|  |  |
| --- | --- |
| **Name** | Manish Shashikant Jadhav |
| **UID** | 2023301005 |
| **Subject** | Design and Analysis of Algorithms (DAA) |
| **Experiment No.** | 2 |
| **Aim** | Experiment based on divide and conquer approach. |
| **Code** | #include <stdio.h>  #include <stdlib.h>  #include <time.h>  void merge(int arr[], int l, int m, int r)  {      int i, j, k;      int n1 = m - l + 1;      int n2 = r - m;      int L[n1], R[n2];      for (i = 0; i < n1; i++)          L[i] = arr[l + i];      for (j = 0; j < n2; j++)          R[j] = arr[m + 1 + j];      i = 0;      j = 0;      k = l;      while (i < n1 && j < n2)      {          if (L[i] <= R[j])          {              arr[k] = L[i];              i++;          }          else          {              arr[k] = R[j];              j++;          }          k++;      }      while (i < n1)      {          arr[k] = L[i];          i++;          k++;      }      while (j < n2)      {          arr[k] = R[j];          j++;          k++;      }  }  void mergeSort(int arr[], int l, int r)  {      if (l < r)      {          int m = l + (r - l) / 2;          mergeSort(arr, l, m);          mergeSort(arr, m + 1, r);          merge(arr, l, m, r);      }  }  void quickSort(int arr[], int low, int high)  {      if (low < high)      {          int pi = partition(arr, low, high);          quickSort(arr, low, pi - 1);          quickSort(arr, pi + 1, high);      }  }  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 swap(int \*a, int \*b)  {      int temp = \*a;      \*a = \*b;      \*b = temp;  }  void main()  {      srand(time(NULL));      FILE \*fileptr;      FILE \*fileptr1;      int n = 100000;      fileptr = fopen("inp.txt", "w");      int arr[n];      for (int j = 0; j < n; j++)      {          arr[j] = rand() % 100000 + 1;          fprintf(fileptr, "%d\n", arr[j]);      }      fclose(fileptr);      fileptr1 = fopen("time.csv", "w");      printf("Block Size\tMerge Sort\tQuick Sort\n");      fileptr = fopen("inp.txt", "r");      for (int p = 99; p < n; p = p + 100)      {          int array[p + 1];          int array1[p + 1];          for (int j = 0; j < p; j++)          {              array[j] = arr[j];              fscanf(fileptr, "%1d", &array1[j]);          }          clock\_t begin = clock();          mergeSort(array, 0, p);          clock\_t mid = clock();          quickSort(array1, 0, p);          clock\_t end = clock();          double time\_spent = (double)(mid - begin);          double time\_spent1 = (double)(end - mid);          printf("%d\t%lf\t%lf\n", p + 1, time\_spent, time\_spent1);          fprintf(fileptr1, "%d,%lf,%lf\n", p + 1, time\_spent, time\_spent1);      }      printf("\n");  } |
| **Graphs** |  |
| **Pseudo Code & Example** |  |
| **Conclusion** | Hence, by completing this experiment I came to know about divide and conquer approach. |