**DESIGN AND ANALYSIS OF ALGORITHMS**

**LAB ASSIGNMAENT**

**Quick Sort Implementation and Time Complexity Analysis**

**Submitted BY:**

**Gaurav Yadav**

**11911038**

**CSE**

* **Source Code:**

#include <stdio.h>

#include <stdlib.h>

#define MAX 100

void random\_shuffle(int arr[])

{

srand(time(NULL));

int i, j, temp;

for (i = MAX - 1; i > 0; i--)

{

j = rand()%(i + 1);

temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

void swap(int \*a, int \*b)

{

int temp;

temp = \*a;

\*a = \*b;

\*b = temp;

}

int partion(int arr[], int p, int r)

{

int pivotIndex = p + rand()%(r - p + 1);

int pivot;

int i = p - 1;

int j;

pivot = arr[pivotIndex];

swap(&arr[pivotIndex], &arr[r]);

for (j = p; j < r; j++)

{

if (arr[j] < pivot)

{

i++;

swap(&arr[i], &arr[j]);

}

}

swap(&arr[i+1], &arr[r]);

return i + 1;

}

void quick\_sort(int arr[], int p, int q)

{

int j;

if (p < q)

{

j = partion(arr, p, q);

quick\_sort(arr, p, j-1);

quick\_sort(arr, j+1, q);

}

}

int main()

{

int i;

int arr[MAX];

for (i = 0; i < MAX; i++)

arr[i] =(rand()%901)+100;

random\_shuffle(arr);

quick\_sort(arr, 0, MAX-1);

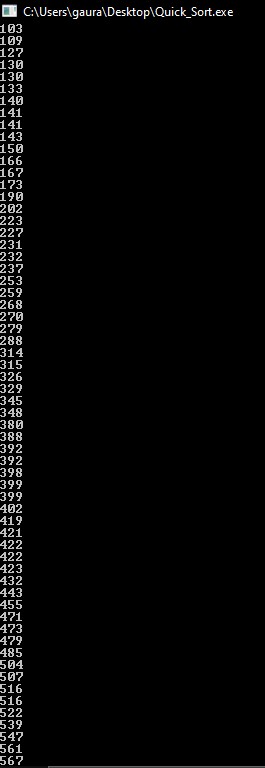
for (i = 0; i < MAX; i++)

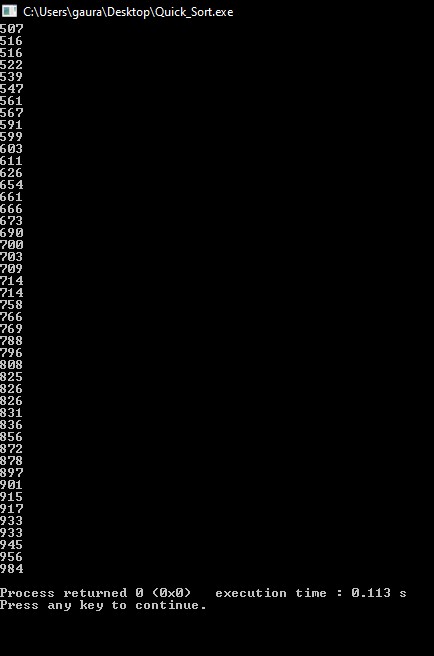
printf("%d \n", arr[i]);

return 0;

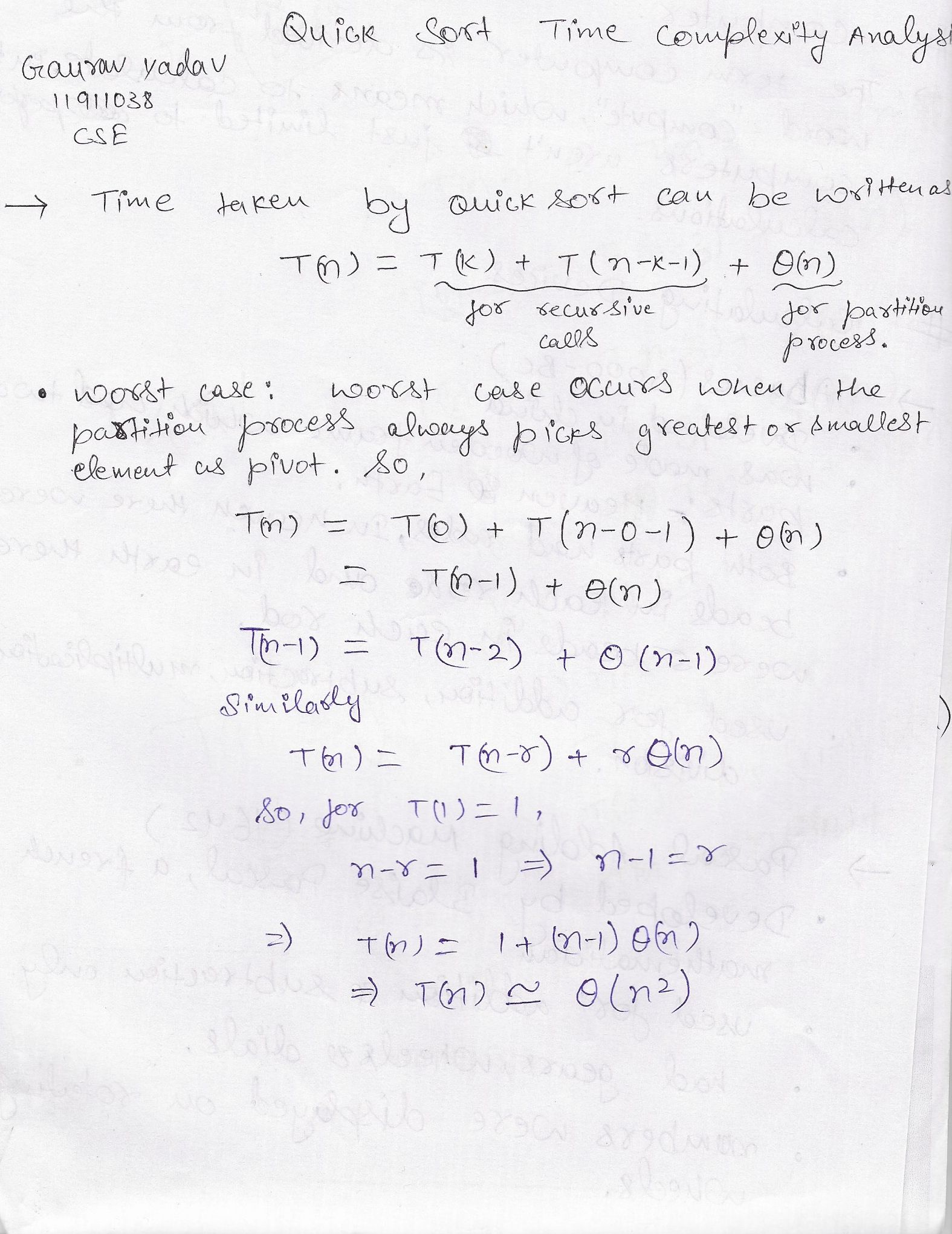
}

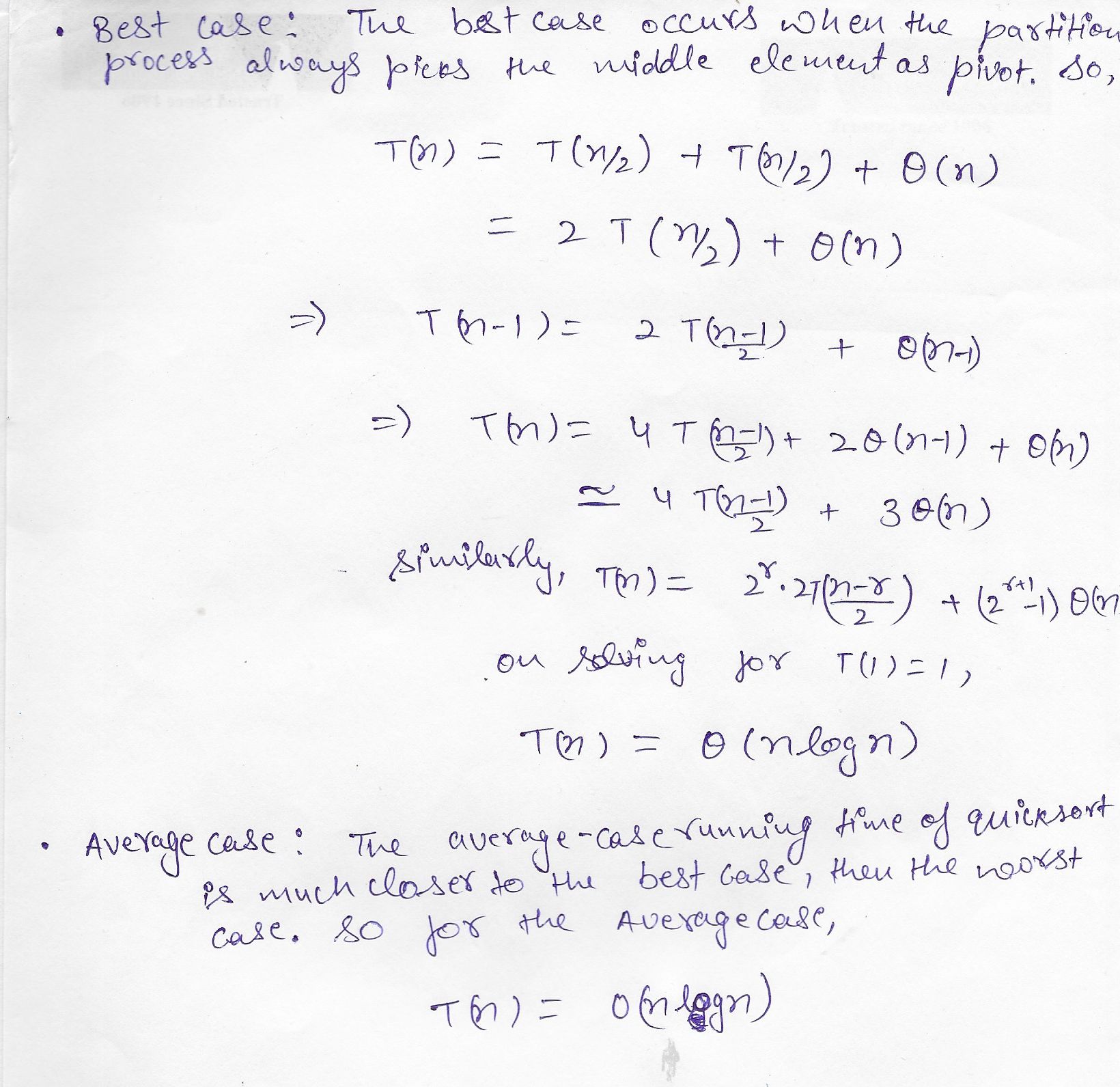
* **Output:**

****

****

* **Time Complexity Analysis:**

****

****