# Name : Shrinivas Hatyalikar

# Div: TY-B (B2)

# Roll No: 26

**K-merge sort**

#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

typedef struct {

    int value;

    int index;

    int size;

    int end;

    int start;

} HeapNode;

void merge(int arr[], int left, int mid, int right){

    int i, j, k;

    int b[1000];

    for(i = left, j = mid + 1, k = left; i <= mid && j <= right; k++){

        if(arr[i] <= arr[j]){

            b[k] = arr[i++];

        }

        else{

            b[k] = arr[j++];

        }

    }

    while(i <= mid){

        b[k++] = arr[i++];

    }

    while(j <= right){

        b[k++] = arr[j++];

    }

    for(k = left; k <= right; k++){

        arr[k] = b[k];

    }

}

void mergeSort(int arr[],int left,int right,int n){

    if(left<right){

    int mid = (left+right)/2;

    mergeSort(arr,left,mid,n);

    mergeSort(arr,mid+1,right,n);

    merge(arr,left,mid,right);

    }

}

void k\_array\_sort(int arr[], int n, int k, int sublist\_size){

    for (int i = 0; i < k; i++) {

        if(i==k-1){

            int l = sublist\_size\*i;

            int temp[n-l];

            int x=0;

            for(int j=l; j<=n-1; ++j) temp[x++] = arr[j];

            mergeSort(temp, 0, n-l-1, n-l);

            x=0;

            for(int j=l; j<=n-1; ++j) arr[j] = temp[x++];

        }

        else{

            int l = sublist\_size\*i;

            int r = l+sublist\_size-1;

            int temp[sublist\_size];

            int x=0;

            for(int j=l; j<=r; ++j) temp[x++] = arr[j];

            mergeSort(temp, 0, sublist\_size-1, sublist\_size);

            x=0;

            for(int j=l; j<=r; ++j) arr[j] = temp[x++];

        }

    }

}

void min\_heapify(HeapNode \*heap, int index, int heap\_size) {

    int smallest = index;

    int left = 2 \* index + 1;

    int right = 2 \* index + 2;

    if (left < heap\_size && heap[left].value < heap[smallest].value) {

        smallest = left;

    }

    if (right < heap\_size && heap[right].value < heap[smallest].value) {

        smallest = right;

    }

    if (smallest != index) {

        HeapNode temp = heap[index];

        heap[index] = heap[smallest];

        heap[smallest] = temp;

        min\_heapify(heap, smallest, heap\_size);

    }

}

HeapNode\* build\_heap(int array[], int sublist\_size, int n, int k){

    HeapNode \*heap = malloc(k \* sizeof(HeapNode));

    // Initialize the heap and indexes

    for (int i = 0; i < k; i++) {

        if(i==k-1){

            int l = i \* sublist\_size;

            heap[i].value = array[l];

            heap[i].start = l;

            heap[i].index = i;

            heap[i].size = n-l;

            heap[i].end = n-1;

        }

        else{

            int l = i \* sublist\_size;

            heap[i].value = array[l];

            heap[i].start = l;

            heap[i].index = i;

            heap[i].size = sublist\_size;

            heap[i].end = l+sublist\_size-1;

        }

    }

    // Build the initial min heap

    for (int i = k / 2 - 1; i >= 0; i--) {

        min\_heapify(heap, i, k);

    }

    return heap;

}

void heap\_merge(int array[], int sublist\_size, int n, int k) {

    HeapNode\* heap = build\_heap(array,sublist\_size,n,k);

    // Merge the sorted subsets using the min heap property

    int final\_arr[n];

    for (int i = 0; i < n; i++) {

        final\_arr[i] = heap[0].value;  // Store the minimum element in the sorted array

        // Replace the minimum element with the next element from its subset

        // int start\_index = heap[0].start;

        int next\_index = heap[0].start + 1;

        if (next\_index <= heap[0].end) {

            heap[0].value = array[next\_index];

            heap[0].start += 1;

        } else {

            heap[0].value = INT\_MAX;  // Mark subset as fully merged

        }

        min\_heapify(heap, 0, k);  // Fix the min heap property

    }

    for (int i = 0; i < n; i++) array[i] = final\_arr[i];

    free(heap);

}

void k\_way(int arr[],int left,int right, int n, int k){

    int sublist\_size = n / k;

    k\_array\_sort(arr,n,k,sublist\_size);

    heap\_merge(arr,sublist\_size,n,k);

}

void Print(int arr[],int n){

    for(int i=0;i<n;i++){

        printf("%d ",arr[i]);

    }

    printf("\n");

}

int main(){

    int n;

    printf("Enter the number of the elements : ");

    scanf("%d", &n);

    int arr[n];

    for(int i=0; i<n; ++i){

        scanf("%d", &arr[i]);

    }

    printf("Unsorted array: \n");

    Print(arr,n);

    printf("Sorted array: \n");

    k\_way(arr,0,n-1,n,4);

    Print(arr,n);

    return 0;

}

Output :

