Analysis & Design of Algorithms

Semester III 2018-19

Lab - 4

Topics: To Compute the performance of sorting algorithms

INTRODUCTION

In this lab we would be evaluating the performance of sorting algorithm. Performance of insertion sort, selection sort bubble sort and heap sort would be carried out.

EXERCISE

1. Implement insertion sort, selection sort bubble sort and heap sort. The number of inputs elements has to be passed from command line arguments. 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.

    e. compute the time taken for sorted array. A sample example of quick sort has been attached.

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.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main(int argc, char *argv[])
{
  int i, N, sw, *a;
  clock_t start1=clock();
  srand(time(NULL));
  if (argc < 2)
  printf("You have not entered enough arguments: N and sw required!\n ");
  else
  {
    N = atoi(argv[1]); // converts string to int
    a = malloc(N*sizeof(int)); // dymanic memeory allocation
    srand(time(0));
    for (i = 0; i < N; i++)
    a[i] = rand()% 100000+1;
    printf("Initial: ");
    for(i = 0; i < N; i++)
    printf("%3d ", a[i]);
    printf("\n");
    /* inital calling sorting algorithm*/
    //insertSort(a, N);
    mergesort( a,0, N-1);
    //quicksort(a, 0, N-1);
    printf("\n");
```

```
27      // quickSortIterative( a, 0, N - 1 );
28      printf("\nAfter: ");
29      // for(i = 0; i < N; i++)
30      //  printf("%3d ", a[i]);
31      printf("\n");
32    }
33    clock_t end1=clock();
34    double seconds1=((double)(end1-start1))/CLOCKS_PER_SEC; // Computing time for
        unsorted array
35    printf("\n  Time taken: %f \n", seconds1);
36    clock_t start2=clock(); // computing time for sorted
37    /*calling sorting algorithm on sorted array */
38    //insertSort(a, N);
39    mergesort( a,0, N-1);
40    //quicksort(a, 0, N-1);
41    printf("\n");
42    // quickSortIterative( a, 0, N - 1 );
43    printf("\nworst case sorted arrary : ");
44    //  for(i = 0; i < N; i++)
45    //  printf("%3d ", a[i]);
46    clock_t end2=clock();
47    double seconds2=((double)(end2-start2))/CLOCKS_PER_SEC;
48    printf("\n  Time taken for sorted array: %f \n", seconds2);
49    printf("\n");
50  return 0;
51  }
```