EXP 4 : APPLICATION OF BFS & DFS

BFS

Problem Formulation :

DFS: Lexicographic sorting of a given set of keys.

→ Given a set of strings, return them in lexicographic order (alphabetical order).

Problem Solving

Lexicographic sorting of a set of keys can be accomplished with a simple Trie-based algorithm by :

- Insert all keys into a Trie
- Print all g keys in the Trie by performing pre-order traversal on Trie to get output in alphabeticaly increasing order.

Algorithm

To print the strings in alphabetical order we have to first insert them in the trie and then perform pre order traversal to print in alphabetical order.

The nodes of trie contain an index[] array which stores the index position of all the strings of arr[] ending at that node Except for the trie's leaf node all the other nodes have the size 0 for the index[] array.

BFS : Chess Knight Problem

Problem formulation :

Given a chessboard, find the shortest distance, (min no. of steps) taken by a knight to reach a given destination from a given source.

Problem solving :

A knight can move in 8 possible directions from a given cell, in below example:

For eg: Input : N = 8 (8 x 8 board)
Source = (7, 0)
Destination = (0, 7)

We can find all the possible locations the knight moves to from the given to location by using the array that stores the relative positions of knight movement from any location.

In BFS, all cells have the shortest path as 1 are visited first, followed by their adjacent cells having the shortest path as 1 + 1 = 2 .... so on. so if we reach any node in BFS, shortest path = shortest path of parent + 1 . So, the destination cell's first occurance gives us the result, & we can stop our search here.

Algorithm

We use Breadth-First Search as it is the shortest path problem.

- Create an empty queue & enqueue the source cell having a distance of 0 from the source.
- Loop till queue is empty:
    1. Dequeue next unvisited node
    2. If the popped node is the destination node, return its distance
    3. Else, we mark the current node as visited. For each of eight possible movements for a knight, enqueue each valid movement with +1 distance (min distance of a given node from the source is one more than the min distance of parent source)