CSCI 255: Lab #8 Binary Heaps

Darwin Jacob Groskleg

Printed: Friday, March 13th, 2020

Contents

Contents i

Questions & Console Output 1

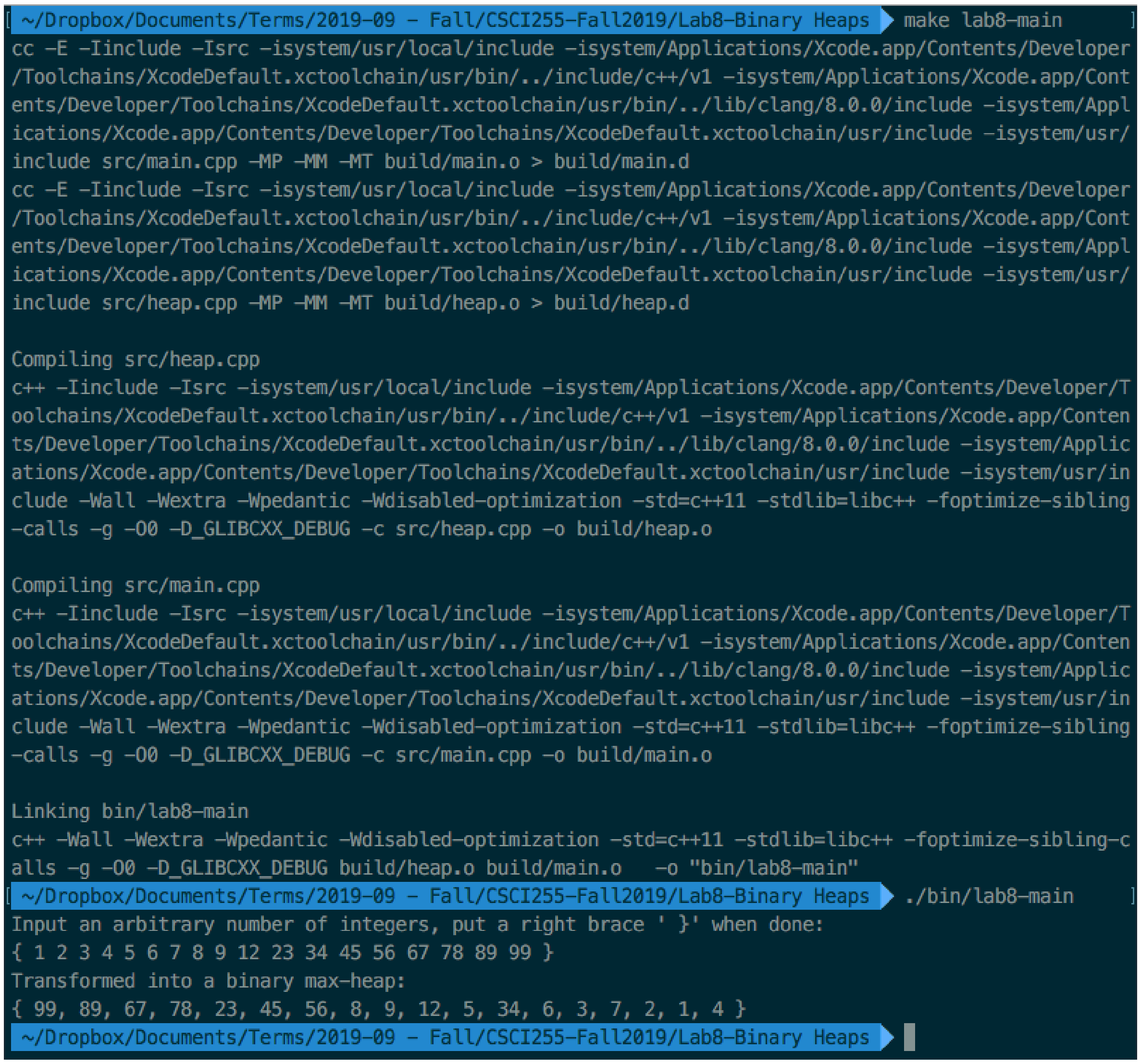
src/main.cpp 2

include/heap.hpp 3

src/heap.cpp 4

include/BinaryHeap.hpp 6

Questions & Console Output



```
[ ~/Dropbox/Documents/Terms/2019-09 - Fall/CSCI255-Fall2019/Lab8-Binary Heaps  make lab8-main      ]
cc -E -Iinclude -Isrc -isystem/usr/local/include -isystem/Applications/Xcode.app/Contents/Developer
/Toolchains/XcodeDefault.xctoolchain/usr/bin/../include/c++/v1 -isystem/Applications/Xcode.app/Cont
ents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/../lib/clang/8.0.0/include -isystem/Appl
ications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/include -isystem/usr/
include src/main.cpp -MP -MM -MT build/main.o > build/main.d
cc -E -Iinclude -Isrc -isystem/usr/local/include -isystem/Applications/Xcode.app/Contents/Developer
/Toolchains/XcodeDefault.xctoolchain/usr/bin/../include/c++/v1 -isystem/Applications/Xcode.app/Cont
ents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/../lib/clang/8.0.0/include -isystem/Appl
ications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/include -isystem/usr/
include src/heap.cpp -MP -MM -MT build/heap.o > build/heap.d

Compiling src/heap.cpp
c++ -Iinclude -Isrc -isystem/usr/local/include -isystem/Applications/Xcode.app/Contents/Developer/T
oolchains/XcodeDefault.xctoolchain/usr/bin/../include/c++/v1 -isystem/Applications/Xcode.app/Conten
ts/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/../lib/clang/8.0.0/include -isystem/Applic
ations/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/include -isystem/usr/in
clude -Wall -Wextra -Wpedantic -Wdisabled-optimization -std=c++11 -stdlib=libc++ -foptimize-sibling
-calls -g -O0 -D_GLIBCXX_DEBUG -c src/heap.cpp -o build/heap.o

Compiling src/main.cpp
c++ -Iinclude -Isrc -isystem/usr/local/include -isystem/Applications/Xcode.app/Contents/Developer/T
oolchains/XcodeDefault.xctoolchain/usr/bin/../include/c++/v1 -isystem/Applications/Xcode.app/Conten
ts/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/../lib/clang/8.0.0/include -isystem/Applic
ations/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/include -isystem/usr/in
clude -Wall -Wextra -Wpedantic -Wdisabled-optimization -std=c++11 -stdlib=libc++ -foptimize-sibling
-calls -g -O0 -D_GLIBCXX_DEBUG -c src/main.cpp -o build/main.o

Linking bin/lab8-main
c++ -Wall -Wextra -Wpedantic -Wdisabled-optimization -std=c++11 -stdlib=libc++ -foptimize-sibling-c
alls -g -O0 -D_GLIBCXX_DEBUG build/heap.o build/main.o   -o "bin/lab8-main"
[ ~/Dropbox/Documents/Terms/2019-09 - Fall/CSCI255-Fall2019/Lab8-Binary Heaps  ./bin/lab8-main     ]
Input an arbitrary number of integers, put a right brace ' }' when done:
{ 1 2 3 4 5 6 7 8 9 12 23 34 45 56 67 78 89 99 }
Transformed into a binary max-heap:
{ 99, 89, 67, 78, 23, 45, 56, 8, 9, 12, 5, 34, 6, 3, 7, 2, 1, 4 }
 ~/Dropbox/Documents/Terms/2019-09 - Fall/CSCI255-Fall2019/Lab8-Binary Heaps 
```

Figure 1: Compiles and Runs.

src/main.cpp

```cpp
/* main.cpp
 * ---------
 * Course:  CSCI 255
 * Lab:     # 8 Binary Heap
 * Authors: Darwin Jacob Groskleg
 *
 * Purpose: Tests the MakeHeap and Heapify functions by asking the user to input
 *          the integers of an arbitrary array, calling MakeHeap on the array
 *          and outputting the array.
 *
 *          This main function needed to be separated from the heap code in
 *          order to be able to call and test the heap code from multiple files.
 *
 * Console Sample:
 *      Input an arbitrary number of integers, put a right brace '}' when done:
 *      { 1 2 3 4 5 6 7 8 9 12 23 34 45 56 67 78 89 99 }
 *      Transformed into a binary max-heap:
 *      { 99, 89, 67, 78, 23, 45, 56, 8, 9, 12, 5, 34, 6, 3, 7, 2, 1, 4 }
 */
#include <iostream>
#include <vector>

#include "heap.hpp"

int main() {
    std::cout << "Input an arbitrary number of integers, "
              << "put a right brace ' }' when done:\n"
              << "{ ";

    std::vector<int> arbitrary_array;
    int number_in;  // Will loop until it does not give an int
    while (std::cin >> number_in)
        arbitrary_array.push_back(number_in);

    MakeHeap(arbitrary_array.data(), arbitrary_array.size());

    auto comma = [&arbitrary_array] (int element) {
        return (element == arbitrary_array.back()) ? " " : ",";
    };
    std::cout << "Transformed into a binary max-heap:\n{";
    for (int el : arbitrary_array)
        std::cout << ' ' << el << comma(el);
    std::cout << "}\n";

    return 0;
}
```

include/heap.hpp

```
/* heap.hpp
 * ---------
 * Course:  CSCI 255
 * Lab:     # 8 Binary Heap
 * Authors: Darwin Jacob Groskleg
 *
 * Purpose: Interface to the c-style imperative/procedural implementations of
 *          MakeHeap and Heapify.
 */
#ifndef HEAP_HPP_INCLUDED
#define HEAP_HPP_INCLUDED

/// MakeHeap
///
/// Creates an integer max-heap from a given set of numbers.
///
///
void MakeHeap(int [], const int);


/// Heapify
///
/// Ensures the heap property holds for given node.
///
void Heapify(int [], const int , int);


/// parent_node
///
/// Given a node position in an array returns the position of its parent.
///
inline int parent_node(int node_position) {
    return node_position / 2;
}

#endif // HEAP_HPP_INCLUDED
```

src/heap.cpp

```
/* heap.cpp
 * ---------
 * Course:  CSCI 255
 * Lab:     # 8 Binary Heap
 * Authors: Darwin Jacob Groskleg
 *
 * Purpose: To implement the MakeHeap and Heapify functions described in the
 *          lecture slides for a max-heap. Then test it by asking the user to
 *          input the integers of an arbitrary array, calling MakeHeap on the
 *          array and outputting the array.
 *
 * NOTICE: the procedural/imperative approach of passing c-arrays to MakeHeap is
 *         very unsafe. Use safer container types like std::array.
 */
#include "heap.hpp"

#include <algorithm>
#include <vector>

/// MakeHeap
///
/// Transforms a partially filled array of unordered integers into a binary
/// max-heap. Needs a given heapsize to determine by how full the array is.
///
/// Complexity:
///     T(n) = O(n)
///
void MakeHeap(int heap_array[], const int heapsize) {
    // Start with the parent of the last node then iterate to the first.
    int last_node = heapsize - 1;
    for (int i = parent_node(last_node); i>=0; i--)
        Heapify(heap_array, heapsize, i);
}


/// Heapify
///
/// Given a heap node that potentially violates the max-heap property,
/// down-rotate the node until the property is resolved.
///
/// Max-Heap Property:
///  0. The largest element is stored at the root.
///  1. For every node in the heap,
///      its value is greater or equal to that of its children.
///
/// Complexity:
///     T(n) = O(lg n) on a subtree of size n.
///
void Heapify(int heap_array[], const int heap_size, int node_position) {
    while (node_position < heap_size) {
        // Compute in one instruction with a binary shift (Cormen, p152).
        int left_i  = 2 * node_position + 1;
        int right_i = 2 * node_position + 2;
        int max_i = node_position;

        auto child_is_bigger = [heap_array, heap_size] (int child, int other) {
            return (child             < heap_size &&
                    heap_array[other] < std::max(heap_array[other],
                                                 heap_array[child]) );
        };

        if (child_is_bigger(left_i,  max_i))    max_i = left_i;
        if (child_is_bigger(right_i, max_i))    max_i = right_i;
        if (max_i == node_position)             break;

        std::swap(heap_array[node_position], heap_array[max_i]);
        node_position = max_i;
    }
}
```

include/BinaryHeap.hpp

```cpp
/* BinaryHeap.hpp
 * --------------
 * Course:  CSCI 255
 * Lab:     # 8 Binary Heap
 * Authors: Darwin Jacob Groskleg
 *
 * Purpose: Provide an interface for the heap code that is separate from main.
 *          This is an object oriented implentation of a max binary heap.
 *
 *          THIS IS NOT REQUIRED CODE BUT INCLUDED OUT OF INTEREST!
 */
#ifndef BINARYHEAP_HPP_INCLUDED
#define BINARYHEAP_HPP_INCLUDED

#include <algorithm>
#include <vector>

#include "PrintableTree.hpp"

//// Max Heap
class BinaryHeap {
    std::vector<int> heap_array;
    int              heap_size;

public:
    /// HeapSort
    /// Sort by removing elements from a heap until it is empty.
    ///
    /// Complexity:
    ///     T(n) = O(n log n) where n is the number of elements in the array.
    static auto heap_sort() -> std::vector<int>;

    /// MakeHeap
    /// XXX returns an owning pointer to a heapified array.
    ///
    /// Complexity:
    ///     T(n) = O(n) where n is the number of elements in the array.
    static BinaryHeap make_heap(int heap_array[], const int heap_size) {
        return BinaryHeap(heap_size, heap_array);
    }

    /// Ctor must be strictest to avoid undefined behaviour while other make
    /// functions may be more permissive and adaptive.
    ///
    /// Validate: ...use vector and defer handing these problems.
    ///     - heap_size_ is not larger than the array_size
    ///     - heap_array_ is not too big VS. array_size, which should throw
    ///       based on aggregate initialization rules.
    BinaryHeap(const int heap_size_, int heap_array_[]) :
        heap_array{ heap_size_ * 2, *heap_array_ },
        heap_size{ heap_size_ }
    {}

   ~BinaryHeap();
```

```cpp

    /// Insert
    ///
    /// Complexity:
    ///     T(n) = O(lg n)
    void insert(int element) {
        if (heap_size == static_cast<int>(heap_array.size()))
            heap_array.push_back(element);
        else // heap_size is less
            heap_array.at(heap_size);
        heap_size++;
//          increase_key(i, key);
    }

    /// ExtractMax
    /// Removes the max value from the heap and returs it.
    /// The memory is not deallocated just ignored.
    ///
    /// Complexity:
    ///     T(n) = O(lg n)
    int extract_max() {
        int max{ heap_array.at(root) };

        heap_array.at(root) = heap_array.at(--heap_size);
        heapify(root);

        return max;
    }

    /// Using DFS will produce an object that can readily be used to represent
    /// the state of the heap visually as a tree.
    //
    /// Complexity:
    ///     T(n) = O(n)
    auto export_state() const -> PrintableTree {
        return PrintableTree(); // not included in project
    }

private:
    const int root = 0;

    /// XXX GUARD node is not negative
    int parent(int node_position) { return (node_position - 1) / 2; }
    int   left(int node_position) { return 2 * node_position + 1; }
    int  right(int node_position) { return 2 * node_position + 2; }

    /// IncreaseKey

    /// Heapify(node);
    void heapify(const int node);
};

#endif // BINARYHEAP_HPP_INCLUDED
```