

**Symbiosis Institute of Technology**

**Faculty of Engineering**

**CSE- Academic Year 2024-25**

**Data Structures – Lab Batch 2023-27**

|  |  |
| --- | --- |
| **Lab Assignment No:- 2** | |
|  | |
| **Name of Student** | Pankhuri Varshney |
| **PRN No.** | 23070122160 |
| **Batch** | 2023-27 |
| **Class** | B2 |
| **Academic Year & Semester** | 2024-25  Semester 3 |
| **Date of Performance** | 28th July 24 |
|  | |
| **Title of Assignment:** | A. Implement following sorting techniques and find the time complexity:  i. Bubble  ii. Selection  iii. Insertion |
| **Theory Questions:** | 1. Prepare table for following 10 different sorting algorithms for their best case, average case and worst-case time complexities. 2. Solve examples of bubble sort, insertion sort and selection sort. Show all passes. 3. Write real world applications of bubble sort, insertion sort and selection sort. 4. How we can optimize bubble sort. |
| **Source Code/Algorithm/Flow Chart:** | **1. Implement following sorting techniques and find the time**  **complexity:**  **i. Bubble Sort (Unoptimized):**  Source Code:  #include <stdio.h>  void bubbleSort(int arr[], int n);  int main(){  int n;  printf("Enter the size of the array: ");  scanf("%d", &n);  int arr[n];  printf("Enter the elements: \n");  for(int i=0; i<n; i++){  scanf("%d", &arr[i]);  }  bubbleSort(arr, n);  printf("SORTED ARRAY 1: \n");  for(int i=0; i<n; i++)  printf("%d\t", arr[i]);  printf("\n");  }  void bubbleSort(int arr[], int n){  int steps=0;  for(int i=0; i<n-1; i++){  for(int j=0; j<(n-1-i); j++){  if(arr[j]>arr[j+1]){  int temp=arr[j];  arr[j]=arr[j+1];  arr[j+1]=temp;  }  }  steps++;  }  printf("NUMBER OF STEPS NORMAL: %d\n", steps);  }  Time Complexity:  Best Case: O(n2)  Worst Case: O(n2)  **ii. Selection Sort:**  Source Code:  #include <stdio.h>  void selectionSort(int arr[], int n) {  int i, j, minIdx, temp;  for (i = 0; i < n-1; i++) {  minIdx = i;  for (j = i+1; j < n; j++) {  if (arr[j] < arr[minIdx])  minIdx = j;  }  temp = arr[minIdx];  arr[minIdx] = arr[i];  arr[i] = temp;  }  }  int main() {  int n;  printf("Enter the size of the array: ");  scanf("%d", &n);  int arr[n];  printf(“Enter the elements:\n”);  for (int i = 0; i < n; i++) {  scanf("%d", &arr[i]);  }  selectionSort(arr, n);  printf("Sorted array: \n");  for (int i = 0; i < n; i++)  printf("%d\t", arr[i]);  printf("\n");  return 0;  }  Time Complexity:  Best Case: O(n2)  Worst Case: O(n2)  **iii. Insertion Sort:**  Source Code:  #include <stdio.h>  void insertionSort(int arr[], int n) {  int i, key, j;  for (i = 1; i < n; i++) {  key = arr[i];  j = i - 1;  while (j >= 0 && arr[j] > key) {  arr[j + 1] = arr[j];  j = j - 1;  }  arr[j + 1] = key;  }  }  int main() {  int n;  printf("Enter the size of the array: ");  scanf("%d", &n);  int arr[n];  for (int i = 0; i < n; i++) {  scanf("%d", &arr[i]);  }  insertionSort(arr, n);  printf("Sorted array: \n");  for (int i = 0; i < n; i++)  printf("%d\t", arr[i]);  printf("\n");  return 0;  }  Time Complexity:  Best Case: O(n)  Worst Case: O(n2)  **Practice Problem: Bubble Sort Optimised**  Source Code:  #include <stdio.h>  void bubbleSort(int arr[], int n);  void bubbleSortOPTIMIZED(int arr[], int n);  int main(){  int n;  printf("Enter the size of the array: ");  scanf("%d", &n);  int arr[n], arr2[n];  printf("Enter the elements: \n");  for(int i=0; i<n; i++){  int temp;  scanf("%d", &temp);  arr[i]=temp;  arr2[i]=temp;  }  bubbleSort(arr, n);  bubbleSortOPTIMIZED(arr2, n);  printf("SORTED ARRAY 1: \n");  for(int i=0; i<n; i++)  printf("%d\t", arr[i]);  printf("\n");  printf("SORTED ARRAY 2: \n");  for(int i=0; i<n; i++)  printf("%d\t", arr2[i]);  printf("\n");  }  void bubbleSort(int arr[], int n){  int steps=0;  for(int i=0; i<n-1; i++){  for(int j=0; j<(n-1-i); j++){  if(arr[j]>arr[j+1]){  int temp=arr[j];  arr[j]=arr[j+1];  arr[j+1]=temp;  }  }  steps++;  }  printf("NUMBER OF STEPS NORMAL: %d\n", steps);  }  void bubbleSortOPTIMIZED(int arr[], int n){  int flag=0;  int steps=0;  for(int i=0; i<n-1; i++){  for(int j=0; j<(n-1-i); j++){  if(arr[j]>arr[j+1]){  flag++;  int temp=arr[j];  arr[j]=arr[j+1];  arr[j+1]=temp;  }  }  steps++;  if(flag==0){break;}    }  printf("NUMBER OF STEPS OPTIMIZED: %d\n", steps);  }  Time Complexity:  Best Case: O(n)  Worst Case: O(n2) |
| **Output Screenshots** | **BUBBLE SORT (OPTIMIZED AND UPOPTIMIZED):**  **Best Case:**    **Average Case:**    **Worst Case:**    **SELECTION SORT:**  **Best Case:**    **Average Case:**    **Worst Case:**    **INSERTION SORT:**  **Best Case:**    **Average Case:**    **Worst Case:** |
| **Practice questions** | 1. Implement Optimized bubble sort 2. o/p screenshot |
| **Conclusion** | Thus we have studied different sorting algorithms and their time complexities. |