LAB

REPORT

CSE 114 : Data Structure and Algorithms Sessional

|  |  |
| --- | --- |
| PREPARED BY  Mehrin Farzana  ID: 2101013  Session: 2021-2022  Date: 22/07/2023 | SUPERVISED BY  Suman Saha  Lecturer  Department of IRE, BDU |

|  |  |
| --- | --- |
| logo1 | BANGABANDHU SHEIKH MUJIBUR RAHMAN DIGITAL UNIVERSITY  (BDU) |
|  |  |

**List of Problems**

1. Write down a program that implements three sorting algorithms (bubble sort, selection sort, and insertion sort). Using a switch statement to choose the desired algorithm.
2. Consider a random array of n different sizes. Now write down a program that measures and records the execution time for each sorting algorithm (bubble sort, selection sort, and insertion sort) to sort the generated arrays. Repeat the experiment multiple times for each input size and calculate the average execution time.
3. Implement optimized versions of the sorting algorithms (bubble sort, selection sort, and insertion sort) to improve the performance.

**Problem No.:** 01

**Problem Statement:**

Write down a program that implements three sorting algorithms (bubble sort, selection sort, and insertion sort). Using a switch statement to choose the desired algorithm.

**Code:**

#include <stdio.h>

void bubble\_sort(int \*a, int n){

for(int i=0; i<n-1; i++){

for(int j=0; j<n-1-i; j++){

if(a[j]>a[j+1]){

int tmp = a[j];

a[j] = a[j+1];

a[j+1] = tmp;

}

}

}

}

void selection\_sort(int \*a, int n){

int min, min\_pos;

for(int j=0; j<n; j++){

min = a[j];

for(int i=j; i<n; i++){

if(min>=a[i]){

min = a[i];

min\_pos = i;

}

}

int tmp = a[j];

a[j] = a[min\_pos];

a[min\_pos] = tmp;

}

}

void insertion\_sort(int \*a, int n){

int key, i, j;

for(i=1; i<n; i++){

key = a[i];

for(j=i-1; j>=0 && key<a[j]; j--){

a[j+1]=a[j];

}

a[j+1]=key;

}

}

int main() {

int n, choose;

scanf("%d", &n);

int a[n];

for(int i=0; i<n; i++)

scanf("%d", &a[i]);

printf("Enter 1 for selecting Bubble sorting algorithm\nEnter 2 for selecting Selection sorting algorithm\nEnter 3 for selecting Insertion sorting algorithm\n");

scanf("%d", &choose);

switch(choose){

case(1):

bubble\_sort(a,n);

break;

case(2):

selection\_sort(a,n);

break;

case(3):

insertion\_sort(a,n);

break;

}

for(int i=0; i<n; i++)

printf("%d ", a[i]);

return 0;

}

**Output:**

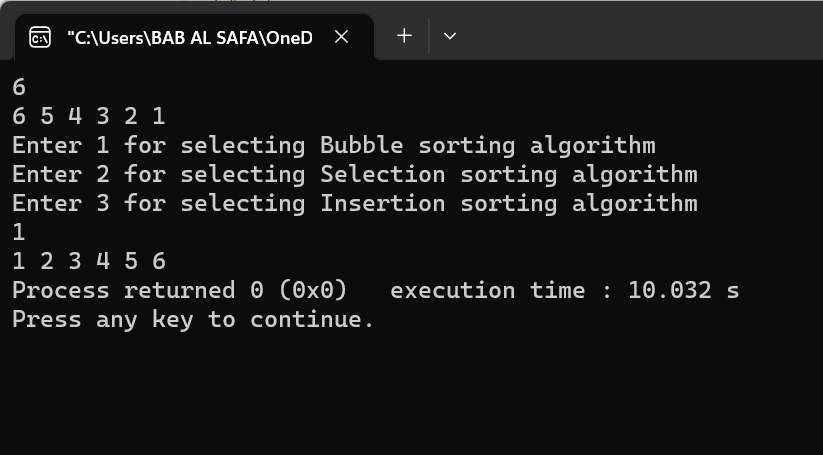


Fig 1.1: Output on console for case 1.

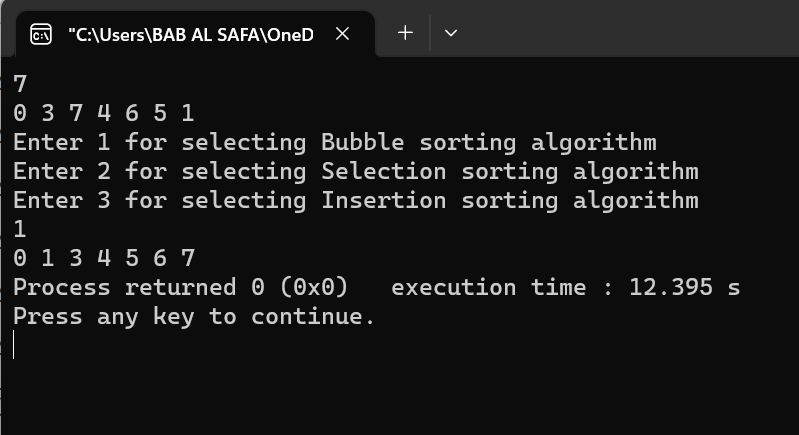


Fig 1.2: Output on console for case 2.

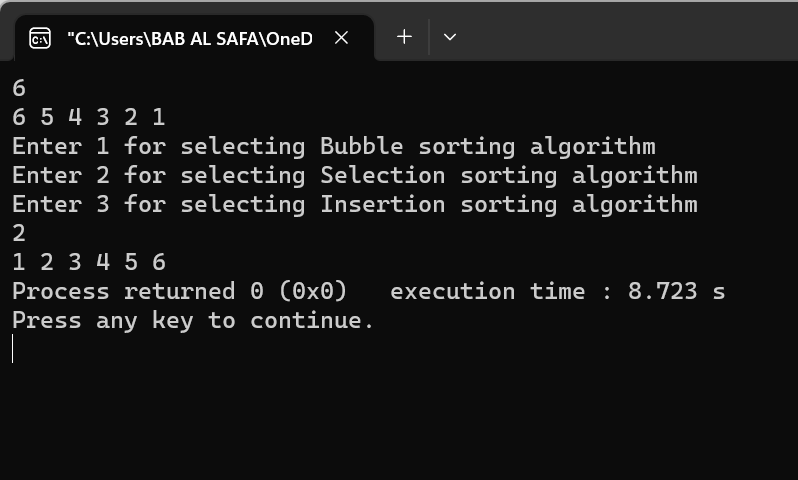
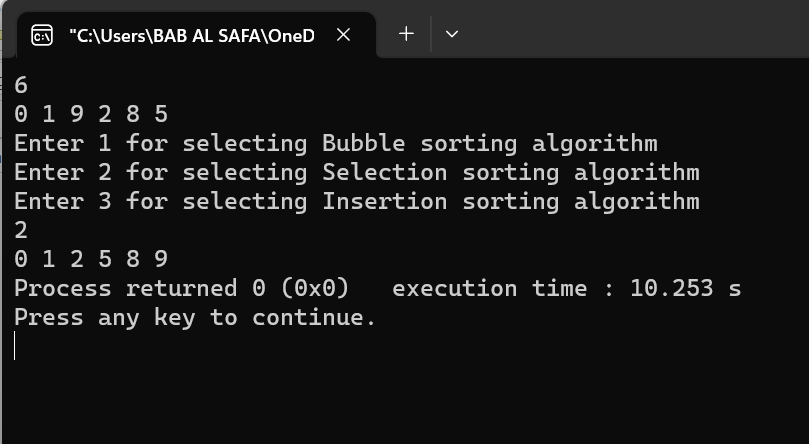


Fig 1.3: Output on console for case 3.

  
Fig 1.4: Output on console for case 4.

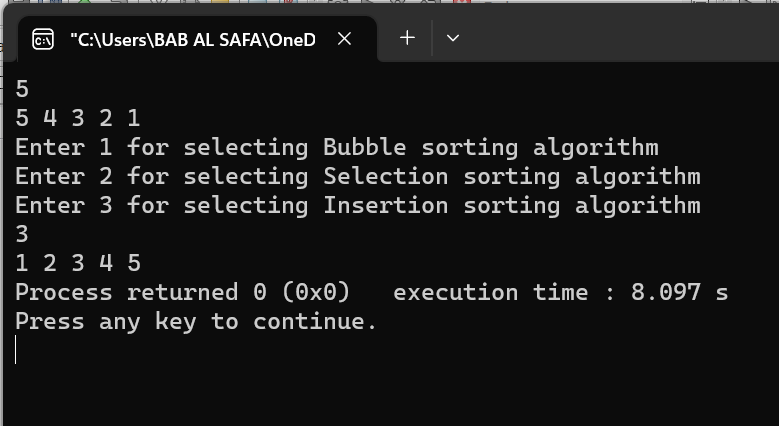


Fig 1.5: Output on console for case 5.

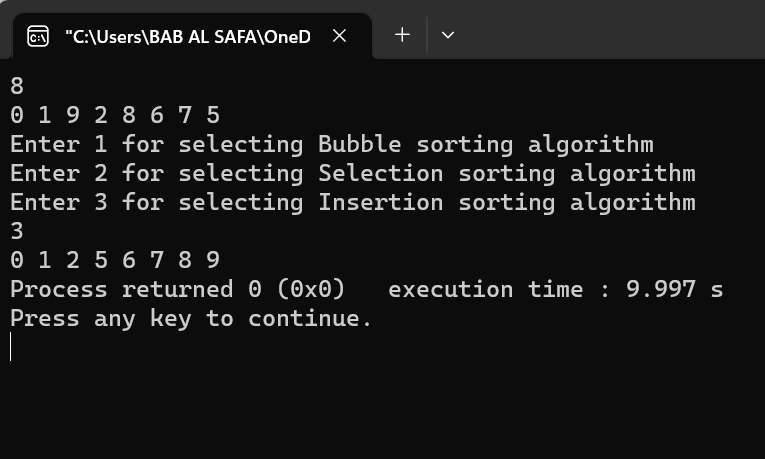


Fig 1.6: Output on console for case 6.

**Problem No.:** 02

**Problem Statement:**

Consider a random array of n different sizes. Now write down a program that measures and records the execution time for each sorting algorithm (bubble sort, selection sort, and insertion sort) to sort the generated arrays. Repeat the experiment multiple times for each input size and calculate the average execution time.

**Code:**

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

void bubble\_sort(int \*a, int n){

for(int i=0; i<n-1; i++){

for(int j=0; j<n-1-i; j++){

if(a[j]>a[j+1]){

int tmp = a[j];

a[j] = a[j+1];

a[j+1] = tmp;

}

}

}

}

void selection\_sort(int \*a, int n){

int min, min\_pos;

for(int j=0; j<n; j++){

min = a[j];

for(int i=j; i<n; i++){

if(min>=a[i]){

min = a[i];

min\_pos = i;

}

}

int tmp = a[j];

a[j] = a[min\_pos];

a[min\_pos] = tmp;

}

}

void insertion\_sort(int \*a, int n){

int key, i, j;

for(i=1; i<n; i++){

key = a[i];

for(j=i-1; j>=0 && key<a[j]; j--){

a[j+1]=a[j];

}

a[j+1]=key;

}

}

int main() {

srand(time(NULL));

struct timespec start\_time, end\_time;

int n, s;

double sum\_bubble=0, sum\_selection=0, sum\_insertion=0;

scanf("%d", &n);

printf("Enter %d array sizes: ", n);

for(int i=0; i<n; i++){

scanf("%d", &s);

int a[s];

for(int j=0; j<s; j++){

a[j] = rand()%100;

}

clock\_gettime(CLOCK\_MONOTONIC, &start\_time);

bubble\_sort(a,n);

clock\_gettime(CLOCK\_MONOTONIC, &end\_time);

double elapsed\_time = (end\_time.tv\_sec - start\_time.tv\_sec) \* 1e9 + (end\_time.tv\_nsec - start\_time.tv\_nsec);

sum\_bubble+=elapsed\_time;

clock\_gettime(CLOCK\_MONOTONIC, &start\_time);

selection\_sort(a,n);

clock\_gettime(CLOCK\_MONOTONIC, &end\_time);

elapsed\_time = (end\_time.tv\_sec - start\_time.tv\_sec) \* 1e9 + (end\_time.tv\_nsec - start\_time.tv\_nsec);

sum\_selection+=elapsed\_time;

clock\_gettime(CLOCK\_MONOTONIC, &start\_time);

insertion\_sort(a,n);

clock\_gettime(CLOCK\_MONOTONIC, &end\_time);

elapsed\_time = (end\_time.tv\_sec - start\_time.tv\_sec) \* 1e9 + (end\_time.tv\_nsec - start\_time.tv\_nsec);

sum\_insertion+=elapsed\_time;

}

printf("Average execution time for Bubble sort: %f\nAverage execution time for Selection sort: %f\nAverage execution time for Insertion sort: %f\n", sum\_bubble/n, sum\_selection/n, sum\_insertion/n);

return 0;

}

**Output:**

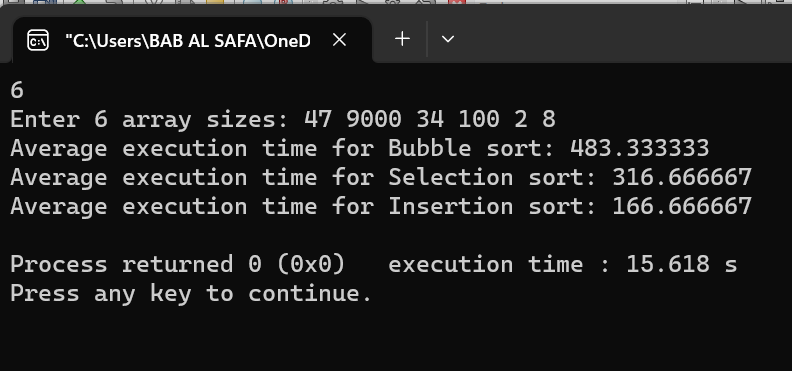


Fig 2.1: Output on console for case 1.

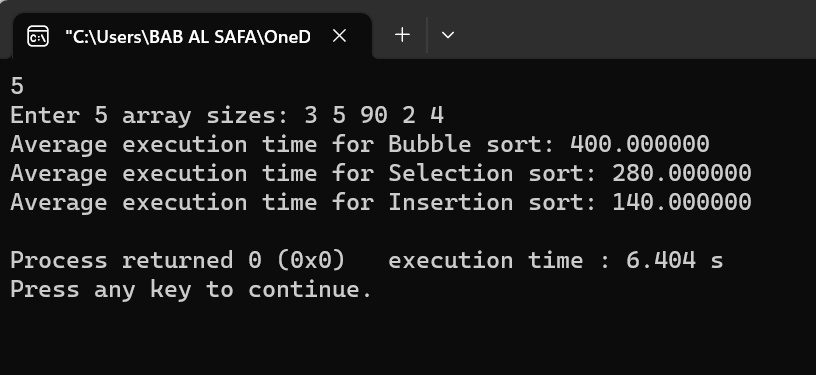


Fig 2.2: Output on console for case 2.

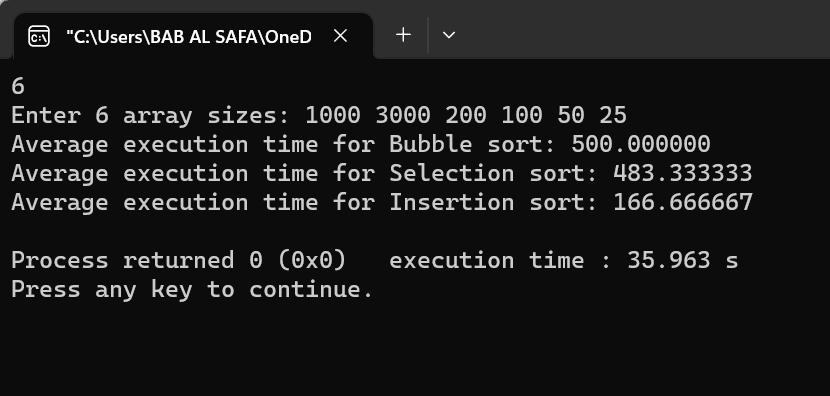


Fig 2.3: Output on console for case 3.

**Problem No.:** 03

**Problem Statement:**

Implement optimized versions of the sorting algorithms (bubble sort, selection sort, and insertion sort) to improve the performance.

**Code:**

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

void bubble\_sort(int \*a, int n){

int flag = 0;

for(int i=0; i<n-1; i++){

flag=0;

for(int j=0; j<n-1-i; j++){

if(a[j]>a[j+1]){

flag=1;

int tmp = a[j];

a[j] = a[j+1];

a[j+1] = tmp;

}

}

if(!flag){

break;

}

}

}

void selection\_sort(int \*a, int n){

int min, min\_pos,max,max\_pos;

for(int j=0; j<n-1-j; j++){

min = a[j];

max = a[j];

for(int i=j; i<n-j; i++){

if(min>a[i]){

min = a[i];

min\_pos = i;

}

if(max<a[i]){

max=a[i];

max\_pos=i;

}

}

int tmp = a[j];

a[j] = a[min\_pos];

a[min\_pos] = tmp;

if(a[min\_pos]==max){

tmp = a[n-j-1];

a[n-j-1] = a[min\_pos];

a[min\_pos] = tmp;

}

else{

tmp = a[n-j-1];

a[n-j-1] = a[max\_pos];

a[max\_pos] = tmp;

}

}

}

void insertion\_sort(int \*a, int n){

int key, i, j;

for(i=1; i<n; i++){

key = a[i];

for(j=i-1; j>=0 && key<a[j]; j--){

a[j+1]=a[j];

}

a[j+1]=key;

}

}

int main() {

int n, s, sum\_bubble=0, sum\_selection=0, sum\_insertion=0;

srand(time(NULL));

struct timespec start\_time, end\_time;

scanf("%d", &n);

printf("Enter %d array sizes: ", n);

for(int i=0; i<n; i++){

scanf("%d", &s);

int a[s];

for(int j=0; j<s; j++){

a[j] = rand()%100;

}

clock\_gettime(CLOCK\_MONOTONIC, &start\_time);

bubble\_sort(a,n);

clock\_gettime(CLOCK\_MONOTONIC, &end\_time);

double elapsed\_time = (end\_time.tv\_sec - start\_time.tv\_sec) \* 1e9 + (end\_time.tv\_nsec - start\_time.tv\_nsec);

sum\_bubble+=elapsed\_time;

clock\_gettime(CLOCK\_MONOTONIC, &start\_time);

selection\_sort(a,n);

clock\_gettime(CLOCK\_MONOTONIC, &end\_time);

elapsed\_time = (end\_time.tv\_sec - start\_time.tv\_sec) \* 1e9 + (end\_time.tv\_nsec - start\_time.tv\_nsec);

sum\_selection+=elapsed\_time;

clock\_gettime(CLOCK\_MONOTONIC, &start\_time);

insertion\_sort(a,n);

clock\_gettime(CLOCK\_MONOTONIC, &end\_time);

elapsed\_time = (end\_time.tv\_sec - start\_time.tv\_sec) \* 1e9 + (end\_time.tv\_nsec - start\_time.tv\_nsec);

sum\_insertion+=elapsed\_time;

}

printf("Average execution time for Bubble sort: %f\nAverage execution time for Selection sort: %f\nAverage execution time for Insertion sort: %f\n", sum\_bubble/n, sum\_selection/n, sum\_insertion/n);

return 0;

}

**Output:**

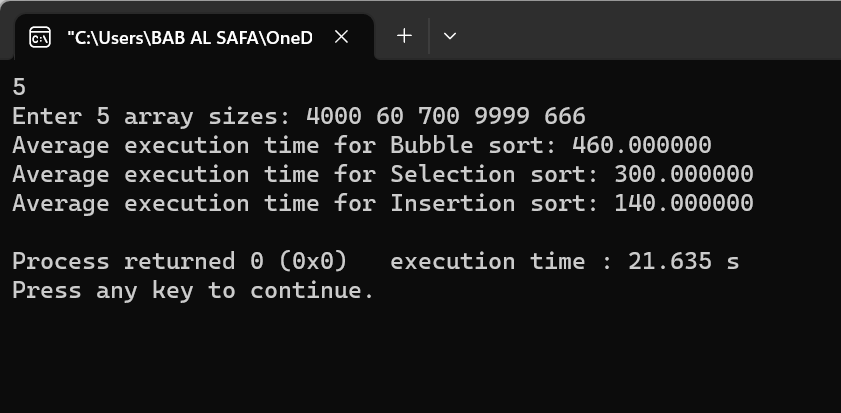


Fig 3.1: Output on console for case 1.

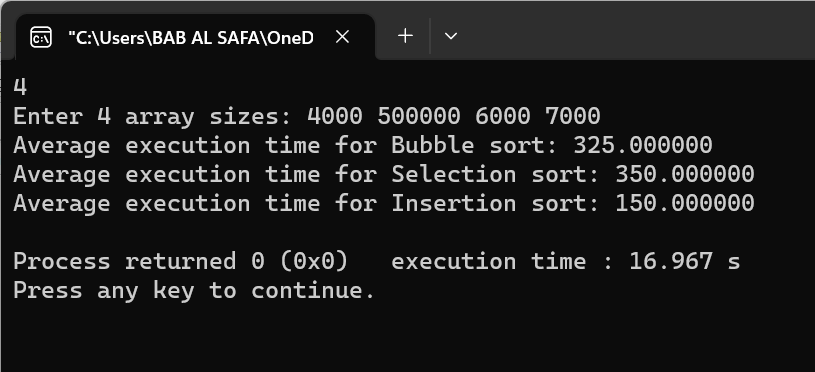


Fig 3.2: Output on console for case 2.