spanning Tree for Unweighted graph.
- In peer to peer Network like Bit Torrent, Breadth first Search is used to find all neighbour nodes.
- GPS Navigation System, Breadth first Search is used to find all neighbouring Location.

Time Complexity :
$T(B) = 1 + b^2 + b^3 + \dots b^d = O(b^d)$

Space Complexity :
Space Complexity of BFS is given by the memory size of frontier which is $O(b^d)$

* Diff. bet' BFS & DFS :

- BFS is a vertex - based algorithm DFS is an edge - based algorithm.
- Queue data structure is used in BFS the other hand, DFS uses stack recursion.
- Memory space is efficiently utilized DFS while space utilization in BFS effective.
- DFS constructs narrow & long trees whereas BFS constructs wide & short tree.

Conclusion :-

In this way we have studied informed search strategy, how to implement depth first search algorithm Breadth first search algorithm, we an Undirected graph & develop a recursive algorithm for searching all the vertices of a graph or tree data structure.

Pune District Education Association's
College Of Engineering
Manjari (Bk.), Hadapsar, Pune-412307.

Assignment no : 2

Title of Assignment :-

Implement A Star algorithm for 8 puzzle game search problems

Prerequisite :-

Basic knowledge of graph tree, informed search, Uniformed search, best first search etc.

Objectives :-

To this experiment, we will be able to the following :-

- To understand informed we will be able to the search strategies.
- To make use of Graph & Tree data structure for implemetation of informed search strategies
- Study how A Star Algorithm is useful for implementation of 8 puzzle game search pro.

Outcome :-

Successfully able to find depth first search & Breadth first search algorithms

Software & Hardware Requirement :-

Open Source C++ Programming tool like G++/GCC, Python, Java & Ubuntu.

Theory :-

Informed Search :-

- Informed Search algorithm contains an array of knowledge such how far we are from the goal, path cost, how to reach the goal node, etc.
- This knowledge helps agent to explore less of the search space and find more efficiently the path which leads to a goal node.
- Informed search algorithm uses the idea of a heuristic, so it is called Heuristic search.

Heuristic function :-

- Heuristic is a function which is used in Informed Search, and it helps to choose the most promising path.
- It takes the current state of the agent as its input and produces the estimation of how close the agent is from the goal.
- The heuristic method, however, might not always give the best solution, but it is guaranteed to find a good solution in reasonable time.
- Heuristic function estimates how close a state is to goal. It represented by h(n), and it calculates the cost of an optimal path between the pair of states.
- The value of the heuristic function is always positive.

A* Search Algorithm.

* Search is the most commonly known form of best-first search.

- It uses the heuristic function h(n) + cost to reach the node n from the start state g(n).
- It has combined features of UCS & greedy best-first search, by which it solves the problem efficiently.
- A* search algorithm finds the shortest path through the search space using the heuristic function.
- This search algorithm expands less search tree & provides optimal results faster.

In A* search algorithm, we use search heuristic as well as the cost to reach the node. Hence we can combine both costs as following & this sum is called fitness number.

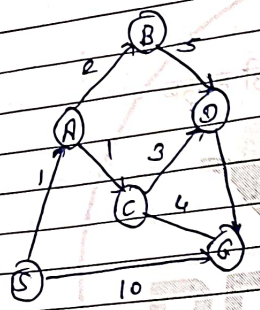
$$f(n) = g(n) + h(n)$$

- $f(n)$: Estimated cost of the cheapest path
- $g(n)$: cost of each node n from start node
- $h(n)$: cost of each node to goal node.

In this example, we will traverse the given graph using the A* algorithm. The heuristic value of all states is given in the below table so we will calculate the f(n) of each state using the formula, $f(n) = g(n) + h(n)$,

where g(n) is the cost to reach any node from start state.

Here we will use Open & closed list.



State	h(n)
S	5
A	3
B	4
C	2
D	6
E	0

Initialization : { (S, 5) }

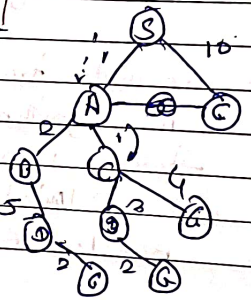
Iteration 1 : { (S --> A, 4) (S --> G, 10) }

Iteration 2 : { (S --> A --> C, 4) (S --> A --> B, 7) }

Iteration 3 : { (S --> A --> C --> G, 6), (S --> A --> C --> D, 11) (S --> A --> B, 7), (S --> G, 10) }

Iteration 4 : we will give the final result S --> A --> C --> G is provides the optimal path with cost 6.

Hence !!-



Pune District Education Association's

College Of Engineering

Manjari (Bk.), Hadapsar, Pune-412307.

* A* Search Algorithm . Advantages :-

- A* Search algorithm is the best algorithm than other search algorithm.
- A* search algorithm is optimal & complete.
- This algorithm can solve very complex problems.

* A* Search Algo. Advantages :-

- A* search algorithm has some complexity issues.
- The main drawback of A* is memory requirement as it keeps all generated nodes in the memory, so it is not practical for various Scale problems.

Time Complexity :-

The time complexity is O(b^d), where 'b' is the branching factor.

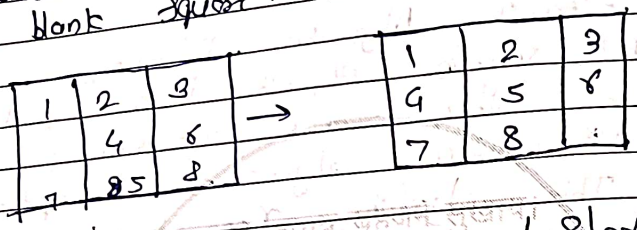
Space Complexity :-

The space complexity of A* search algorithm is O(b^d)

8 puzzle Algorithm :-

The 8 Puzzle problem is a puzzle invented & popularized by Noyes palmer Chapman in the 1870's. It is played on a 3-by-3 grid with 8 square block labelled 1 through 8 & a blank square. Your goal is to rearrange the block so that they are in order. You are permitted to slide

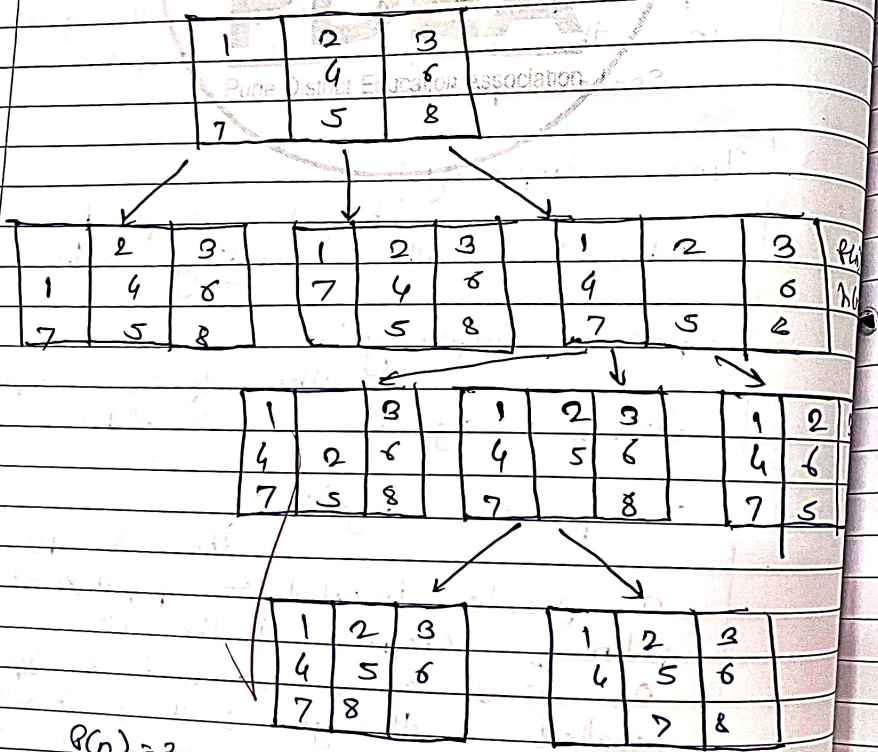
blocks horizontally or vertically into the blank space.



Initial State → goal State.

Solution A* algorithm :

h(n) = no. of misplaced tiles :



f(n) = 3
h(n) = 0
final state (Goal)

f(n) = 5
h(n) = 2

Conclusion :-

In this way we have studied informed search strategy, how to calculate heuristic function & implementation of 8 puzzle game search problem using A Star algorithm.

Practical No :- 3

- Title of Assignment :-
  Implement Greedy search algorithm for Selection Sort.
- Prerequisite :-
  Basic knowledge of Greedy algorithm & Sorting Concept.
- Objective :-
  In the experiment, we will be able to do the following :
  - Study how Selection Sort works greedy search algorithm.
- Outcome :-
  Successfully able to Sort, Unsorted list of numbers
- Software & hardware Requirement :-
  Open Source c++ programming tool like G++ / Gcc, python, Java & Ubuntu.

Theory :-

Selection Sort is a Comparison-based Sorting algorithm. It Sort an array be repeatedly Selecting the Smallest (or largest)

element from the Unsorted portions & ... it with the first Unsorted element. Th... process continues Until the entire ... Sorted.

1) first we find the smallest among remaining elements (or second). & Swap ... first element. This way we get the ... element at its correct position.

2) Then we find the smallest among re... element (or second smallest) & swap ... the second element.

3) We keep doing this Until we get ... element moved to correct position.

In Selection Sort, we take this simplest, ... initiative approach to sort an array. Choose the smallest number, placing it in first position. Then choose the next smallest number out of remaining elements & place it in the second position & so on ... till the end.

The Selection Sort algorithm sort maintains two parts :

① The first part that is already sorted
② The second part is yet to be sorted.

The algorithm works by repeatedly finding the minimum element from the Unsorted

part & putting it to at the end of the Sorted part.

Initiation behind the Algorithm :-

Initial Unsorted Array | 15 | 21 | 19 | 5 | 23 |

Smallest element out of these grp of element : 05

| 15 | 21 | 19 | 5 | 23 → 5 |

Smallest ele. out of these grp of element : 15

| 15 | 21 | 19 | 23 → 5 | 15 |

Smallest ele. out of these grp of ele : 19

| 21 | 19 | 23 → 5 | 15 | 19 |

Smallest ele. out of these grp. of ele : 20

| 21 | 23 → 5 | 15 | 19 | 21 |

Smallest ele out of these grp of ele. 23

| 23 → 5 | 15 | 19 | 21 | 23 |

Selection Sort algorithm :

Initial Unsorted Array

| 34 | 20 | 3 | 14 |

i=1

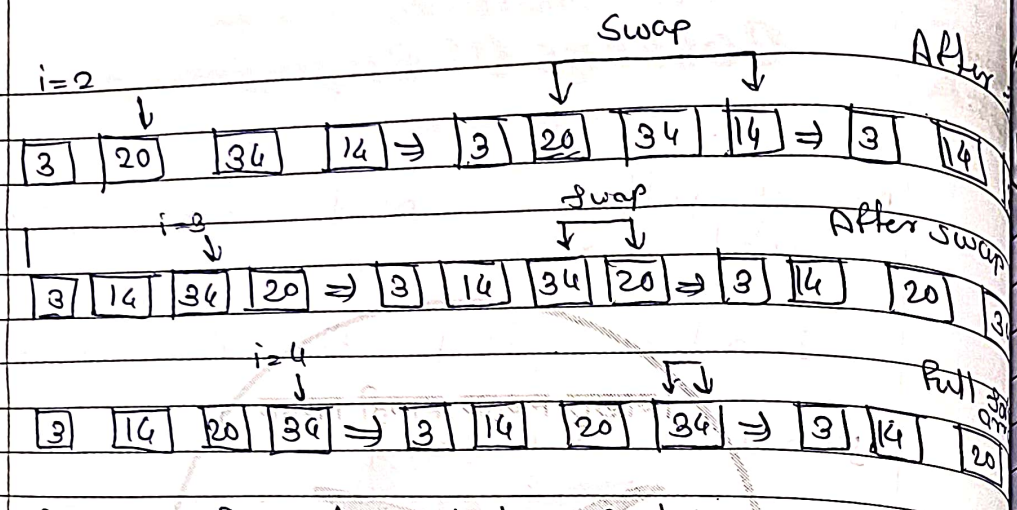
↓

| 34 | 20 | 3 | 14 → |

Swap.

| 34 | 20 | 3 | 14 → 3 | 20 | 34 |

Array After Sorting



Running Time of Selection Sort :-

Let's assume we are sorting N elements of a given array using Selection sort.

- To complete one iteration, we traverse a portion of the array (from index i to end) exactly once. Since the longest length we ever traverse in any given iteration is N, time complexity of completing one iteration is O(N).
- In selection sort, we can N iterations, each of which takes O(N) time. Hence overall time complexity become O(N*N).

Space Complexity of Selection Sort :

While swapping two elements, we need some extra space to store temporary value. Other than that, the sorting can be done-in-place. Hence the space complexity is O(1) or constant space.

Pune District Education Association's

College Of Engineering

Manjari (Bk.), Hadapsar, Pune-412307.

What is Stable Sort Algorithm :-

A sorting Algorithm to be stable if two objects with equal keys appear in the same order in sorted output as they in the input unsorted array.

Unsorted Array

56	10	23	9	34	23*

9	10	23	23*	34	56

Is selection Sort Stable?

Yes, Selection sort is a stable sorting algorithm. when looking for the smallest element we choose the element with lower index in case there are two or more equal elements that are the smallest element in the array. This makes sure that we preserve the relative ordering between equal elements.

Conclusion :-

In this way we have studied how to sort, unsorted list of numbers using selection sort.

Practical no :- 4

Title of Assignment :-
Implement Greedy Search algorithm for Prim's minimal spanning tree algorithm.

Prerequisite :- बहुजन हिताय, बहुजन सुखाय।
- Basic knowledge of Graph theory & Spanning Trees
- Understanding of Greedy Algorithms & Priority Queues.

Objective :-
- In this experiment we will be able to :
- Understand the concept of minimum Spanning Tree (MST)
- Compare ~~time~~ ~~Com~~ Prim's algorithm with Kruskal's Algorithm.

Outcome :-
- Successfully able to implement & find the Minimum Spanning Tree (MST) using Prim's Algorithm.
- Understand the greedy strategy used in MST algorithms.
- Compare time complexities of different MST algorithm.

Software & Hardware Requirement :-

- Open-Source programming tools such as C++, Java.
- Operating System : Window / Linux / Ubuntu.

Theory :-

Prim's Algorithm

- Prim's Algorithm is a greedy algorithm that grows the MST from a single starting vertex.
- It adds the smallest possible edge that connects a new vertex to the growing MST.

Working of Prim's Algorithm

- Start from an arbitary vertex (usually vertex 0).
- Add the smallest edge that connects a new vertex to the MST.
- Repeat Until all vertices are included in the MST.

Steps of Prim's Algorithm :-

① Initialize :-
- select an arbitary vertex as the starting node.
- Create a visited array to keep track of nodes in the MST.
- Use a priority queue (min-heap) to pick the smallest edge.

② Iterate Until MST is formed :
- Pick the edge with the smallest weight that connects a new vertex.
- mark the newly added vertex as visited.
- Add its adjacent edges to the priority queue.

Pune District Education Association's
College Of Engineering
Manjari (Bk.), Hadapsar, Pune-412307.
Accredited by NAAC

② Repeat Until All vertices are Added :

let's Consider a graph with 5 vertices (A, B, C, D, E) & the following weighted edges :

Edge	Weight
A-B	2
A-C	3
B-C	1
B-D	4
C-D	5
C-E	8
D-E	7

Step-by-Step Execution of Prim's Algorithm

1) Start with any vertex (let's start with A).
2) Choose the smallest edge from A → (A,B) = 2
3) Add B to MST.
4) Choose the smallest edge from {A,B} → (B,C) = 1
5) Add C to MST.
6) Choose the smallest edge from {A,B,C} → (A,C) = 3 (already added, Skip) → (C,D) = 4.
7) Add D to MST.
8) Choose the smallest edge from {A,B,C,D} → (C,E) = 8
9) Add E to MST

find MST (Minimum Spanning tree)

Edges included in the MST :

- A - B (2)
- B - C (1)
- C - D (4)
- C - E (6)

- Total Cost of MST = 2 + 1 + 4 + 6 = 13

Prim's Algorithm Advantages :-

- Works well with dense graphs (When the num of edges is large).
- Always finds the optimal solution (i.e., MST w the smallest weight).
- Efficient with priority queues ($O(E \log V)$ using a min-heap).

Prim's Algorithm Disadvantages :-

- Not suitable for disconnected graphs (does not work if all nodes are not reachable)
- More complex than Kruskal's Algorithm for sparse graphs.

Conclusion :-

In this way we have studied how Prim's Algorithm is a greedy method for finding the minimum spanning tree (MST). It uses a priority queue (min-heap) for efficient edge selection. It is particularly efficient for dense graphs.

Practical no :- 5

Title of Assignment :-
Implementation of kruskal's Minimal Spanning Tree (MST) algorithm.

Prerequisite :-
- Basic knowledge of Graph theory & Spanning tree
- Understand of Greedy Algorithm & Disjoint Set (Union-Find) Data Structure.

Objective :-
In this experiment, we will be able to :
- Understand the concept of minimum Spanning tree
- Implement kruskal's Algorithm for Constructing the MST of a given graph
- Compare kruskal's algorithm with Prim's Algorithm

Outcome :-
- Successfully able to implement & find the Minimum Spanning Tree (MST) using kruskal's algorithm
- Understand the greedy strategy used in MST algorithm.
- Compare time Complexities of different MST algorithms.

Software & Hardware Requirement :

- Open-Source programming tools Such as C++, Java.
- Operating System : Windows / Linux / Ubuntu.

Theory :

Minimum Spanning Tree (MST)

- A minimum Spanning tree (MST) is a subset of edges of a weighted graph that :
  1) forms a tree (i.e. it is connected & has no cycles)
  2) Includes all vertices of the graph.
  3) Has the minimum possible total edge weight.
- There are two major algorithms to find MST :
  1) Prim's Algorithm (greedy algorithm using priority queue).
  2) Kruskal's Algorithm (greedy algorithm using edge Sorting & Union-Find).

Kruskal's Algorithm :

- Kruskal's Algorithm is a greedy algorithm that builds the MST by adding edges in increasing order of weight while ensuring no cycles are formed.
- It uses the Union-Find (Disjoint Set) data Structure to check for cycles efficiently.

Working of Kruskal's Algorithm :-

1) Sort all edges in ascending order of weight.
2) Pick the smallest edge & check if adding it forms a cycle using the Union-Find data Structure.
3) If adding the edge does not form a cycle, add it to the MST.
4) Repeat Until (V-1) edges are included in the MST (Where V is the number of vertices)

Graphs Representation with weight :-

Edge	weight
A-B	2
A-C	3
B-C	1
B-D	4
C-D	5
C-E	6
D-E	7

Step-by-Step Execution of Kruskal's Algorithm :

1) Sort all edges by weight :
   (B, C, 1) → (A, B, 2) → (A, C, 3) → (B, D, 4) → (C, D, 5) → (C, E, 6)
   → (D, E, 7)

2) Pick the smallest edge → (B,C,1) (no cycle, add)
3) Pick the next smallest edge → (A,B,2) (no cycle, add)
4) Pick the next smallest edge → (A,C,3) (no cycle, add)
5) Pick the next smallest edge → (B,D,4) (no cycle, add)
6) Pick the next smallest edge → (C,D,5) (cycle detected)
7) Pick the next smallest edge → (C,E,6) (no cycle, add)

Final MST (Minimum Spanning Tree).

Edges included in the MST:

- B - C (1)
- A - B (2)
- A - C (3)
- B - D (4)
- C - E (6)

- Total Cost of MST = 1+2+3+4+6 = 16

Advantages of Kruskal's Algorithm:

- Works well for sparse graphs (fewer edges).
- More efficient for edges list representation of graphs
- Easy to implement using sorting & Union-Find.

Disadvantages of Kruskal's Algorithm:

- Slower for dense graphs compared to prim's Algorithm
- Requires sorting of edges, which adds O(E log E) complexity.

Comparison: Prim's vs Kruskal's Algorithm

Feature	Prim's Algorithm	Kruskal's Algorithm
Approach	Greedy, vertex-based	Greedy, edge-based.
Data Structure	Priority Queue (Min-heap)	Disjoint Set (Union-Set)
Complexity	O(E log V) (with Min-heap)	O(E log E) (with sorting)
Best for	Dense Graphs	Sparse Graphs.

Conclusion:-

In this way we have studied how kruskal's Algorithm is a greedy method for finding the Minimum Spanning Tree (MST). It uses edge sorting & the Disjoint Set (Union-Find) data structure to efficiently construct the MST. It is most efficient for sparse graphs, while Prim's Algorithm is better for dense graphs.

Pune District Education Association's
College Of Engineering
Manjari (Bk.), Hadapsar, Pune-412307.
Accredited by NAAC

Practical Assignment No :- 6

Title of Assignment :-
Implementation of Dijkstra's Minimal Spanning Tree Algorithm

Prerequisite :-
- Basic knowledge of Graph theory & weighted Graphs
- Understanding of Greedy Algorithms & Priority Queue (min-heap).

Objective :-
In this experiment we will be able to :
- Understand Dijkstra's Algorithm & its real-world application
- Implement Dijkstra's Algorithm to find the shortest path in a weighted graph.
- Compare Dijktra's Algorithm with other shortest path algorithms like Bellman-ford.

Outcome :-
- Successfully able to implement & finds the shortest path from a source vertex to all other vertices using Dijktra's Algorithm.
- Understand the greedy approach used in path-finding algorithms.
- learn to use priority queues (min-heaps) to optimize

Dijkstra's Algorithm.

Software & Hardware Requirement :

- Open-Source programming tools such as C++/Java.
- Operating System : Window / Linux / Ubuntu.

Theory :

Dijkstra's Shortest Path algorithm :

- Dijkstra's Algorithm is a greedy algorithm used to find the shortest path from a single source vertex to all other vertices in a weighted graph.
- It is application to graphs with non-negative weights.
- It works using a priority Queue (min-heap) to always expand the closet vertex first.

Working Dijkstra's Algorithm :

1) Initialize :
   - Set the distance to the source vertex as 0 & all other vertices as ∞ (infinity).
   - Create a priority queue (min-heap) to store (distance, vertex) pairs.

2) Process Vertices :-
   - Pick the vertex with the smallest distance from the priority queue.
   - Update distances for all its adjacent vertices if a shorter path is found.
   - Push updated distances into the priority Queue.

3) Repeat Until all Vertices are Processed.

Graph Representation with Weights :

Edge	Weight
A-B	4
A-C	1
B-C	2
B-D	5
C-D	8
C-E	10
D-E	2

Step-by-Step Execution of Dijkstra's Algorithm (Starting from A) :

1) Initialize distances : A=0, B=∞, C=∞, D=∞, E=∞
2) Pick the smallest distance node (A=0)
   - Update : B=4, C=1
3) Pick the smallest distance node (C=1).
   - Update : B=3, D=9, E=11
4) Pick the smallest distance node (B=3)
   - Update : D=8

5) Pick the smallest distance node (D = 8)
   - Update : E = 10
6) Pick the smallest distance node (E = 10) (No Updates)

Final shortest distance from A:

- A → A = 0
- A → B = 3
- A → C = 1
- A → D = 8
- A → E = 10

Advantages of Dijkstra's Algorithm:

- Guranteed to find the shortest path in graphs with non-negative weights.
- Efficient for graphs with small edge weights.
- Can be optimized using Fibonacci Heaps O(V log V + E).

Disadvantages of Dijkstra's Algorithm:

- Does not work with negative weights (use Bellman-Ford Algorithm instead).
- More complex than BFS for unweighted graphs.
- Inefficient for dense graphs without a priority queue (O(V²) complexity).

Pune District Education Association's
College Of Engineering
Manjari (Bk.), Hadapsar, Pune-412307.
Accredited by NAAC

Comparison : Dijkstra's vs other shortest path algorithms.

Feature	Dijkstra's Algo.	Bellman Ford Algo.	Floyd-Warshall Alg.
Approach	Greedy	Dynamic programming	Matrix-based.
Works for	Non-negative weights	Negative weights allowed	All pairs shortest path
Time Complexity	O(V log V+E)	O(VE)	O(V³)
Best for	Single-Source shortest path	Single-Source, negative edges	All pairs shortest paths.

Conclusion :-

- Dijkstra's Algorithm is a greedy algorithm used to find the shortest path from a single source to all vertices in a weighted graph.
- It is optimal for non-negative weights & is widely used in routing algorithms like Google Maps, GPS navigation, & network path optimization.

Assignment no:- 7

- Title of Assignment :-
  Implement a Solution for a Constraint Satisfaction Problem using Branch & Bound / Backtracking for n-queens problem.

- Prerequisite :-
  Basic knowledge of OP, Backtracking

- Objective :-
  In this experiment, we will be able to do the following :
  - Study how to place N queens on board with non attacking mode using backtracking.
  - What is OP problem.

- Outcome :-
  Successfully able to place N queens on board with non attacking mode using backtracking.

- Software & Hardware Requirement :
  Open Sourc C++ programming tool like G++ / GCC ; python ; Java & Ubuntu.

- Relevant Theory / literature Survey :-
Constraint means theory Satisfaction problem.
CSP means Solving a problem under certain Constraints or rules
CSP depends on three Components
X : It is a set of variables
D : It is set of domains where the variables reside There is a specific domain for each variable
C : It is set of constraints which are followed by the set of variables.

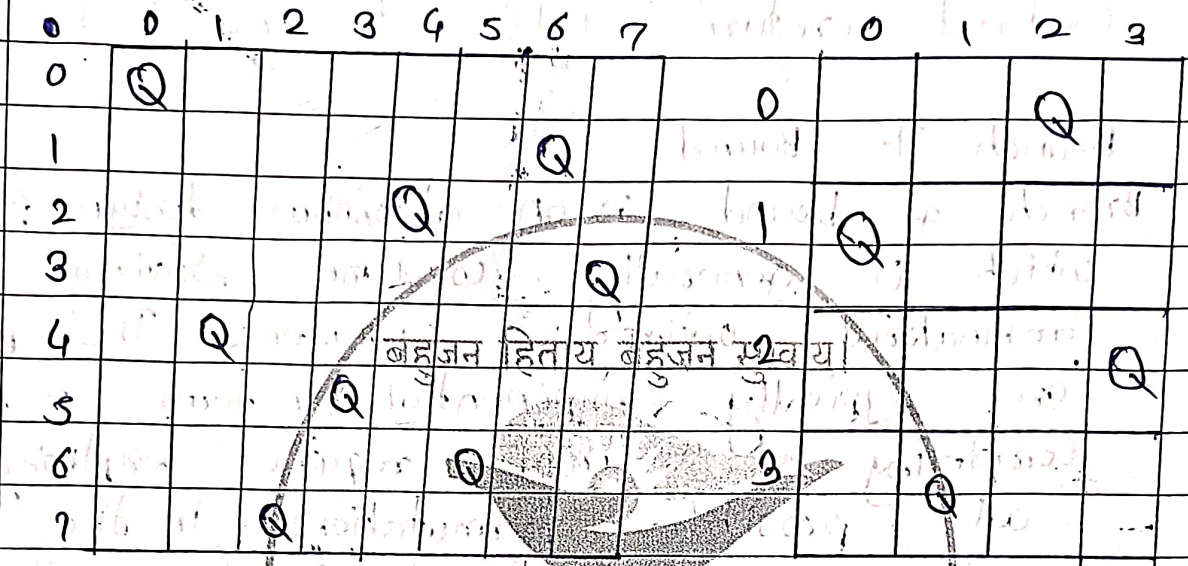
Backtracking Algorithm

Backtracking is an algorithm technique for solving problems recursively by trying to build a solution incrementally one piece of time, removing those solutions that fail to satisfy the Constraints of the problem at any point of time (by time here, is referred to the time elapsed till reaching any level of the search tree)

N-Queens problem

The N-Queens problem is a puzzle of placing exactly N queens on an N*N chessboard, such that no two queens can attack each other in that configuration. Thus, no two queens can lie in the same

row, Column or diagonal



Solution : {0,4,7,5,2,6,1,3} Solution : {1,3,0,2}

N- Queens problem Algorithm

1) Start in the leftmost column
2) If all queens are placed return true
3) Try all rows in the current Column. Do the following for every tried row.
   a) If the queen can be placed safely in this row then mark this [row, column] as part of the solution & recursively check if placing the queen here leads to a solution
   b) If placing the queen in [row, column] leads to a solution then return true.

c) If placing queen doesn't lead to a solution (Backtrack) & go to step (a) to try other row.

4) If all rows have been tried & nothing worked return false to trigger backtracking.

Branch & Bound

Branch & bound is an algorithm design paradigm which is generally used for solving Combinatorial optimization problems. These problems are typically exponential in terms of time Complexity & may require exploring all possible permutations in the worst case. The Branch & Bound Algorithm technique solves these problems relatively quickly.

Applying the branch & bound approach:

The branch & bound approach suggests that we create a partial solution & use it to ascertain whether we need to continue in a particular direction or not.

Pune District Education Association's

College Of Engineering

Manjari (Bk.), Hadapsar, Pune-412307.

Accredited by NAAC

Parameter	Backtracking	Branch & Bound
Approach	Backtracking is used to find all possible solutions available to a problem. when it realize that it has made a bad choice, it undoes the last choice by backing it up. It searches the state space tree until it has found a solution for the problem.	Branch & Bound is used to solve optimization problems. when it realize that it already has a better optimal solution that the pre-solution leads to, it abandons that pre-solution. It completely searches the state space tree to get an optimal solution.
Traversal	Backtracking traverses the state space tree by DFS (Depth First Search) manner	Branch-And-Bound traverse the tree in any manner, DFS or BFS
~~Function~~ Problems	Backtracking is used for solving Decision Problems	Branch-&-bound is used for solving the optimization problems
Function	Backtracking involves a feasibility function.	Branch-and-Bound involves a bounding function.

Efficiency	Backtracking is more efficient	Branch and Bound is less efficient
Application	Useful in Solving n-Queen problem, Sum of Subset	Useful in Solving Knapsack Problem, Traveling Salesman Problem.

- Conclusion :-
  In this way we have studied how to Solve the CSP problem & how to place N queens on board with non attacking mode using backtracking.