Dijkstra's algorithm

It is a widely-used algorithm for finding the shortest path between nodes in a graph, particularly in scenarios where all edge weights are non-negative. It's a popular algorithm used in various applications such as routing, network design, and more.

Here's a brief overview of Dijkstra's algorithm:

- Initialization: Set initial node distance to 0, others to infinity, and maintain unvisited nodes.
- Iteration: Repeat until all nodes visited:
- Select unvisited node with smallest distance.
- Update distances of its neighbors if shorter via current node.
- Selection of Next Node: Choose node with smallest tentative distance.
- Termination: Repeat until no unvisited nodes remain.

1. DFS Algorithm:

Step 1: Push the root node in the Stack.

Step 2: Loop until stack is empty.

Step 3: Peek the node of the stack.

Step 4: If the node has unvisited child nodes, get the unvisited child node, mark it as traversed and push it on stack.

Step 5: If the node does not have any unvisited child nodes, pop the node from the stack.

2. BFS Algorithm:

Step 1: Push the root node in the Queue.

Step 2: Loop until the queue is empty.

Step 3: Remove the node from the Queue.

Step 4: If the removed node has unvisited child nodes, mark them as visited and insert the unvisited children in the queue.

Algorithm for **Merge Sort chicken ka**

Step 1: Find the middle index of the array.

Middle = 1 + (last – first)/2

Step 2: Divide the array from the middle.

Step 3: Call merge sort for the first half of the array

MergeSort(array, first, middle)

Step 4: Call merge sort for the second half of the array.

MergeSort(array, middle+1, last)

Step 5: Merge the two sorted halves into a single sorted array.

Fibonacci

1. Generate a list of 50 random numbers within the range of 0 to 50.
2. Check if a number is a Fibonacci number using mathematical properties.
3. Filter Fibonacci numbers from the list of random numbers, ensuring uniqueness.
4. Sort the list of Fibonacci numbers in ascending order.
5. Traverse through the list of Fibonacci numbers to play songs.
    - Display the current playing song number.
    - Prompt the user to input '>' to play the next song, '<' to play the previous song, or 'x' to quit.
    - If the input is '>', move to the next song.
    - If the input is '<', move to the previous song.
    - If the input is 'x', stop playing music and exit.
    - Handle invalid inputs by displaying an error message and prompting again.