



Department of IT and Computer Science
Pak-Austria Fachhochschule: Institute of Applied Sciences
and Technology, Haripur, Pakistan

COMP-201L Data Structures and Algorithms Lab

Lab Report 11

Class: **Computer Science**
Name: **Yaseen Ejaz Ahmed**
Registration No.: **B20F0283CS014**
Semester: **Third**
Submitted to: **Engr. Rafi Ullah**

Instructor Signature

Lab No. 11

Heap implementation and operations

Objectives:

- Heap implementation in C++
- Various operation of heap using C++

Tools/Software Required:

C++ Compiler

Introduction:

The efficiency of the heap data structure provides itself to a surprisingly simple and very efficient sorting algorithm called heapsort. The heap sort algorithm uses a heap tree to sort an array either in ascending or descending order. It consists of two phases:

- Using unsorted data array, build the heap by repeatedly inserting each element into the heap.
- Remove the top element (item at the root) from the heap and place it in the last location of the array. Rebuild the heap from the remaining elements of the array. This process is repeated until all items from the heap are deleted. Finally, the array is in sorted order.

A heap is a complete binary tree such that the root is the largest item (max heap). The ordering in a heap is top-down, but not left-to-right. Each root is greater than or equal to each of its children, but some left siblings may be greater than their right siblings and some may be less. Since heaps are typically stored as arrays, we can apply the heap operations to an array of integers. We illustrate the operations as though they are being performed on binary trees, while they are really defined for the arrays that represent them by natural mapping.

Lab Tasks:

Lab Task 01: The member function `removemin` should be replaced by a new function called `removemax`.

Code:

```
#include <iostream>
using namespace std;

void heapify(int arr[], int n, int i)
{
    int largest = i;           // largest as root
    int l = 2 * i + 1;         //2*i + 1
    int r = 2 * i + 2;         //2*i + 2

    if (l < n && arr[l] > arr[largest])
        largest = l;

    if (r < n && arr[r] > arr[largest])
        largest = r;

    if (largest != i) {
        swap(arr[i], arr[largest]);

        heapify(arr, n, largest);
    }
}

void removemax(int arr[], int& n)
{
    int lastElement = arr[n - 1];

    arr[0] = lastElement;

    n = n - 1;

    heapify(arr, n, 0);
}

void printArray(int arr[], int n)
{
    for (int i = 0; i < n; ++i)
        cout << arr[i] << " ";

    cout << "\n";
}
```

```
int main()
{
    int arr[] = { 9, 6, 2, 3, 5 };

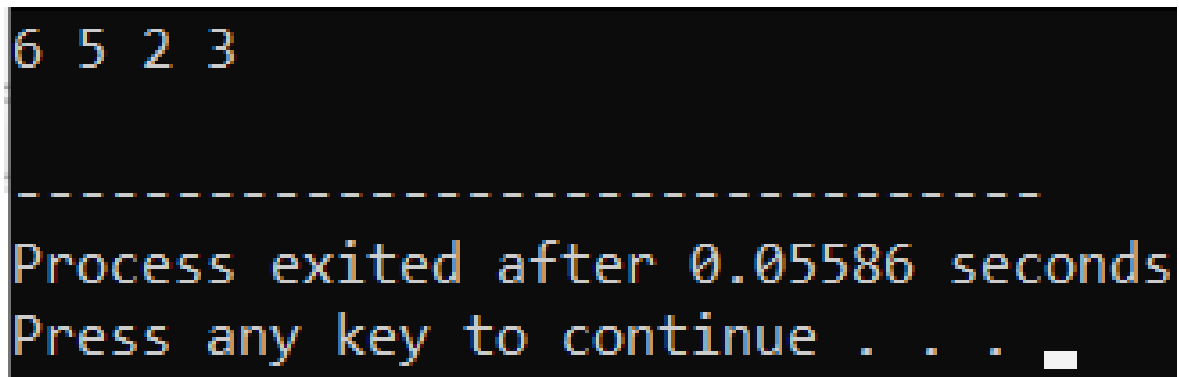
    int n = sizeof(arr) / sizeof(arr[0]);

    removemax(arr, n);

    printArray(arr, n);

    return 0;
}
```

Output:



```
6 5 2 3
-----
Process exited after 0.05586 seconds
Press any key to continue . . .
```

Results & Observations:

In this lab we have learnt about the basic concepts and operations of a max and min heap. Min heap has the lowest and max heap has the highest root. With each insertion of a new data, the heap structure shall change to make the roots smaller than the childs.