|  |  |
| --- | --- |
|  | Department of Computer Engineering |

|  |  |
| --- | --- |
| Semester | 4th |
| Subject | Analysis Of Algorithms |
| Subject Professor In-charge | Sanjeev Dwivedi |
| Assisting Teachers |  |
| Laboratory |  |

|  |  |  |
| --- | --- | --- |
| Student Name | Pravin Padalkr | |
| Roll Number | 20102B0028 | |
| Grade and Subject Teacher’s Signature |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Experiment Number | | 02 | | | | | | | | | | | | | | |
| Experiment Title | |  | *TO IMPLEMENT QUICK SORT FOR RANGE OF INPUT AND PLOT TIME TAKEN* | | | | | | | | | | | | | |
| *WRT INPUT SIZE AND CINCLUDE COMPLEXITY.* | | | | | |  | | | | | | | |
| Resources / Apparatus Required | | Ms teams | | | | | |  | | | | | | | | |
| Objectives  (Skill Set / Knowledge  Tested / Imparted) | |  | | | | | | | | | | | | | | |
| Description | |  | | **QuickSort is a Divide and Conquer algorithm. It picks an** | | | | | | | | | | |  | |
| **element as pivot and partitions the given array around the** | | | | | | | | | | | |  |
| **picked pivot. There are many different versions of** | | | | | | |  | | | | |
| **quickSort that pick pivot in different ways.** | | | | | |  |
|  | | | | | |
| **Always pick first element as pivot.** | | | |  | |
| **Always pick last element as pivot (implemented below)** | | | | | | | | |  | | |
| **Pick a random element as pivot.** | | |  | | | | | |
| **Pick median as pivot.** |  | |
| **The key process in quickSort is partition(). Target of** | | | | | | | |  |
| **partitions is, given an array and an element x of array as** | | | | | | | | | | |  |
| **pivot, put x at its correct position in sorted array and put** | | | | | | | | | | |
| **all smaller elements (smaller than x) before x, and put all** | | | | | | | | | | |
| **greater elements (greater than x) after x. All this should** | | | | | | | | | |  |
| **be done in linear time.** | |  | | | | | | | |
|  | | Department of Computer Engineering | | | | | | | | | | | | | | | | |

|  |  |
| --- | --- |
|  | • |
| Program | #include <stdio.h>  #include <time.h> #include <stdlib.h> swap(int \*a, int \*b)  { int t = \*a; \*a = \*b;  \*b = t; }  int partition(int a[], int low, int high)  {  int i, j, t;  int x = a[low]; i = low; j = high; while (i <= j)  {  while (a[i] <= x) i++;  while (x < a[j])  j--; if (i < j)  {  swap(&a[i], &a[j]);  }  }  swap(&a[low], &a[j]); return j;  } quicksort(int a[], int low, int high)  {    int pos;  if (low < high)  {  pos = partition(a, low, high);    quicksort(a, low, pos - 1);    quicksort(a, pos + 1, high);  } }  int main()  {  int i, j, \*inp;  FILE \*input, \*output; |

|  |  |  |
| --- | --- | --- |
|  | | Department of Computer Engineering      clock\_t end, strt; float total;    input = fopen("input.txt", "w");  output = fopen("output.txt", "w");    for (i = 5000; i <= 25000; i = i + 5000)  {  inp = (int \*)malloc(i \* sizeof(int)); printf("\nFor i = %d", i);    for (j = 0; j < i; j++)  {  fprintf(input, "%d ", rand() % i);  }  fclose(input);  input = fopen("input.txt", "r");  for (j = 0; j < i; j++)  {  fscanf(input, "%d", &inp[j]);  }  strt = clock(); quicksort(inp, 0, i - 1); end = clock();    total = (float)(end - strt) / CLOCKS\_PER\_SEC; for (j = 0; j < i; j++)  {  fprintf(output, "%d ", inp[j]);  }  printf("\nTime taken : %f\n\n", total);  }  fclose(input); fclose(output); return 0;  } |
|  | | Department of Computer Engineering | | |

|  |  |  |
| --- | --- | --- |
| output | |  |
|  | | Department of Computer Engineering | | |

|  |  |
| --- | --- |
|  |  |
|  |  |
| Conclusion | Then time complexity of Quick Sort is O(nlogn) |