Table of Dry Runs and Algo8rithms for Practice

1. Bubble Sort

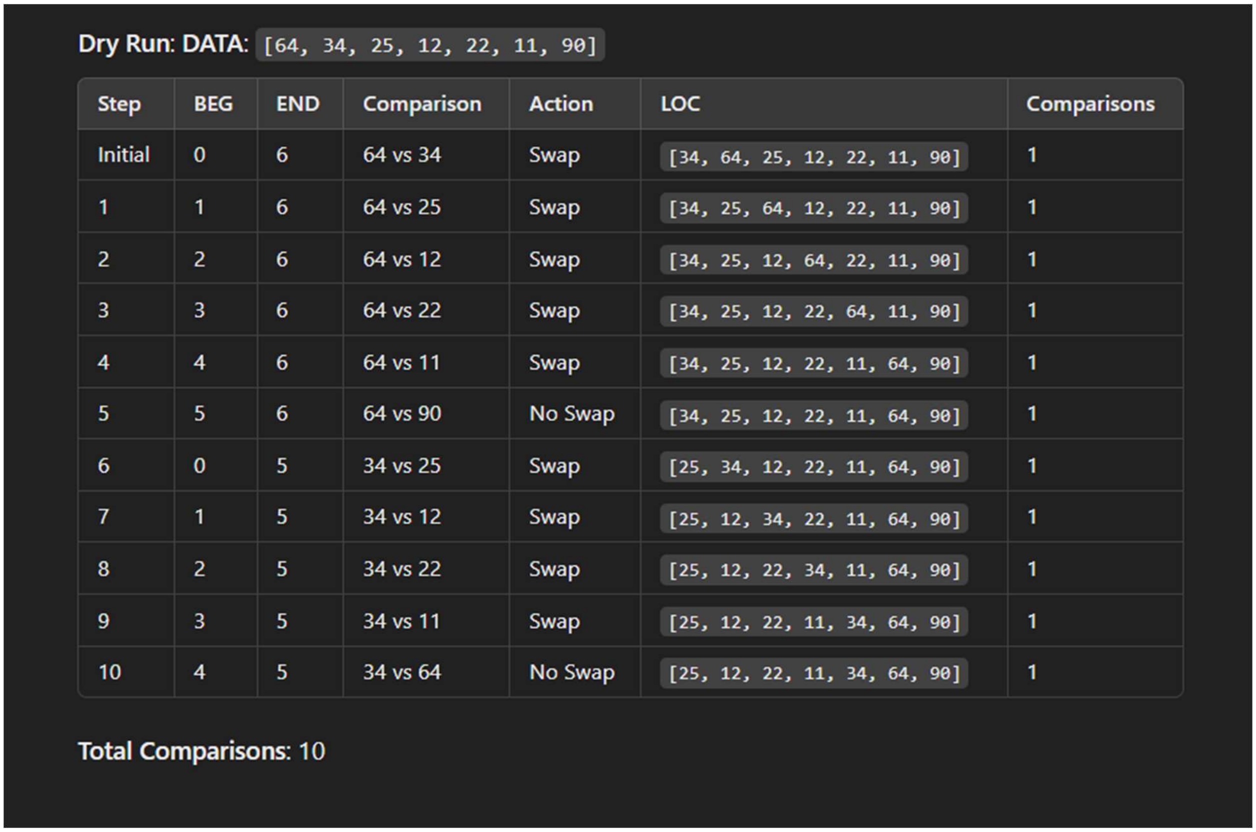
Algorithm:

Compare adjacent elements and swap if needed.

Repeat for all elements, reducing the number of elements to compare after each pass.

Stop when no swaps are needed.

Dry Run: DATA: [64, 34, 25, 12, 22, 11, 90]



Dry Run: DATA: `[64, 34, 25, 12, 22, 11, 90]`

Step	BEG	END	Comparison	Action	LOC	Comparisons
Initial	0	6	64 vs 34	Swap	`[34, 64, 25, 12, 22, 11, 90]`	1
1	1	6	64 vs 25	Swap	`[34, 25, 64, 12, 22, 11, 90]`	1
2	2	6	64 vs 12	Swap	`[34, 25, 12, 64, 22, 11, 90]`	1
3	3	6	64 vs 22	Swap	`[34, 25, 12, 22, 64, 11, 90]`	1
4	4	6	64 vs 11	Swap	`[34, 25, 12, 22, 11, 64, 90]`	1
5	5	6	64 vs 90	No Swap	`[34, 25, 12, 22, 11, 64, 90]`	1
6	0	5	34 vs 25	Swap	`[25, 34, 12, 22, 11, 64, 90]`	1
7	1	5	34 vs 12	Swap	`[25, 12, 34, 22, 11, 64, 90]`	1
8	2	5	34 vs 22	Swap	`[25, 12, 22, 34, 11, 64, 90]`	1
9	3	5	34 vs 11	Swap	`[25, 12, 22, 11, 34, 64, 90]`	1
10	4	5	34 vs 64	No Swap	`[25, 12, 22, 11, 34, 64, 90]`	1

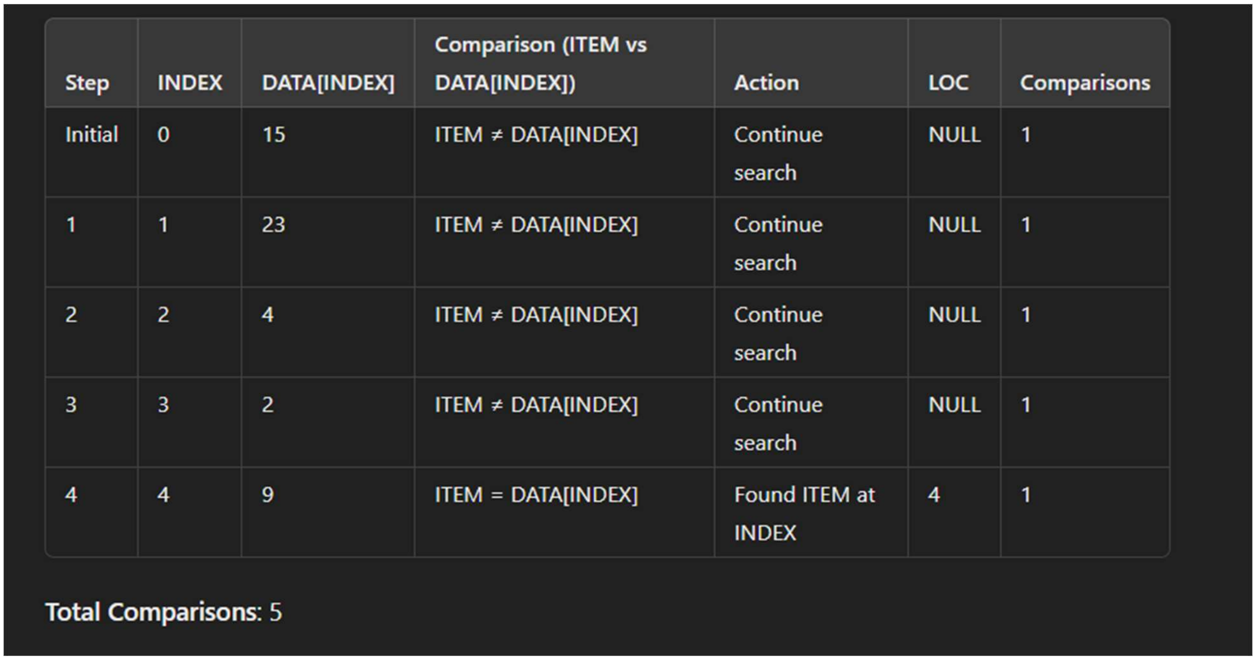
Total Comparisons: 10

Table of Dry Runs and Algo8rithms for Practice

2. Linear Search

Algorithm:

Start from the first element.
Compare the element with the target.
If found, return the index, else continue.
Return -1 if not found.
Dry Run: DATA: [15, 23, 4, 2, 9, 5]
ITEM: 9



Step	INDEX	DATA[INDEX]	Comparison (ITEM vs DATA[INDEX])	Action	LOC	Comparisons
Initial	0	15	ITEM ≠ DATA[INDEX]	Continue search	NULL	1
1	1	23	ITEM ≠ DATA[INDEX]	Continue search	NULL	1
2	2	4	ITEM ≠ DATA[INDEX]	Continue search	NULL	1
3	3	2	ITEM ≠ DATA[INDEX]	Continue search	NULL	1
4	4	9	ITEM = DATA[INDEX]	Found ITEM at INDEX	4	1

Total Comparisons: 5

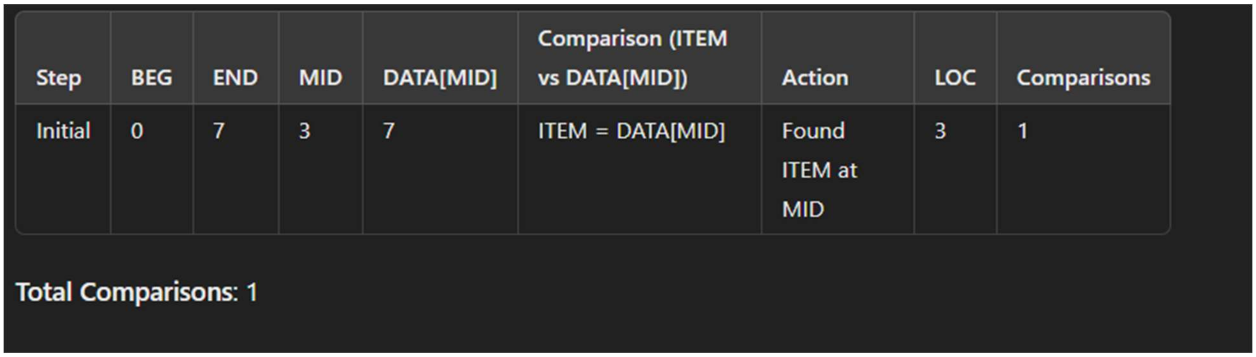
Table of Dry Runs and Algo8rithms for Practice

3. Binary Search

Algorithm:

1. **Start with the middle element.**
2. **Compare the target with the middle element.**
   - **If equal, return the index.**
   - **If target is less, search left half.**
   - **If target is greater, search right half.**
3. **Repeat until found.**

Dry Run: DATA: [1, 3, 5, 7, 9, 11, 13, 15]
ITEM: 7



Step	BEG	END	MID	DATA[MID]	Comparison (ITEM vs DATA[MID])	Action	LOC	Comparisons
Initial	0	7	3	7	ITEM = DATA[MID]	Found ITEM at MID	3	1

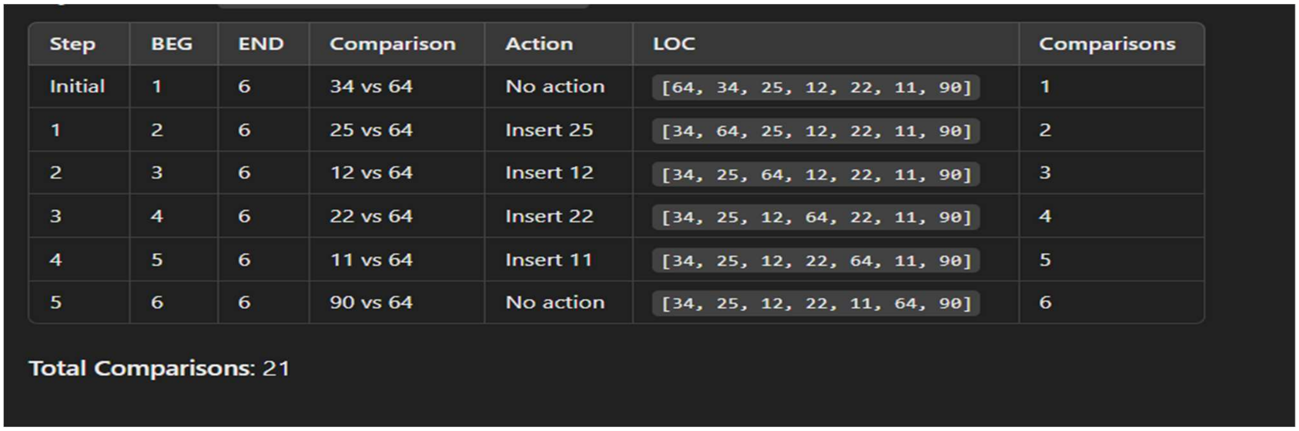
Total Comparisons: 1

4. Insertion Sort

Algorithm:

1. **Start from the second element.**
2. **Compare it with previous elements and insert it in the correct position.**
3. **Repeat for each element.**

Dry Run: DATA: `[64, 34, 25, 12, 22, 11, 90]`



Step	BEG	END	Comparison	Action	LOC	Comparisons
Initial	1	6	34 vs 64	No action	`[64, 34, 25, 12, 22, 11, 90]`	1
1	2	6	25 vs 64	Insert 25	`[34, 64, 25, 12, 22, 11, 90]`	2
2	3	6	12 vs 64	Insert 12	`[34, 25, 64, 12, 22, 11, 90]`	3
3	4	6	22 vs 64	Insert 22	`[34, 25, 12, 64, 22, 11, 90]`	4
4	5	6	11 vs 64	Insert 11	`[34, 25, 12, 22, 64, 11, 90]`	5
5	6	6	90 vs 64	No action	`[34, 25, 12, 22, 11, 64, 90]`	6

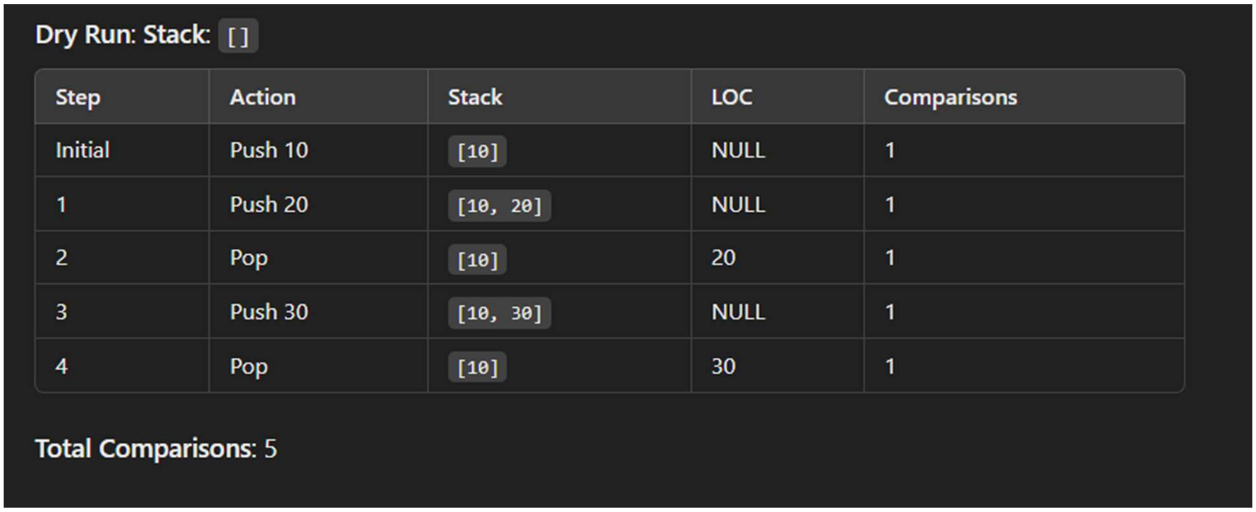
Total Comparisons: 21

Table of Dry Runs and Algo8rithms for Practice

5. Stack Operations

Algorithm:

1. **Push: Add an item to the stack.**
2. **Pop: Remove an item from the stack.**



Dry Run: Stack: `[]`

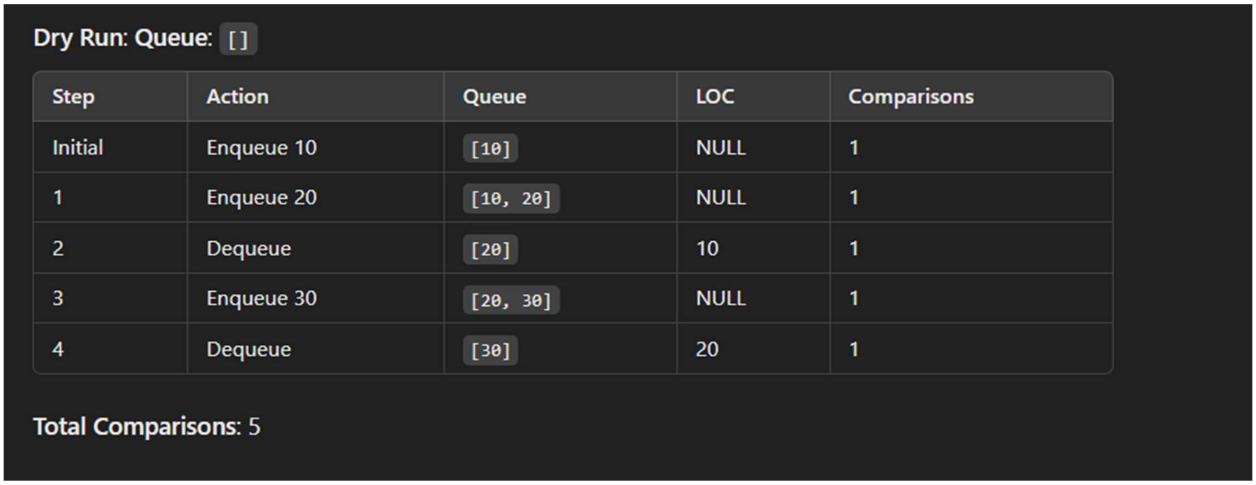
Step	Action	Stack	LOC	Comparisons
Initial	Push 10	`[10]`	NULL	1
1	Push 20	`[10, 20]`	NULL	1
2	Pop	`[10]`	20	1
3	Push 30	`[10, 30]`	NULL	1
4	Pop	`[10]`	30	1

Total Comparisons: 5

6. Queue Operations

Algorithm:

1. **Enqueue**: Add an item to the queue.
2. **Dequeue**: Remove an item from the queue.



Dry Run: Queue: `[]`

Step	Action	Queue	LOC	Comparisons
Initial	Enqueue 10	`[10]`	NULL	1
1	Enqueue 20	`[10, 20]`	NULL	1
2	Dequeue	`[20]`	10	1
3	Enqueue 30	`[20, 30]`	NULL	1
4	Dequeue	`[30]`	20	1

Total Comparisons: 5

Table of Dry Runs and Algo8rithms for Practice

7. Brute Force Approach for Finding Largest Element in an Array

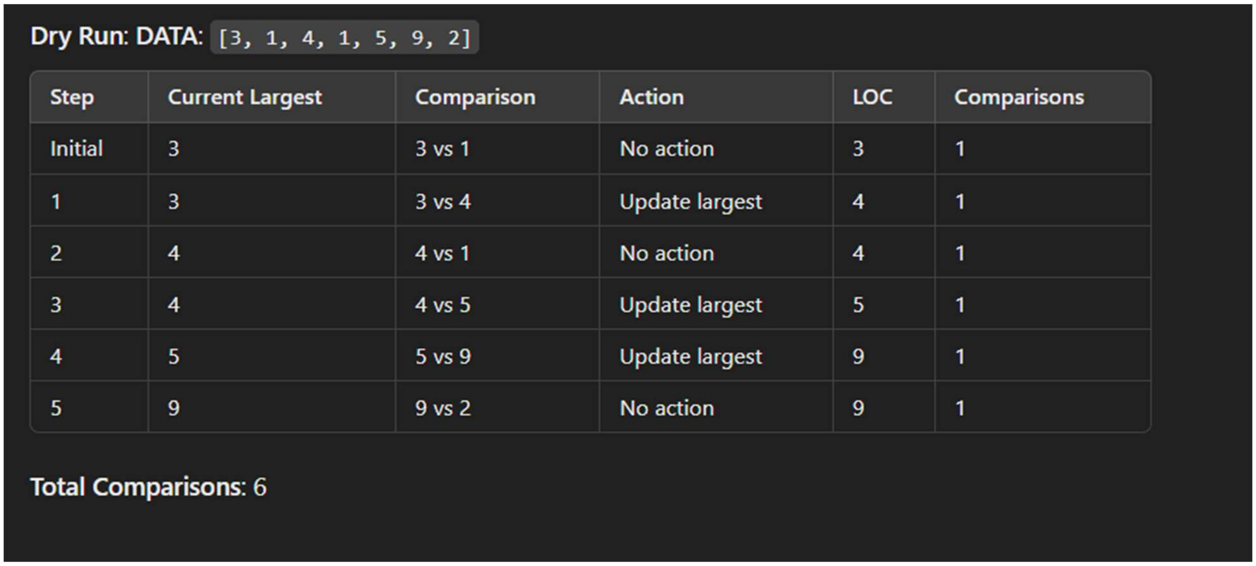
Algorithm:

Start by assuming the first element is the largest.

Compare each subsequent element with the current largest.

If a larger element is found, update the current largest.

Return the largest element.



Dry Run: DATA: `[3, 1, 4, 1, 5, 9, 2]`

Step	Current Largest	Comparison	Action	LOC	Comparisons
Initial	3	3 vs 1	No action	3	1
1	3	3 vs 4	Update largest	4	1
2	4	4 vs 1	No action	4	1
3	4	4 vs 5	Update largest	5	1
4	5	5 vs 9	Update largest	9	1
5	9	9 vs 2	No action	9	1

Total Comparisons: 6

Table of Dry Runs and Algo8rithms for Practice

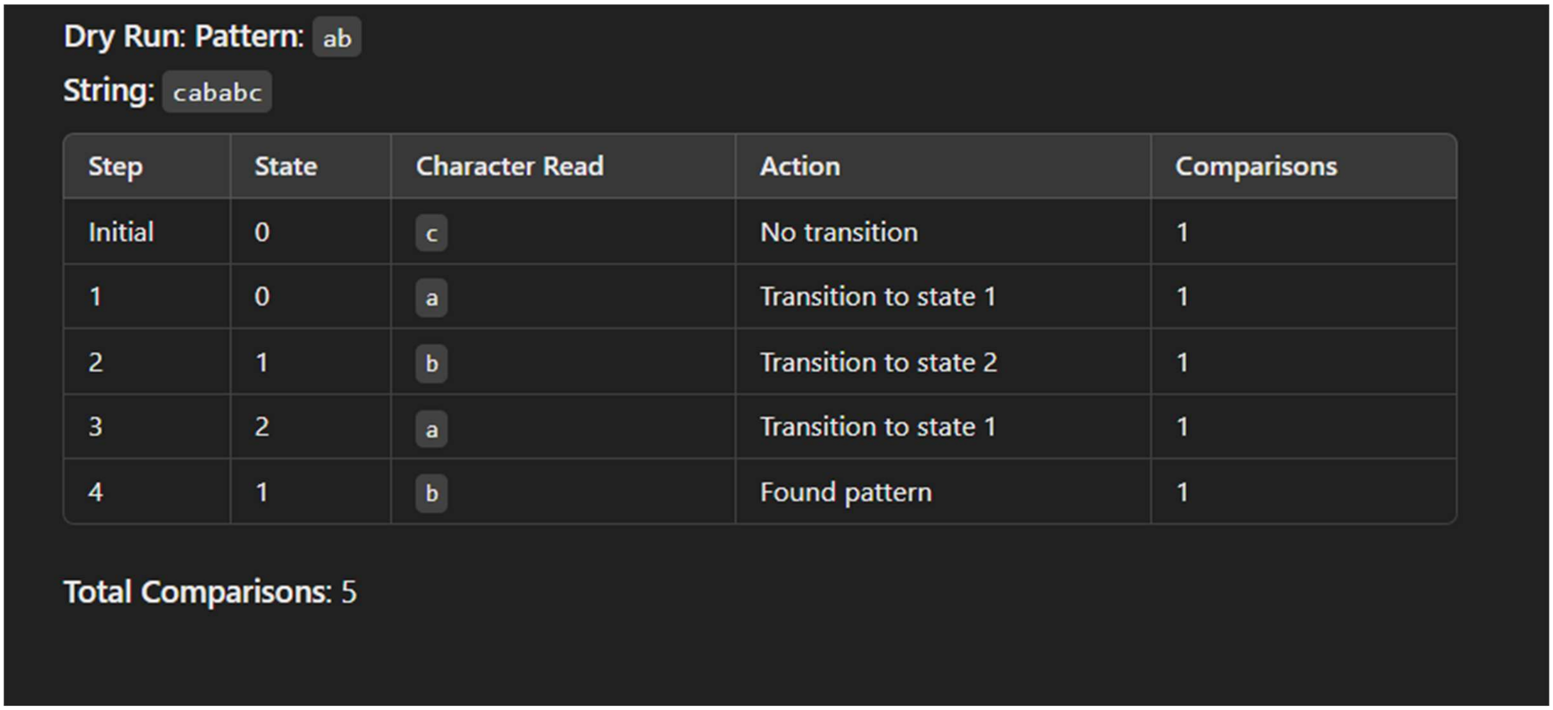
8. Finite Automata for Simple String Matching

Algorithm:

Begin at the initial state.

For each character of the string, move through the states as per the transition rules.

If the string matches a pattern, accept; otherwise, reject.



Dry Run: Pattern: `ab`

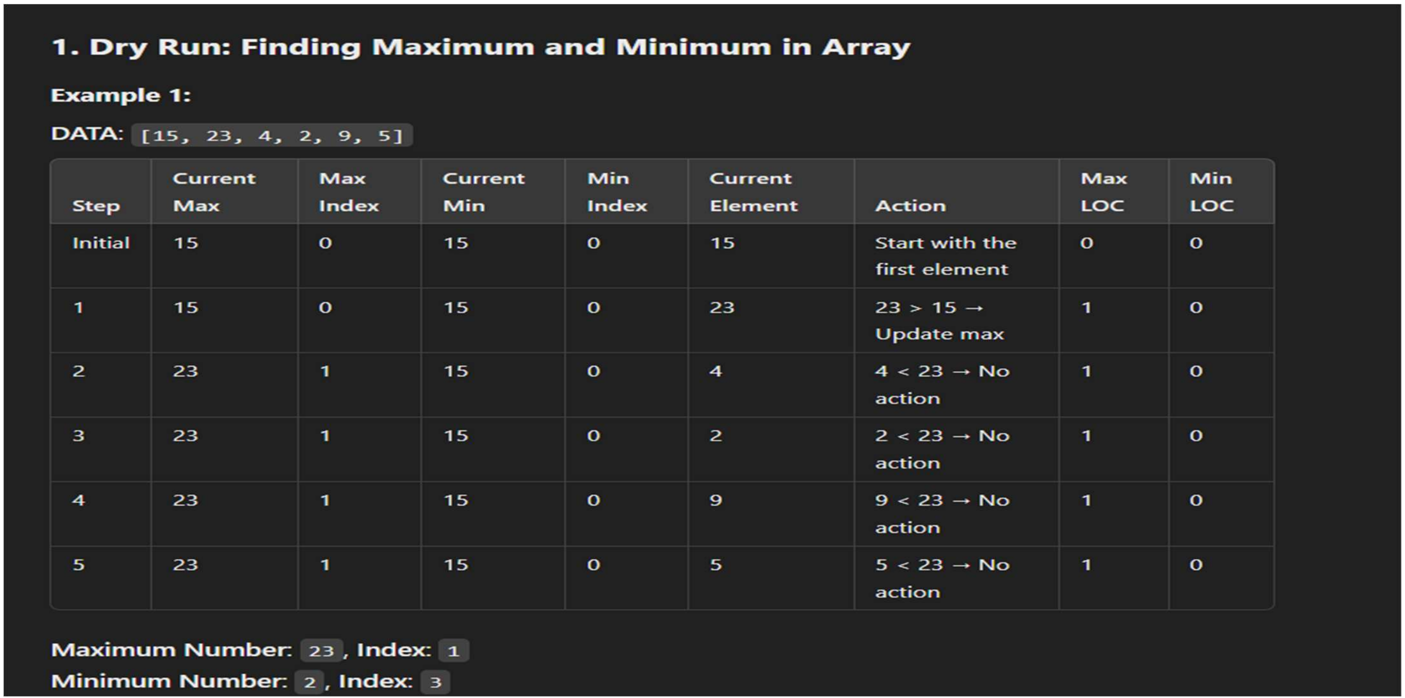
String: `cababc`

Step	State	Character Read	Action	Comparisons
Initial	0	`c`	No transition	1
1	0	`a`	Transition to state 1	1
2	1	`b`	Transition to state 2	1
3	2	`a`	Transition to state 1	1
4	1	`b`	Found pattern	1

Total Comparisons: 5

Table of Dry Runs and Algo8rithms for Practice

9. MAXIMUM AND MINIMUM



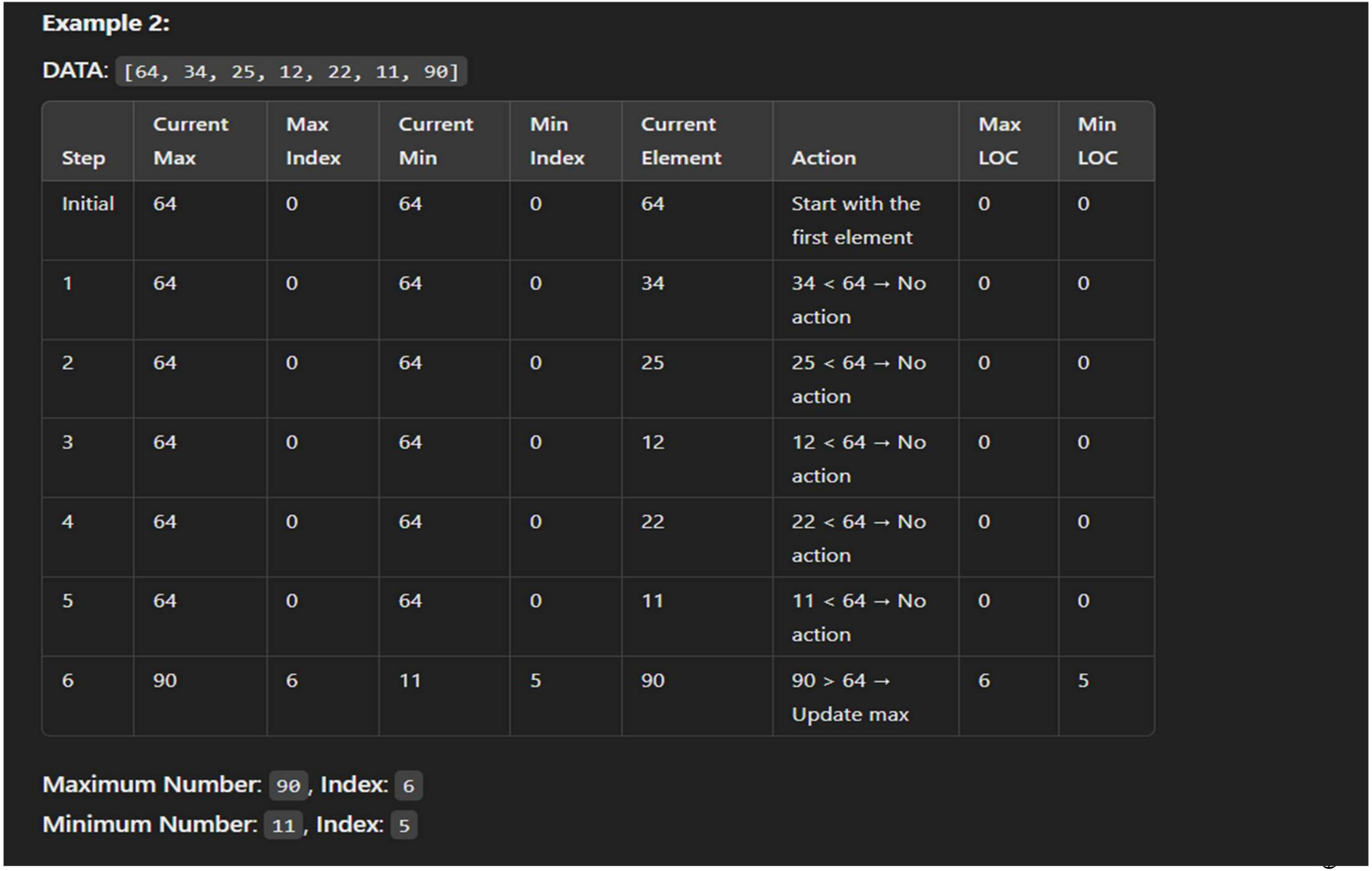
1. Dry Run: Finding Maximum and Minimum in Array

Example 1:

DATA: `[15, 23, 4, 2, 9, 5]`

Step	Current Max	Max Index	Current Min	Min Index	Current Element	Action	Max LOC	Min LOC
Initial	15	0	15	0	15	Start with the first element	0	0
1	15	0	15	0	23	23 > 15 → Update max	1	0
2	23	1	15	0	4	4 < 23 → No action	1	0
3	23	1	15	0	2	2 < 23 → No action	1	0
4	23	1	15	0	9	9 < 23 → No action	1	0
5	23	1	15	0	5	5 < 23 → No action	1	0

Maximum Number: `23`**, Index:** `1`
Minimum Number: `2`**, Index:** `3`



Example 2:

DATA: `[64, 34, 25, 12, 22, 11, 90]`

Step	Current Max	Max Index	Current Min	Min Index	Current Element	Action	Max LOC	Min LOC
Initial	64	0	64	0	64	Start with the first element	0	0
1	64	0	64	0	34	34 < 64 → No action	0	0
2	64	0	64	0	25	25 < 64 → No action	0	0
3	64	0	64	0	12	12 < 64 → No action	0	0
4	64	0	64	0	22	22 < 64 → No action	0	0
5	64	0	64	0	11	11 < 64 → No action	0	0
6	90	6	11	5	90	90 > 64 → Update max	6	5

Maximum Number: `90`**, Index:** `6`
Minimum Number: `11`**, Index:** `5`

Table of Dry Runs and Algo8rithms for Practice

Practice Sheet Questions:

1. **How many comparisons are made in the Bubble Sort algorithm for the array [64, 34, 25, 12, 22, 11, 90]?**
2. **How many comparisons are made in the Linear Search algorithm for finding the item 9 in [15, 23, 4, 2, 9, 5]?**
3. **How many comparisons are made in Binary Search for finding 7 in [1, 3, 5, 7, 9, 11, 13, 15]?**
4. **How many comparisons are made in Insertion Sort for the array [64, 34, 25, 12, 22, 11, 90]?**
5. **How many comparisons are made in the Stack operations (Push and Pop)?**
6. **How many comparisons are made in the Queue operations (Enqueue and Dequeue)?**
7. **How many comparisons are made in Brute Force for finding the largest element in [3, 1, 4, 1, 5, 9, 2]?**
8. **How many comparisons are made in Finite Automata string matching for the pattern ab in cababc?**
9. **Given the array [3, 8, 15, 6, 2, 9, 1], find the maximum and minimum numbers and their indices.**
10. **Given the array [10, 20, 30, 40, 50, 60], find the maximum and minimum numbers and their indices.**
11. **For the array [15, 34, 23, 5, 11, 17], find the largest and smallest elements and their respective indices.**
12. **In the array [1, 2, 3, 4, 5, 6, 7, 8, 9], identify the maximum and minimum values and their indices.**

Filename:	Document1
Directory:	
Template:	C:\Users\Abdullah\AppData\Roaming\Microsoft\Templates\Normal.dotm
Title:	
Subject:	
Author:	Abdullah
Keywords:	
Comments:	
Creation Date:	11/17/2024 2:30:00 PM
Change Number:	1
Last Saved On:	
Last Saved By:	
Total Editing Time:	18 Minutes
Last Printed On:	11/17/2024 2:48:00 PM
As of Last Complete Printing	
Number of Pages:	8
Number of Words:	429 (approx.)
Number of Characters:	2,450 (approx.)