**Heap Sort**

# Heap Sort using Min-Heap

#include <stdio.h>

#include <stdlib.h>

// Max heap of length len and values stored in array

struct MinHeap{

int size;

int\* array;

};

// Function declarations

void createHeap(struct MinHeap\*);

void heapify(struct MinHeap\* , int );

void heapSort(struct MinHeap\*);

void swap(int\*,int\*);

void printArray(int\*, int);

int main(){

int numelems;

scanf("%d",&numelems);

int \* arr = (int\*) malloc(sizeof(int) \* numelems);

int i;

for(i=0;i<numelems;i++)

scanf("%d",&arr[i]);

struct MinHeap\* minHeap = (struct MinHeap\*) malloc(sizeof(struct MinHeap));

minHeap->size = numelems; // initialize size of heap

minHeap->array = arr; // Assign address of first element of array

createHeap(minHeap);

heapSort(minHeap);

printArray(minHeap->array,numelems);

return 0;

}

// heapSort function

void heapSort(struct MinHeap\* minHeap){

// Repeat following steps while heap size is greater than 1.

while (minHeap->size > 1){

// The smallest item in Heap is stored at the root.

// Replace it with the last item of the heap followed by reducing the size of heap by 1.

swap(&minHeap->array[0], &minHeap->array[minHeap->size - 1]);

--minHeap->size; // Reduce heap size

// heapify the root of tree.

heapify(minHeap, 0);

}

}

// function swap 2 integers

void swap(int\* num1, int\* num2){

int temp = \*num1;

\*num1 = \*num2;

\*num2 = temp;

}

// prints an array of given size

void printArray(int\* a, int len){

int i;

for (i = len-1; i >=0 ; i--)

printf("%d ", a[i]);

}

void createHeap(struct MinHeap\* minHeap) {

int i = 0;

for(i = (minHeap->size) / 2; i >= 0; i--) {

heapify(minHeap, i);

}

}

void heapify(struct MinHeap\* minHeap, int idx) {

int smallest = idx;

int l = 2\*idx + 1; //std::cout << "left = " << l << std::endl;

int r = 2\*idx + 2; //std::cout << "right = " << r << std::endl;

if((l < minHeap->size) && (minHeap->array[l] < minHeap->array[idx])) {

smallest = l;

}

if((r < minHeap->size) && (minHeap->array[r] < minHeap->array[smallest])) {

smallest = r;

}

//std::cout << "smallest = " << smallest << std::endl;

if(smallest != idx) {

swap(&minHeap->array[idx], &minHeap->array[smallest]);

heapify(minHeap, smallest);

}

}

Input

6

5 30 15 26 96 47

Output

5 15 26 30 47 96

# References

<https://www.geeksforgeeks.org/heap-data-structure/>