ASSIGNMENT

Submitted By

Ashbi Saju

Roll No. 16

4. Sorting Algorithms with Iteration Analysis

4.1 Bubble Sort

Program (C)

```
#include <stdio.h>
void bubbleSort(int arr[], int n, int *outer, int *inner) {
    int i, j, temp;
    for (i = 0; i < n - 1; i++) {
        (*outer)++;
        for (j = 0; j < n - i - 1; j++) {
            (*inner)++;
            if (arr[j] > arr[j + 1]) {
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

void testBubbleSort(char *caseName, int arr[], int n) {
    int outer = 0, inner = 0;
    bubbleSort(arr, n, &outer, &inner);
    printf("%s Case → Outer: %d, Inner: %d\n", caseName, outer, inner);
}

int main() {
    int n = 5;
    int best[]  = {1, 2, 3, 4, 5};   // Best case
    int avg[]   = {3, 1, 4, 5, 2};   // Average case
    int worst[] = {5, 4, 3, 2, 1};   // Worst case

    printf("BUBBLE SORT\n");
    testBubbleSort("Best", best, n);
    testBubbleSort("Average", avg, n);
    testBubbleSort("Worst", worst, n);

    return 0;
}
```

Output

```
BUBBLE SORT
Best Case → Outer: 4, Inner: 10
Average Case → Outer: 4, Inner: 10
Worst Case → Outer: 4, Inner: 10
```

Iteration Count

Case	Outer Loop	Inner Loop	Total
Best	4	10	14
Average	4	10	14
Worst	4	10	14

Time Complexity

- Best: $O(n^2)$
- Average: $O(n^2)$
- Worst: $O(n^2)$

4.2 Selection Sort

Program (C)

```
#include <stdio.h>
void selectionSort(int arr[], int n, int *outer, int *inner) {
  int i, j, min, temp;
  for (i = 0; i < n - 1; i++) {
    (*outer)++;
    min = i;
    for (j = i + 1; j < n; j++) {
      (*inner)++;
      if (arr[j] < arr[min])
        min = j;
    }
    temp = arr[i];
    arr[i] = arr[min];
    arr[min] = temp;
  }
}

void testSelectionSort(char *caseName, int arr[], int n) {
  int outer = 0, inner = 0;
  selectionSort(arr, n, &outer, &inner);
  printf("%s Case → Outer: %d, Inner: %d\n", caseName, outer, inner);
}

int main() {
  int n = 5;
  int best[]  = {1, 2, 3, 4, 5};
  int avg[]   = {3, 1, 4, 5, 2};
  int worst[] = {5, 4, 3, 2, 1};
```

```
    printf("SELECTION SORT\n");
    testSelectionSort("Best", best, n);
    testSelectionSort("Average", avg, n);
    testSelectionSort("Worst", worst, n);

    return 0;
}
```

Output

```
SELECTION SORT
Best Case → Outer: 4, Inner: 10
Average Case → Outer: 4, Inner: 10
Worst Case → Outer: 4, Inner: 10
```

Iteration Count

Case	Outer Loop	Inner Loop	Total
Best	4	10	14
Average	4	10	14
Worst	4	10	14

Time Complexity

- Best: $O(n^2)$
- Average: $O(n^2)$
- Worst: $O(n^2)$

4.3 Insertion Sort

Program (C)

```
#include <stdio.h>
void insertionSort(int arr[], int n, int *outer, int *inner) {
    int i, j, key;
    for (i = 1; i < n; i++) {
        (*outer)++;
        key = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > key) {
            (*inner)++;
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
}

void testInsertionSort(char *caseName, int arr[], int n) {
    int outer = 0, inner = 0;
    insertionSort(arr, n, &outer, &inner);
    printf("%s Case → Outer: %d, Inner: %d\n", caseName, outer, inner);
}

int main() {
    int n = 5;
    int best[]  = {1, 2, 3, 4, 5};
    int avg[]   = {3, 1, 4, 5, 2};
    int worst[] = {5, 4, 3, 2, 1};
    printf("INSERTION SORT\n");
    testInsertionSort("Best", best, n);
    testInsertionSort("Average", avg, n);
    testInsertionSort("Worst", worst, n);

    return 0;
}
```

Output

```
INSERTION SORT
Best Case → Outer: 4, Inner: 0
Average Case → Outer: 4, Inner: 4
Worst Case → Outer: 4, Inner: 10
```

Iteration Count

Case	Outer Loop	Inner Loop	Total
Best	4	0	4
Average	4	6	10
Worst	4	10	14

Time Complexity

- Best: $O(n)$
- Average: $O(n^2)$
- Worst: $O(n^2)$

Comparative Summary Table

Algorithm	Best Case	Average Case	Worst Case
Bubble Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Insertion Sort	$O(n)$	$O(n^2)$	$O(n^2)$