Analysis & Design of Algorithms

Semester III 2018-19

Lab - 3

**Topics: Iterative Vs Recursion**

---

# INTRODUCTION

In this lab we would be implementation two sorting algorithm, quick sort and merge sort. The implementation would be carried out using iterative and recursive methods.

# EXERISE

1. Implement merge sort and quick sort using iterative and recursive methods. The number of inputs elements has to be passed from command line agruements. The elements has to be generated randomly within the code. Compute:
    a. Check the performance of program by varying the number of elements.
    b. Compute the time taken by each case (for particular number of inputs).
    c. plot a graph with number of inputs to time taken in seconds.
    d. Compute the time complexity of the sorting algorithms.

# HELP

## Random Numbers

The code to generate random numbers is given below

```
for (i = 0; i < N; i++)
  a[i] = rand()% 100000+1;
```

## Time

To compute the time taken by the program is given below

```
#include <time.h>
clock_t start, end;
double cpu_time_used;
start = clock();
  ... /* Do the work. */
end = clock();
cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;
```

## EXAMPLE

A sample example to write sorting algorithm.

```
int main(int argc, char *argv[])
{
  int i, N, sw, *a;
  clock_t start=clock(); // clock begin
  srand(time(NULL));
  if (argc < 3)
    printf("You have not entered enough arguments: N and sw required!\n ");
  else
  {
    N = atoi(argv[1]); // converts string to int N= number of elements
    a = malloc(N*sizeof(int));
    sw = atoi(argv[2]);
    srand(time(0));
    if (sw)
      for (i = 0; i < N; i++)
      a[i] = rand()% 100000+1; computes the max number of numbers
    else
      while (scanf("%d", &a[N]) == 1)
      N++;
      printf("Initial: ");
      for(i = 0; i < N; i++)  // printing initail unsorted elements
      printf("%3d ", a[i]);
      //quicksort(a, 0, N-1); // calling sorting algorithm
      printf("\n");
      printf("\nAfter: "); // printing sorted elements
      for(i = 0; i < N; i++)
      printf("%3d ", a[i]);
      printf("\n");
    }
  clock_t end=clock(); // clock ends
```

```
31   double seconds=((double)(end-start))/CLOCKS_PER_SEC; // time
32   printf("\n  Time taken: %f \n", seconds);
33   return 0;
34 }
```

## PRACTICE: HOMEWORK

Implement a simple quick sort algorithm on an array of student records, where each record must contain the following: Student:
Name : single word (at most 20 characters)
Marks : unsigned integer
You must sort the records based on marks of the students. Use the last element as the pivot. Also implement a partial quick sort algorithm as per the instructions given in the following.

1. readData : Key:0 = Indicates that next M lines will contain data to be read. Each line will contain student's name followed by marks, separated by space. Input format: 0 - M.

2. Call complete quick sort and print final result.

Sample Input.txt

MAR 9937
MAY 30344
NOV 31441
AUG 17078
APR 24489
JAN 22172
DEC 23239
JUL 23072
FEB 9718
JUN 29687
OCT 31011
SEP 29969

# PRACTICE CODING


**Figure 1:** Learning=Theory+Practice