Eric Blasko

CSE 330 Data Structures

Winter 2018

Lab 10 – Priority queue

* **Status**

100% complete

* **Time Complexity**

The following methods are O(1) as they are constant in time

* bool empty() { return c.empty(); }
* unsigned int size() { return c.size(); }
* const T & top() const { return c.front(); }

The following methods are O(logn) were n is the size of the vector. When vector is viewed as a tree, only one branch is ever traversed during push or pop

* void push(const T & x);
* void push(T && x);
* void pop();
* **Source Code**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Eric Blasko

\* Priority\_queue.h

\* 03/19/2018

\* This class mimics the priority\_queue found in STL. Using a vector, this data structure

\* maintains its largest value in the front of the vector. When a element is added, it

\* is added to the end of the vector and swapped with its parent if bigger in size. When

\* elements are removed, first value in vector is popped, and vector is reorganized to

\* ensure biggest value is in the front

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#ifndef PRIORITY\_QUEUE\_H

#define PRIORITY\_QUEUE\_H

// Priority\_queue.h

#include <vector>

using namespace std;

template <class T>

class Priority\_queue {

public:

//returns bool if vector is empty

bool empty() { return c.empty(); }

//returns size of vector

unsigned int size() { return c.size(); }

//add element to vector

void push(const T & x);

//add element to vector

void push(T && x);

//remove first value from vector

void pop();

//returns first value in vector

const T & top() const { return c.front(); }

private:

vector<T> c;

};

//push element to back of vector. If its parent is larger, swap until its in correct

//location. copy method

template <class T>

void Priority\_queue<T>::push(const T & x)

{

c.push\_back(x);

unsigned n = c.size() - 1;

unsigned parent = (n - 1) / 2;

while (n > 0 && c[parent] < c[n])

{

swap(c[parent], c[n]);

n = parent;

parent = (n - 1) / 2;

}

}

//push element to back of vector. If its parent is larger, swap until its in correct

//location.move method

template <class T>

void Priority\_queue<T>::push(T && x)

{

c.push\_back(move(x));

unsigned n = c.size() - 1;

unsigned parent = (n - 1) / 2;

while (n > 0 && c[parent] < c[n])

{

swap(c[parent], c[n]);

n = parent;

parent = (n - 1) / 2;

}

}

//move last element to the front. pop the last value. then loop to see if first element

//is larger than its children and swap if needed. if element is in correct location,

//break

template <class T>

void Priority\_queue<T>::pop()

{

if (c.empty()) return;

c[0] = c.back();

c.pop\_back();

unsigned n = 0;

unsigned left = 1;

unsigned right = 2;

while (left < c.size())

{

unsigned biggest = left;

if (right < c.size() && c[left] < c[right])

biggest = right;

if (c[n] < c[biggest])

{

swap(c[n], c[biggest]);

n = biggest;

left = 2 \* n + 1;

right = 2 \* n + 2;

}

else

break;

}

}

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Eric Blasko

\* Priority\_queue\_test1.cpp

\* 03/19/2018

\* This program test the methods found in Priority\_queue.h. The program will ask for the user to input

\* integers in any order, then copy its values to another instance of Priority\_queue, and then display and

\* pop its first value. Highest values in the vector should always be popped first

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include "Priority\_queue.h"

using namespace std;

//Main function to test methods in Priority\_queue.h

int main()

{

Priority\_queue<int> p, q, pq;

int x;

while (cin >> x)

p.push(x);

pq = q = p;

while (!pq.empty()) {

cout << pq.top() << " ";

pq.pop();

}

cout << endl;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Eric Blasko

\* Priority\_queue\_test1.cpp

\* 03/19/2018

\* This program test the methods found in Priority\_queue.h. After each push and pop, assert will verify

\* the top element or the size of the queue to ensure that the methods in Priority\_queue.h are

\* performing as intended.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <cassert>

#include "Priority\_queue.h"

using namespace std;

//Main function to test methods in Priority\_queue.h

int main()

{

Priority\_queue<int> pq;

assert(pq.size() == 0);

assert(pq.empty());

pq.push(10);

assert(pq.top() == 10);

pq.push(20);

assert(pq.top() == 20);

pq.push(30);

assert(pq.top() == 30);

pq.push(40);

assert(pq.top() == 40);

pq.push(50);

assert(pq.top() == 50);

pq.push(5);

assert(pq.top() == 50);

pq.pop();

assert(pq.top() == 40);

pq.pop();

assert(pq.top() == 30);

pq.pop();

assert(pq.top() == 20);

pq.pop();

assert(pq.top() == 10);

pq.pop();

assert(pq.top() == 5);

pq.pop();

assert(pq.size() == 0);

Priority\_queue<int> pq2;

pq2.push(30);

pq2.push(11);

pq2.push(7);

pq2.pop();

assert(pq2.top() == 11);

pq2.pop();

assert(pq2.top() == 7);

pq2.pop();

assert(pq2.empty());

cout << "All tests passed." << endl;

}

* **Sample Runs**

**Test 1**

Script started on 2018-03-19 08:14:03-0700

^[]0;005670557@csusb.edu@jb358-1:~/cse330/lab10^G[005670557@csusb.edu@jb358-1 lab10]$ g\_\_^H^[[K^H^[[K++ 0c ^H^[[K^H^[[K^H^[[K-c Priority\_queue\_test1.cpp^M

^[]0;005670557@csusb.edu@jb358-1:~/cse330/lab10^G[005670557@csusb.edu@jb358-1 lab10]$ g++ Priority\_queue\_test1.o^M

^[]0;005670557@csusb.edu@jb358-1:~/cse330/lab10^G[005670557@csusb.edu@jb358-1 lab10]$ ./a.out^M

1^M

2^M

3^M

4^M

5^M

4^M

3^M

2^M

1^M

5 4 4 3 3 2 2 1 1 ^M

^[]0;005670557@csusb.edu@jb358-1:~/cse330/lab10^G[005670557@csusb.edu@jb358-1 lab10]$ exit^M

Script done on 2018-03-19 08:14:44-0700

**Test 2**

Script started on 2018-03-19 08:14:54-0700

^[]0;005670557@csusb.edu@jb358-1:~/cse330/lab10^G[005670557@csusb.edu@jb358-1 lab10]$ g++ -c Priority\_queue\_test2.t^H^[[Kcpp^M

^[]0;005670557@csusb.edu@jb358-1:~/cse330/lab10^G[005670557@csusb.edu@jb358-1 lab10]$ g++ Priority\_queue\_test2.o^M

^[]0;005670557@csusb.edu@jb358-1:~/cse330/lab10^G[005670557@csusb.edu@jb358-1 lab10]$ a^H^[[K./a.out^M

All tests passed.^M

^[]0;005670557@csusb.edu@jb358-1:~/cse330/lab10^G[005670557@csusb.edu@jb358-1 lab10]$ ^C^M

^[]0;005670557@csusb.edu@jb358-1:~/cse330/lab10^G[005670557@csusb.edu@jb358-1 lab10]$ exit^M

Script done on 2018-03-19 08:15:22-0700