Student Name:- Chaudhary Hamdan

Student Roll No.:- 1905387

Algorithm Lab. Class Assignment-4

CSE Group 1

Date: - 30th July 2021

1. **Write a C program for bubble sort to**
   - **I. Compare the time complexity with the given data set given below and calculate the time complexity based on the CPU clock.**
   - **II. Plot a graph showing the comparison (n, the input data Vs. CPU times)**

Sl No.	Value of n	Bobble Sort (Time Complexity)		
		Best case	Average case	Worst case
1	5000	0.000000	0.081000	0.071000
2	10000	0.000000	0.330000	0.359000
3	15000	0.000000	0.807000	0.793000
4	20000	0.001000	1.346000	1.616000
5	25000	0.000000	1.952000	2.348000
6	30000	0.001000	2.778000	3.601000
7	35000	0.000000	4.184000	4.841000
8	40000	0.000000	5.691000	6.707000
9	45000	0.000000	6.954000	8.340000
10	50000	0.000000	8.543000	10.418000

Program

```
// Author: Chaudhary Hamdan


#include <stdio.h>
#include <time.h>
#include <stdlib.h>


#define sf(x)            scanf("%d", &x)
#define pf               printf
#define pfs(x)           printf("%d ", x)
#define pfn(x)           printf("%d\n", x)
#define pfc(x)           printf("%d, ", x)
#define F(i,x,y)         for(int i = x; i < y; i++)
#define FI(i,x,y,inc)     for(int i = x; i < y; i += inc)
#define RF(i,x,y)         for(int i = x; i >= y; i--)
#define pfa(i,a,n)        for(int i = 0; i < n-1; i++) printf("%d ",a[i]); printf("%d\n", a[n-1]);



void i_o_from_file() {

#ifndef ONLINE_JUDGE
    freopen("C:\\Users\\KIIT\\input", "r", stdin);
    freopen("C:\\Users\\KIIT\\output", "w", stdout);
#endif
}
void swap(int *x, int *y)
{
    int temp = *x;
```

```
    *x = *y;
    *y = temp;
}

void bubbleSort(int arr[], int n)
{
    int i, j;
    for (i = 0; i < n - 1; i++) {
        int swaps = 0;
        for (j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                swap(&arr[j], &arr[j + 1]);
                swaps++;
            }
        }
        if (swaps == 0) {
            break;
        }
    }
}


int main() {

    i_o_from_file();

    /* ********************************************** */
```

```
pf("n\t\tbest\t\tavg\t\t\tworst\n\n");

int sizes;
sf(sizes);

F(i, 0, sizes) {
	int n;
	sf(n);

	pf("%d\t", n);
	int arr[n];
	time_t start, end;
	double time;

	// Best
	F(j, 0, n) {
		arr[j] = j + 1;
	}
	start = clock();
	bubbleSort(arr, n);
	end = clock();

	time = (end - start) * 1.0 / CLOCKS_PER_SEC;

	pf("%f\t", time);

	// Avg
	F(j, 0, n) {
```

```
                arr[j] = n - j;
        }
        start = clock();
        bubbleSort(arr, n);
        end = clock();

        time = (end - start) * 1.0 / CLOCKS_PER_SEC;

        pf("%f\t", time);

        // Worst
        F(j, 0, n) {
                arr[j] = rand() % 10000;
        }

        start = clock();
        bubbleSort(arr, n);
        end = clock();

        time = (end - start) * 1.0 / CLOCKS_PER_SEC;

        pf("%f\n", time);
    }

    return 0;
}
```

Output

input

```
10
5000
10000
15000
20000
25000
30000
35000
40000
45000
50000
```

output

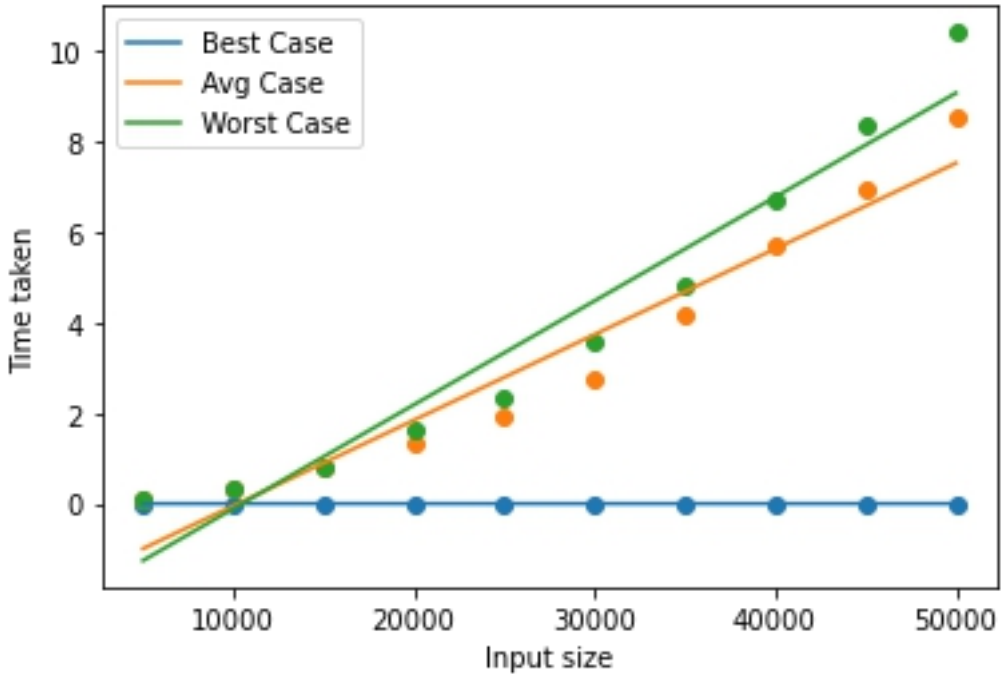
```
n       best        avg         worst

5000    0.000000    0.081000    0.071000
10000   0.000000    0.330000    0.359000
15000   0.000000    0.807000    0.793000
20000   0.001000    1.346000    1.616000
25000   0.000000    1.952000    2.348000
30000   0.001000    2.778000    3.601000
35000   0.000000    4.184000    4.841000
40000   0.000000    5.691000    6.707000
45000   0.000000    6.954000    8.340000
50000   0.000000    8.543000    10.418000
```

Tab Size: 4 C

31... ENG 13:39

Graph



2. **Write a C program for selection sort to**
    I. **Compare the time complexity with the given data set given below and calculate the time complexity based on the CPU clock.**
    II. **Plot a graph showing the comparison (n, the input data Vs. CPU times)**

Sl No.	Value of n	Selection Sort (Time Complexity)		
		Best case	Average case	Worst case
1	5000	0.040000	0.028000	0.032000
2	10000	0.126000	0.130000	0.146000
3	15000	0.313000	0.347000	0.330000
4	20000	0.552000	0.501000	0.537000
5	25000	0.922000	0.919000	0.836000
6	30000	1.186000	1.157000	1.248000
7	35000	1.706000	1.559000	1.581000
8	40000	2.269000	1.991000	2.259000
9	45000	2.605000	2.671000	2.654000
10	50000	3.411000	3.240000	3.788000

Program

```
// Author: Chaudhary Hamdan


#include <stdio.h>
#include <time.h>
#include <stdlib.h>


#define sf(x)            scanf("%d", &x)
#define pf               printf
#define pfs(x)           printf("%d ", x)
#define pfn(x)           printf("%d\n", x)
#define pfc(x)           printf("%d, ", x)
#define F(i,x,y)         for(int i = x; i < y; i++)
#define FI(i,x,y,inc)     for(int i = x; i < y; i += inc)


void i_o_from_file() {


#ifndef ONLINE_JUDGE
    freopen("C:\\Users\\KIIT\\input", "r", stdin);
    freopen("C:\\Users\\KIIT\\output", "w", stdout);
#endif
}
void swap(int *x, int *y)
{
    int temp = *x;
    *x = *y;
    *y = temp;
}
```

```
void selectionSort(int arr[], int n)
{
    int i, j, min_idx;

    for (i = 0; i < n - 1; i++)
    {
            min_idx = i;
            for (j = i + 1; j < n; j++) {
                    if (arr[j] < arr[min_idx]) {
                            min_idx = j;
                    }
            }

            swap(&arr[min_idx], &arr[i]);
    }
}

int main() {

    i_o_from_file();

    /* ********************************************** */

    pf("n\t\tbest\t\tavg\t\t\tworst\n\n");

    int sizes;
    sf(sizes);
```

```
F(i, 0, sizes) {
        int n;
        sf(n);

        pf("%d\t", n);
        int arr[n];
        time_t start, end;
        double time;

        // Best
        F(j, 0, n) {
                arr[j] = j + 1;
        }

        start = clock();
        selectionSort(arr, n);
        end = clock();

        time = (end - start) * 1.0 / CLOCKS_PER_SEC;

        pf("%f\t", time);

        // Avg
        F(j, 0, n) {
                arr[j] = n - j;
        }
```

```
            start = clock();
            selectionSort(arr, n);
            end = clock();

            time = (end - start) * 1.0 / CLOCKS_PER_SEC;

            pf("%f\t", time);



            // Worst
            F(j, 0, n) {
                  arr[j] = rand() % 10000;
            }

            start = clock();
            selectionSort(arr, n);
            end = clock();

            time = (end - start) * 1.0 / CLOCKS_PER_SEC;

            pf("%f\n", time);


      }

      return 0;
}
```

Output

input

```
10
5000
10000
15000
20000
25000
30000
35000
40000
45000
50000
```

output

```
n       best        avg         worst

5000    0.040000    0.028000    0.032000
10000   0.126000    0.130000    0.146000
15000   0.313000    0.347000    0.330000
20000   0.552000    0.501000    0.537000
25000   0.922000    0.919000    0.836000
30000   1.186000    1.157000    1.248000
35000   1.706000    1.559000    1.581000
40000   2.269000    1.991000    2.259000
45000   2.605000    2.671000    2.654000
50000   3.411000    3.240000    3.788000
```

Tab Size: 4 C

31... ENG 13:41

Graph