# Department of Computing

# School of Electrical Engineering and Computer Science

**CS-250: Data Structure and Algorithms**

# Class: BESE 13A

# Lab 11: Implementation of Binary Heap (min heap)

**Date: 8th December, 2023**

**Time: 10 am - 1 pm**

**Name:** Aimen Munawar

**Class:** BESE-13-A

**CMS ID:** 415867

**Lab Engineer: Anum Asif**

# 

# Lab 11: Implementation of Binary Heap (min heap)

**Introduction**

This lab is based on the Binary Heap (min heap).

**Objectives**

The objective of this lab is to implement binary heap (min heap).

**Tools/Software Requirement**

Visual Studio C++

**Helping Material**

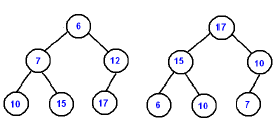
Lecture slides, text book

**Description**

A binary heap is a complete binary tree which satisfies the heap ordering property. The ordering can be one of two types:

 the *min-heap property*: the value of each node is greater than or equal to the value of its parent, with the minimum-value element at the root.

 the *max-heap property*: the value of each node is less than or equal to the value of its parent, with the maximum-value element at the root.

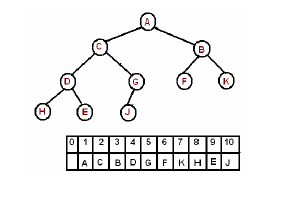


In a heap the highest (or lowest) priority element is always stored at the root, hence the name "heap". A heap is not a sorted structure and can be regarded as partially ordered.

A heap is a useful data structure when you need to remove the object with the highest (or lowest) priority. A common use of a heap is to implement a priority queue.

**Array Implementation**

A complete binary tree can be uniquely represented by storing its level order traversal in an array.



The root is the second item in the array. We skip the index zero cell of the array for the convenience of implementation. Consider k-th element of the array,

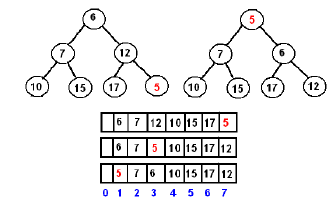
its left child is located at 2\*k index

its right child is located at 2\*k+1. index

its parent is located at k/2 index

**Insert**

The new element is initially appended to the end of the heap (as the last element of the array). The heap property is repaired by comparing the added element with its parent and moving the added element up a level (swapping positions with the parent).



**Delete Min**

The minimum element can be found at the root, which is the first element of the array. We remove the root and replace it with the last element of the heap and then restore the heap property

**Lab Tasks**

Implement a binary heap using an array & implement the following functions.

**Top** - returns min element without removing it from the heap

**Pop** - Make sure that the tree remains a complete binary tree.

**Push** – Insert a number into the heap and make sure the heap maintains its key property

**isEmpty** – return TRUE if the heap is empty

**size** - returns the number of elements in the heap

**height** - returns the height of the tree

**buildHeap(array)** - converts any array of numbers into a heap.

**print** - prints the tree

**Important Note:** Practice your knowledge of OOP with C++ when creating a solution. Remember to comment your code properly. Inappropriate or no comment may result in deduction of marks.

**Solution**

|  |
| --- |
| Solution |
| Lab Task Screen Shots (all functions):                  Lab Task Code:  #include <iostream>  #include <cmath>  using namespace std;  // Function to swap two values  void swap(int &a, int &b)  {  int temp = a;  a = b;  b = temp;  }  // Class definition for Min Heap  class Heap  {  public:  int values[30]; // Array to store heap elements  int size; // Current size of the heap  public:  // Constructor to initialize an empty heap  Heap()  {  size = 0;  }  // Check if the heap is empty  bool isEmpty()  {  return size == 0;  }  // Check if the heap is full  bool isFull()  {  return size == 30;  }  // Get the top element of the heap (minimum element in a Min Heap)  int top()  {  if (isEmpty())  {  cout << "Heap is empty!\n";  return -1;  }  else  {  return values[1];  }  }  // Insert a new element into the heap  void push(int value)  {  if (isFull())  {  cout << "Heap is full!\n";  return;  }  // Increment the size and add the new element to the end  size++;  values[size] = value;  // Move the element up to its correct position to maintain the heap property  int current = size;  while (current > 1 && values[current] < values[current / 2])  {  swap(values[current], values[current / 2]);  current = current / 2;  }  }  // Fix the heap property starting from a given index  void heapify(int index)  {  int left = 2 \* index;  int right = 2 \* index + 1;  int smallest = index;  // Check if the left child is smaller than the current smallest  if (left <= size && values[left] < values[index])  smallest = left;  // Check if the right child is smaller than the current smallest  if (right <= size && values[right] < values[smallest])  smallest = right;  // If the smallest is not the current index, swap and recursively heapify  if (smallest != index)  {  swap(values[index], values[smallest]);  heapify(smallest);  }  }  // Build a heap from the existing elements  void buildheap()  {  // Start from the last non-leaf node and heapify each node  for (int i = size / 2; i >= 1; i--)  {  heapify(i);  }  }  // Build a heap from an array of elements  void buildheapfromarray(int \*arr, int size)  {  // Push each element into the heap and then build the heap  for (int i = 0; i < size; i++)  push(arr[i]);  buildheap();  }  // Delete the minimum element (top element) from the heap  void deleteMin()  {  if (isEmpty())  {  cout << "Heap is empty!\n";  return;  }  // Replace the top element with the last element and heapify  values[1] = values[size];  size--;  heapify(1);  }  // Pop an element from a specific index  int pop(int index)  {  if (index < 1 || index > size)  {  cout << "Invalid index!\n";  return -1;  }  else  {  // Remove the element at the specified index and heapify  int poppedValue = values[index];  values[index] = values[size];  size--;  heapify(index);  return poppedValue;  }  }  // Display the elements of the heap  void print()  {  if (isEmpty())  {  cout << "Heap is empty!\n";  return;  }  for (int i = 1; i <= size; i++)  {  cout << values[i] << " ";  if (i == 1 || i == 3 || i == 7 || i == 15)  cout << endl;  }  cout << endl;  }  // Get the current size of the heap  int getSize()  {  return size;  }  // Get the height of the heap  int getHeight()  {  return log2(size) + 1;  }  };  int main()  {  Heap myHeap = Heap();  const int arraySize = 15;  int array[arraySize];  for (int i = 0; i < arraySize; i++)  array[i] = (rand() % 100) + 1;  myHeap.buildheapfromarray(array, arraySize);  // Menu for all functions  int choice;  do  {  // Display menu options  cout << "1. Top\n2. Pop\n3. Push\n4. Is Empty\n5. Size\n6. Height\n7. Build Heap from Array\n8. Print\n0. Exit\n";  cout << "Enter your choice: ";  cin >> choice;  switch (choice)  {  case 1:  cout << "Top element: " << myHeap.top() << endl;  break;  case 2:  cout << "Enter index to pop: ";  int index;  cin >> index;  cout << "Popped element: " << myHeap.pop(index) << endl;  break;  case 3:  cout << "Enter element to push: ";  int element;  cin >> element;  myHeap.push(element);  cout << "Element pushed.\n";  break;  case 4:  cout << (myHeap.isEmpty() ? "Heap is empty.\n" : "Heap is not empty.\n");  break;  case 5:  cout << "Size of the heap: " << myHeap.getSize() << endl;  break;  case 6:  cout << "Height of the heap: " << myHeap.getHeight() << endl;  break;  case 7:  {  cout << "Enter size of the array: ";  int arrSize;  cin >> arrSize;  int arr[arrSize];  cout << "Enter elements of the array:\n";  for (int i = 0; i < arrSize; i++)  cin >> arr[i];  myHeap.buildheapfromarray(arr, arrSize);  cout << "Heap built from array.\n";  break;  }  case 8:  {  cout << "Heap elements:\n";  myHeap.print();  break;  }  case 0:  cout << "Exiting...\n";  break;  default:  cout << "Invalid choice!\n";  }  } while (choice != 0);  return 0;  } |

### Deliverables

Compile a single word document by filling in the solution part and submit this Word file on LMS. You must show the implementation of the tasks in the designing tool, along with your complete Word document to get your work graded. You must also submit this Word document on the LMS.