**Assignment – Nirbhay Gandhi**

**Searching Methods**

**GitHub link:** <https://github.com/Nirbhay-Gandhi/Python-Programming/tree/Prod/Assignment>

**Initial Considerations**

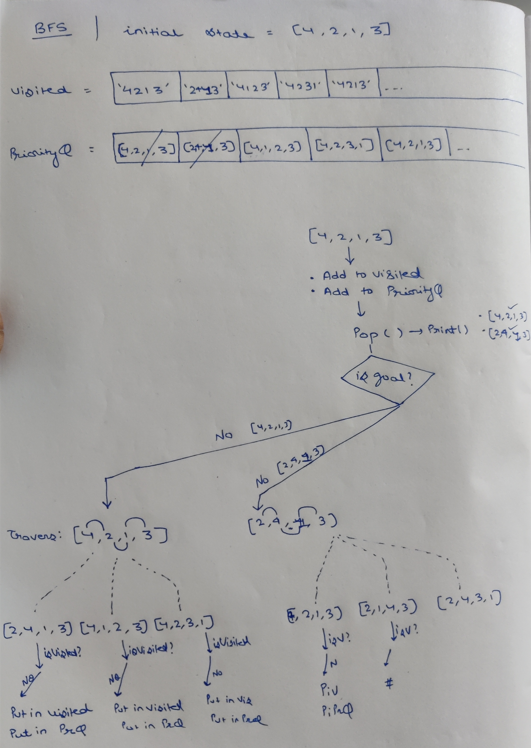
* **start\_state :** unsorted list of double type (user input)
* **successive\_state :** list with swapped neighboring element
* **goal\_state :** sorted list in ascending order
* **cost to travel a successive state :** 1 unit per action

**Data Structures in use**

- Two data structures in use :

* Queue - Replicated by List
* Visited Array – To store all the visited states, replicated by using Set

**Algorithm Overview using flowchart**



**1. Breadth First Search (BFS)**

**Algorithm**

1. **Queue & Visited array** both to be initialized **empty.**
2. Adding of the element in Queue: In BFS to add the element in the Queue, we will simple **Push initial\_state** into the **queue** any mark it visited
3. Removal of the element from Queue: In BFS for popping (removal) of the element, will be simply based on the **FIFO(First in First out) principle.**
4. **Check**, is the given\_state equal to the **goal state**, if so algorithm comes to end. Else proceed next steps
5. **Generate the neighboring states** : Each neighboring state is generated by **swapping the ith and (i+1)th element** of the initial\_state.
6. Visit the **unvisited neighbors.**
7. Repeat from step **2**.

**\*\*Note:** A similar approach of algorithm will be followed in rest all the Searching algorithms.

**Only difference will be in Step-2 (Addition of Element) and Step-3 (Removal of Element)**. Different methods are followed that will illustrate in that particular algorithm section.

**2. Depth First Search (DFS)**

**Algorithm**

1. **Queue & Visited array** both to be initialized **empty.**
2. Adding of the element in Stack: In DFS to add the element in the Stack, we will simple **Push initial\_state** into the **stack** any mark it visited.
3. Removal of the element from Stack: In DFS for popping (removal) of the element, will be based on the **LIFO(Last in First out) principle.**
4. **Check**, is the given\_state equal to the **goal state**, if so algorithm comes to end. Else proceed next steps
5. **Generate the neighboring states** : Each neighboring state is generated by **swapping the ith and (i+1)th element** of the initial\_state.
6. Visit the **unvisited neighbors.**
7. Repeat from step **2**.

**2. Uniform Cost Search (DFS)**

**Algorithm**

1. **Queue & Visited array** both to be initialized **empty.**
2. Adding of the element in Stack: In DFS to add the element in the Stack, we will simple **Push initial\_state** into the **stack** any mark it visited.
3. Removal of the element from Stack: In DFS for popping (removal) of the element, will be based on the **LIFO(Last in First out) principle.**
4. **Check**, is the given\_state equal to the **goal state**, if so algorithm comes to end. Else proceed next steps
5. **Generate the neighboring states** : Each neighboring state is generated by **swapping the ith and (i+1)th element** of the initial\_state.
6. Visit the **unvisited neighbors.**
7. Repeat from step **2**.