|  |
| --- |
| # Python program for implementation of Merge Sort  def mergeSort(arr):      if len(arr) > 1:             # Finding the mid of the array          mid = len(arr)//2            # Dividing the array elements          L = arr[:mid]            # into 2 halves          R = arr[mid:]            # Sorting the first half          mergeSort(L)            # Sorting the second half          mergeSort(R)            i = j = k = 0            # Copy data to temp arrays L[] and R[]          while i < len(L) and j < len(R):              if L[i] <= R[j]:                  arr[k] = L[i]                  i += 1              else:                  arr[k] = R[j]                  j += 1              k += 1            # Checking if any element was left          while i < len(L):              arr[k] = L[i]              i += 1              k += 1            while j < len(R):              arr[k] = R[j]              j += 1              k += 1    # Code to print the list      def printList(arr):      for i in range(len(arr)):          print(arr[i], end=" ")      print()      # Driver Code  if \_\_name\_\_ == '\_\_main\_\_':      arr = [12, 11, 13, 5, 6, 7]      print("Given array is", end="\n")      printList(arr)      mergeSort(arr)      print("Sorted array is: ", end="\n")      printList(arr) |
| **Output:**  Given array is  12 11 13 5 6 7  Sorted array is  5 6 7 11 12 13 |
| // CPP Program to implement merge sort using  // multi-threading  #include <iostream>  #include <pthread.h>  #include <time.h>  // number of elements in array  #define MAX 20  // number of threads  #define THREAD\_MAX 4  using namespace std;  // array of size MAX  int a[MAX];  int part = 0;  // merge function for merging two parts  void merge(int low, int mid, int high)  {  int\* left = new int[mid - low + 1];  int\* right = new int[high - mid];  // n1 is size of left part and n2 is size  // of right part  int n1 = mid - low + 1, n2 = high - mid, i, j;  // storing values in left part  for (i = 0; i < n1; i++)  left[i] = a[i + low];  // storing values in right part  for (i = 0; i < n2; i++)  right[i] = a[i + mid + 1];  int k = low;  i = j = 0;  // merge left and right in ascending order  while (i < n1 && j < n2) {  if (left[i] <= right[j])  a[k++] = left[i++];  else  a[k++] = right[j++];  }  // insert remaining values from left  while (i < n1) {  a[k++] = left[i++];  }  // insert remaining values from right  while (j < n2) {  a[k++] = right[j++];  }  }  // merge sort function  void merge\_sort(int low, int high)  {  // calculating mid point of array  int mid = low + (high - low) / 2;  if (low < high) {  // calling first half  merge\_sort(low, mid);  // calling second half  merge\_sort(mid + 1, high);  // merging the two halves  merge(low, mid, high);  }  }  // thread function for multi-threading  void\* merge\_sort(void\* arg)  {  // which part out of 4 parts  int thread\_part = part++;  // calculating low and high  int low = thread\_part \* (MAX / 4);  int high = (thread\_part + 1) \* (MAX / 4) - 1;  // evaluating mid point  int mid = low + (high - low) / 2;  if (low < high) {  merge\_sort(low, mid);  merge\_sort(mid + 1, high);  merge(low, mid, high);  }  }  // Driver Code  int main()  {  // generating random values in array  for (int i = 0; i < MAX; i++)  a[i] = rand() % 100;  // t1 and t2 for calculating time for  // merge sort  clock\_t t1, t2;  t1 = clock();  pthread\_t threads[THREAD\_MAX];  // creating 4 threads  for (int i = 0; i < THREAD\_MAX; i++)  pthread\_create(&threads[i], NULL, merge\_sort,  (void\*)NULL);  // joining all 4 threads  for (int i = 0; i < 4; i++)  pthread\_join(threads[i], NULL);  // merging the final 4 parts  merge(0, (MAX / 2 - 1) / 2, MAX / 2 - 1);  merge(MAX / 2, MAX/2 + (MAX-1-MAX/2)/2, MAX - 1);  merge(0, (MAX - 1)/2, MAX - 1);  t2 = clock();  // displaying sorted array  cout << "Sorted array: ";  for (int i = 0; i < MAX; i++)  cout << a[i] << " ";  // time taken by merge sort in seconds  cout << "Time taken: " << (t2 - t1) /  (double)CLOCKS\_PER\_SEC << endl;  return 0;  }  **Output:**  **Sorted array**: 15 21 26 26 27 35 36 40 49 59 62 63 72 77 83 86 86 90 92 93  **Time taken**: 0.001023 |