CHAPTER 9: SORTING ALGORITHMS

9.1 BUBBLE SORT

9.1.1 How It Works

1. Compare adjacent elements
2. Swap if in wrong order
3. Repeat for entire list
4. Each pass places largest unsorted element at end
5. Continue until no swaps needed

9.1.2 Pseudocode

<PSEUDOCODE>

```
DECLARE myList : ARRAY[0:8] OF INTEGER
DECLARE UB : INTEGER
DECLARE LB : INTEGER
DECLARE index : INTEGER
DECLARE swap : BOOLEAN
DECLARE temp : INTEGER
DECLARE top : INTEGER

UB ← 8
LB ← 0
top ← UB

REPEAT
 swap ← FALSE
 FOR index ← LB TO top - 1
 IF myList[index] > myList[index + 1] THEN
 temp ← myList[index]
```

9.1.3 Python Code

<PYTHON>

```
def bubbleSort(myList):
 n = len(myList)
 for i in range(n):
 swap = False
 for j in range(0, n-i-1):
 if myList[j] > myList[j+1]:
 myList[j], myList[j+1] = myList[j+1], myList[j]
 swap = True
 if not swap:
 break
```

9.1.4 Complexity

bubbleSort(myList)

print(myList)

- **Best Case:** O(n) - already sorted
- **Worst Case:** O(n²) - reverse sorted

- **Space:** O(1)

9.2 INSERTION SORT

9.2.1 How It Works

1. Assume first element is sorted
2. Take next element
3. Insert into correct position in sorted portion
4. Repeat until all elements sorted

9.2.2 Pseudocode

<PSEUDOCODE>

```
DECLARE myList : ARRAY[0:9] OF INTEGER
DECLARE UB : INTEGER
DECLARE LB : INTEGER
DECLARE index : INTEGER
DECLARE key : INTEGER
DECLARE place : INTEGER

UB ← 9
LB ← 0

FOR index ← LB + 1 TO UB
 key ← myList[index]
 place ← index - 1
 WHILE place >= LB AND myList[place] > key
 place ← place - 1
```

9.2.3 Python Code

<PYTHON>

```
def insertionSort(arr):
 for i in range(1, len(arr)):
 key = arr[i]
 j = i - 1
 while j >= 0 and key < arr[j]:
 arr[j + 1] = arr[j]
 j -= 1
```

9.2.4 Complexity

- **Best Case:** O(n) - already sorted
- **Worst Case:** O(n^2) - reverse sorted
- **Space:** O(1)

Revision #1
Created 2026-03-16 12:18:06 UTC by Samuel Lee
Updated 2026-03-16 12:18:17 UTC by Samuel Lee