Aayush Shah
19BCE245
28 July 2021

Design and Analysis of Algorithms

Practical 2

- **Code :**

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


void swap(int *xp, int *yp){
   int temp = *xp;
   *xp = *yp;
   *yp = temp;
}

int partition (int arr[], int low, int high)
{
  int pivot = arr[high];
  int i = (low - 1);
  for (int j = low; j <= high - 1; j++){
       if (arr[j] < pivot){
            i++;
            swap(&arr[i], &arr[j]);
       }
  }
  swap(&arr[i + 1], &arr[high]);
  return (i + 1);
}

void quickSort(int arr[], int low, int high){
  if (low < high){
       int pi = partition(arr, low, high);
       quickSort(arr, low, pi - 1);
```

```
30.       quickSort(arr, pi + 1, high);
31.   }
32.}
33.
34.void printArray(int arr[], int size){
35.   int i;
36.   for (i = 0; i < size; i++)
37.       printf("%d ",arr[i]);
38.}
39.
40.int main() {
41.   int lower = 0;
42.   int upper = 100000;
43.   int count = 1000000;
44.   int arr[count];
45.
46.   clock_t start, end;
47.   double cpu_time_used;
48.
49.   srand(time(0));
50.
51.   for (int i = 0; i < count; i++){
52.       int num = (rand() % (upper - lower + 1)) + lower;
53.       arr[i] = num;
54.   }
55.
56.//     int arr[] = {64, 34, 25, 12, 22, 11, 90};
57.   int n = sizeof(arr)/sizeof(arr[0]);
58.
59.   //QUICK SORT
60.   start = clock();
61.   quickSort(arr, 0, n-1);
62.   end = clock();
63.   cpu_time_used = ((double) (end - start)) /
   CLOCKS_PER_SEC;
64.   printf("Sorted array in %f seconds with QUICK sort:
   \n",cpu_time_used);
65.//     printArray(arr, n);
66.
67.
68.   return 0;
69.}
```

- **Execution time in seconds :**

Numbers	Time taken
500	0.000051
1000	0.000133
5000	0.000791
10000	0.001538
50000	0.007806
100000	0.016710
150000	0.023415
160000	0.025759
170000	0.028243
200000	0.031532
500000	0.082039
1000000	0.166498
1500000	0.259056
2000000	0.358336