Saad Saiyed | 200415258

Assignment 3: CS 340

Problem 1 (3+3+1 marks).

(a) Show the result of inserting 10, 12, 1, 14, 6, 5, 8, 15, 3, 9, 7, 4, 11, 13, and 2, one at a time, into an initially empty binary heap. Additionally, show the intermediate result after every single insertion. (This is an extension of textbook problem 6.2(a).)

(b) Show the result of using the linear-time algorithm to build a binary heap using the same input. (This is textbook problem 6.2(b).)

(c) Show the results of two consecutive deleteMin operations on the 3-heap in Figure 6.19 in the textbook.

Ans: You can see the attachment under the name '*Assignment 3 Q1.pdf*.

Problem 2 (2+3 marks).

(a) Show that, in any heap, the maximum item must be at one of the leaves. (This is textbook problem 6.8(a).)

(b) Show that any binary heap contains exactly ⌈N/2⌉ leaves. (This is textbook problem 6.8(b).)

Ans:

a) For minHeap we have all parent nodes as the minimum of their child nodes.

Considering this, root node's value will be the smallest in the whole tree. If we have only one node which is root at level 0 and that's it, we say the statement verifies the case. For a tree which has more than one node i.e. tree with level greater than 0, each child will be greater than its parent which follows the statement.

Now for maxHeap we have all parent nodes as the maximum of their child nodes.

Considering this, root node's value will be the largest in the whole tree and this proves the statement for only one node in a tree. If we have more than one node i.e. we have at least level 1 in a tree, the nodes at level 1 will be larger than level 2 or its child elements as this is a maxHeap. And that follows the statement as it is the largest value as a child of the root node for the respective tree.

Problem 3 (4+2+2 marks).

(a) Implement the following algorithm, which is given a duplicate-free array array as input, in C++.

whatDoIDo (array):

1) Build a heap from an array (using buildHeap as explained in class), where the heap starts at position array[0].

2) Starting from j = size of array - 1, as long as j>0:

i. Swap the entries array[0] and array[j].

ii. Percolate down array[0], but only within the subarray array[0..j-1].

iii. Decrement j by 1.

Provide three input/output examples for duplicate-free arrays of size about 10.

(b) What does whatDoIDo do? Explain your answer.

(c) What is the worst-case running time of whatDoIDo, in dependence of the size N of the given array? Explain your Answer.

Ans: You can find the Visual Studio Project of this program in the attachment under the folder name '*Assignment 3 Q3*

```
/*
* CS 340 Assignment 3 Q3
*
* Compiled with Visual Studio
*/

// A)
#include <iostream>

using namespace std;

void maxHeapify(int* array, int i, int n);
void buildHeap(int* array);
void print_array(int* a);
void whatDoIDo(int* array);

const int arrSize = 10;

int main()
{
    int a[arrSize];
    int b[arrSize];
    int c[arrSize];

    cout << "Give 10 numbers with space in between a):";
    int i = 0;
    while (i < 10)
    {
        cin >> a[i++];
    }

    cout << "Give 10 numbers with space in between b):";
    i = 0;
    while (i < 10)
    {
        cin >> b[i++];
    }

    cout << "Give 10 numbers with space in between c):";
    i = 0;
```

```
    while (i < 10)
    {
        cin >> c[i++];
    }

    whatDoIDo(a);
    whatDoIDo(b);
    whatDoIDo(c);

    return 0;
}

void maxHeapify(int* array, int i, int n)
{
    int value = array[i];
    int j = 2 * i;

    while (j <= n)
    {
        if (array[j + 1] > array[j] && j < n) j = j + 1;
        if (value > array[j]) break;
        else if (value <= array[j])
        {
            array[j / 2] = array[j];
            j = 2 * j;
        }
    }
    array[j / 2] = value;
}
void buildHeap(int* array)
{
    for (int i = arrSize / 2; i >= 1; i--)
    {
        maxHeapify(array, i, arrSize - 1);
    }
}
void print_array(int* array)
{
    cout << "------START------\n  {";
```

```
    int i = 0;
    while (i < arrSize)
    {
        cout << *(array + i++);
        if (i < arrSize) cout << ", ";
    }

    cout << "}\n-------END-------\n" << endl;
}
/*
Input/Output for duplicate-free arrays of size about 10.

    Give 10 numbers with space in between a):96 22 12 36 8 93 17 13 11 68
    Give 10 numbers with space in between b):99 96 100 5 85 49 46 15 29 74
    Give 10 numbers with space in between c):80 71 61 51 78 86 19 10 33 41
    ------START------
      {93, 68, 36, 22, 12, 17, 13, 11, 8, 96}
    -------END-------

    ------START------
      {100, 96, 46, 85, 49, 5, 15, 29, 74, 99}
    -------END-------

    ------START------
      {86, 78, 51, 71, 61, 19, 10, 33, 41, 80}
    -------END-------
*/


/*
* B)
* Firstly we call 'buildHeap()' function to create an array in heap space.
* And then we start a loop from j = size of array - 1 as long as j > 0.
* Inside loop we do following things
*   i.      Swap the entries array[0] and array[j].
*   ii.     Percolate down array[0].
*   iii.    Decrement j by 1.
*
* Explanation:
*   Basically we go in the process of buildHeap we perform maxHeapify
```

```
*   which is a method that converts a binary tree where every node has
*   child less than or equal to itself. After that we perform swapping
*   which is pretty basic. And at last we print the resulting array.
*/

void whatDoIDo(int* array)
{
    buildHeap(array);

    int j = arrSize - 1;
    int tempArrSwapper;
    while (j > 0)
    {
        tempArrSwapper = array[0];
        array[0] = array[j];
        array[j--] = tempArrSwapper;
    }

    print_array(array);
}



/*
* C)
* Worst case run time is O(n^2 log(n))
*/
```