Friday, June 10, 2011 SourceTricks (http://www.blogger.com/profile/18153422615549744777) Data Structures (http://www.sourcetricks.com/search/label/Data%20Structures)
6 comments (http://www.sourcetricks.com/2011/06/c-heaps.html#comment-form)

C++ Heaps (http://login2win.blogspot.com/heaps.html)

What is a heap data structure? How to implement in C++?

This article explains the heap data structure and provides a sample implementation using C++.

- Heap is a binary tree that stores priorities (or priority-element pairs) at the nodes.
- It has the following properties:
  - All levels except last level are full. Last level is left filled.
  - Priority of a node is at-least as large as that of its parent (min-heap) (or) vice-versa (max-heap).
  - If the smallest element is in the root node, it results in a min-heap.
  - If the largest element is in the root node, it results in a max-heap.
  - A heap can be thought of as a priority queue. The most important node will always be at the root of the tree.
  - Heaps can also be used to sort data, heap sort.
  - The two most interesting operations in a heap is heapifyup and heapifydown.
    - Heapify-up (assumption min-heap)
      - Used to add a node to the heap. To add a node, it is inserted at the last empty space and heapifyup process is done.
      - When a node is added, its key is compared to its parent. If parent key is smaller than the current node it is swapped. The process is repeated till the heap property is met.
    - Heapify-down
      - Used during removal of a node. When a node is removed which is always the root (lowest in priority) the last available node in heap is replaced as the root and heapifydown process is done.
      - The key of parent node is compared with the children. If any of the children have lower priority it is swapped with the parent. The process is repeated for the newly swapped node till the heap property is met.

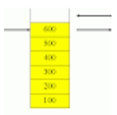
Implementation of a heap in C++. Demonstrates min-heap using arrays.

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

class Heap {
public:
    Heap();
    ~Heap();
    void insert(int element);
    int deletemin();
    void print();
    int size() { return heap.size(); }
private:
    int left(int parent);
    int right(int parent);
    int parent(int child);
    void heapifyup(int index);
    void heapifydown(int index);
private:
    vector<int> heap;
};

Heap::Heap()
{
}

Heap::~Heap()
{
}

void Heap::insert(int element)
{
    heap.push_back(element);
    heapifyup(heap.size() - 1);
}

int Heap::deletemin()
{
    int min = heap.front();
    heap[0] = heap.at(heap.size() - 1);
    heap.pop_back();
    heapifydown(0);
    return min;
}

void Heap::print()
{
    vector<int>::iterator pos = heap.begin();
    cout << "Heap = ";
    while ( pos != heap.end() ) {
        cout << *pos << " ";
        ++pos;
    }
    cout << endl;
}

void Heap::heapifyup(int index)
{
    //cout << "index=" << index << endl;
    //cout << "parent(index)=" << parent(index) << endl;
    //cout << "heap[parent(index)]=" << heap[parent(index)] << endl;
    //cout << "heap[index]=" << heap[index] << endl;
    while ( ( index > 0 ) && ( parent(index) >= 0 ) &&
            ( heap[parent(index)] > heap[index] ) )
    {
        int tmp = heap[parent(index)];
        heap[parent(index)] = heap[index];
        heap[index] = tmp;
        index = parent(index);
    }
}

void Heap::heapifydown(int index)
{
    //cout << "index=" << index << endl;
    //cout << "left(index)=" << left(index) << endl;
    //cout << "right(index)=" << right(index) << endl;
    int child = left(index);
    if ( ( child > 0 ) && ( right(index) > 0 ) &&
         ( heap[child] > heap[right(index)] ) )
    {
        child = right(index);
    }
    if ( child > 0 )
    {
        int tmp = heap[index];
        heap[index] = heap[child];
        heap[child] = tmp;
        heapifydown(child);
    }
}

int Heap::left(int parent)
{
    int i = ( parent << 1 ) + 1; // 2 * parent + 1
    return ( i < heap.size() ) ? i : -1;
}

int Heap::right(int parent)
{
    int i = ( parent << 1 ) + 2; // 2 * parent + 2
    return ( i < heap.size() ) ? i : -1;
}

int Heap::parent(int child)
{
    if (child != 0)
    {
        int i = (child - 1) >> 1;
        return i;
    }
    return -1;
}

int main()
{
    // Create the heap
    Heap* myheap = new Heap();
    myheap->insert(700);
    myheap->print();
    myheap->insert(500);
    myheap->print();
    myheap->insert(100);
    myheap->print();
    myheap->insert(800);
    myheap->print();
    myheap->insert(200);
    myheap->print();
    myheap->insert(400);
    myheap->print();
    myheap->insert(900);
    myheap->print();
    myheap->insert(1000);
    myheap->print();
    myheap->insert(300);
    myheap->print();
    myheap->insert(600);
    myheap->print();

    // Get priority element from the heap
    int heapSize = myheap->size();
    for ( int i = 0; i < heapSize; i++ )
        cout << "Get min element = " << myheap->deletemin() << endl;

    // Cleanup
    delete myheap;
}
```

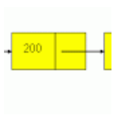
```
OUTPUT:-
Heap = 700
Heap = 500 700
Heap = 100 700 500
Heap = 100 700 500 800
Heap = 100 200 500 800 700
Heap = 100 200 400 800 700 500
Heap = 100 200 400 800 700 500 900
Heap = 100 200 400 800 700 500 900 1000
Heap = 100 200 400 300 700 500 900 1000 800
Heap = 100 200 400 300 600 500 900 1000 800 700
Get min element = 100
Get min element = 200
Get min element = 300
Get min element = 400
Get min element = 500
Get min element = 600
Get min element = 700
Get min element = 800
Get min element = 900
Get min element = 1000
```

Related Posts:

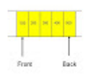
C++ Heaps (http://www.sourcetricks.com/2011/06/c-heaps.html)
What is a heap data structure? How to implement in C++? This article explains the heap data structure and provides a sample implementation using C++. Heap … Read More (http://www.sourcetricks.com/2011/06/c-heaps.html)



C++ Stacks (http://www.sourcetricks.com/2008/07/c-stacks.html)
What is a stack? How to implement stacks using C++? This article explains about the basics of a stack data structure and demonstrates with simple examples imp… Read More (http://www.sourcetricks.com/2008/07/c-stacks.html)



C++ Singly Linked Lists (http://www.sourcetricks.com/2008/07/c-singly-linked-lists.html)
What is a singly linked list? How to implement singly linked lists in C++? This article explains the concept of singly linked list data structure and provide… Read More (http://www.sourcetricks.com/2008/07/c-singly-linked-lists.html)



C++ Queues (http://www.sourcetricks.com/2008/07/c-queues.html)
What is a queue? How to implement queues using C++? This article explains the queue data structure and demonstrates sample implementation using C++. Queue… Read More (http://www.sourcetricks.com/2008/07/c-queues.html)

Binary Search Trees in C++ (http://www.sourcetricks.com/2011/06/binary-search-trees-in-c.html)
What is binary search trees? How to implement in C++? This article explains the concept of binary search trees (BST) and provides a sample implementation in … Read More (http://www.sourcetricks.com/2011/06/binary-search-trees-in-c.html)

← Newer Post (http://www.sourcetricks.com/2011/06/command-pattern.html) Older Post → (http://www.sourcetricks.com/2011/06/binary-search-trees-in-c.html)

6 comments :

swagat (http://www.blogger.com/profile/16462841016579669200) January 23, 2012 at 6:10 PM (http://www.sourcetricks.com/2011/06/c-heaps.html?showComment=1327322440873#c2250252463677855723)

Hi, in your heapifydown function, heap[index] is not being compared to heap[left] and heap[right]. I.e., (in a min heap), even if the index node has a smaller value than it's children, it is being swapped with one of them.

Reply

Replies

Luke A. January 31, 2013 at 3:00 AM (http://www.sourcetricks.com/2011/06/c-heaps.html?showComment=1359581418160#c3936355402875430884)

Yeah, you need the following in your heapifydown (this is how I did it):
// after child-right switch logic
if (left(index) > 0 && right(index) > 0)
{
if (heap[Left(index)] > heap[index] && heap[Right(index)] > heap[index])
{
child = -1;
}
}
// before child > 0

greg swanson (http://www.blogger.com/profile/04659273860268855474) March 12, 2013 at 3:11 PM (http://www.sourcetricks.com/2011/06/c-heaps.html?showComment=1363081301857#c6879944176148428574)

I don't follow your logic Luke A. because you're missing the case when there right(index) < 0 and left(index) > 0, however the above code is wrong as swagat said

the following line within heapifydown could be altered "if (child > 0)" to:

if ((child > 0) && (heap[child] < heap[index]))

this of course is to compare the lesser child weight with the index weight

Reply

Anonymous January 9, 2013 at 7:38 AM (http://www.sourcetricks.com/2011/06/c-heaps.html?showComment=1357697313270#c6865542752903825624)

Swagat -- pretty sure that the algorithm is correct. The logic should be something like:

1. Find the highest priority child node from the input starting node(which is the bigger node for a max heap, the smaller node for a min heap).

2. Swap the higher priority node with the starting node.

3. Recursively call the function on the new child node (which is now the old starting node since it's called after the swap).

Because of the heap property, you don't have to worry about the current node you are comparing against being less than (for min heap) or greater than (for max heap) any child node since heapifydown is called *after* moving the last element in the heap array to the first element. Essentially heapifydown terminates when a leaf node is reached, which in this code is signified by a -1 index return from "left" and "right" functions.

Here is it worked out by example.

Starting min heap:

array:
0243569

binary view:
0
| \
2 4
|\ |\
3 5 6 9

deletemin removes the first index, which is the lowest (here, 0), and puts the last node of the array into the first node (here, 9). Then it pops back the last element in the array. this results in a heap of:

9
| \
2 4
|\ |
3 5 6

Notice the 4 doesn't have a right child.

Now, the heapifydown function calls starting at the 0th index (9).

The first step of the heapifydown function is to find which of the children are smaller (comparing them to each other). It should return 2 (since 2 < 4). Then it swaps the starting index with the smaller of the children. Now the heap looks like:

2
| \
9 4
|\ |
3 5 6

Now it continues recursively, which results in:

2
| \
3 4
|\ |
9 5 6

Which in array-form looks like:
234956

Now, the current index doesn't have either a left child *or* a right child, so it stops. The heap now completely obeys the heap property (which is each node is bigger than either of it's children).

Reply

Anonymous May 24, 2013 at 10:12 AM (http://www.sourcetricks.com/2011/06/c-heaps.html?showComment=1369370564292#c2778287943863111312)

Hi there,
thanks for this, I used it as a base to build my own heap (with templates) and as I was doing the tests I noticed that it doesn't work well, I realized that the heapifydown wasn't doing the things right and reading Cormen I found the mistake, in line 46 you have this: if (child > 0) and you should have this: if (child > 0 && (heap[child] < heap[index]))
This is the test that failed:
int main () {
Heap *miHeap = new Heap;
miHeap->insert(3);
miHeap->insert(2);
miHeap->insert(4);
miHeap->insert(8);
miHeap->insert(5);
miHeap->insert(1);
miHeap->insert(1);
while (!miHeap->empty()){ //empty is equal to size == 0
cout << miHeap->deleteMin() << endl;
}

delete miHeap;
return 0;
}

The result of the test should be the numbers in ascendent order. Here are the results:
Whithout the correction:
1

1
3
4
2
5
8
With the correction:
1
1
2
3
4
5
8

Hope this is information to be useful to you.
Regards

Reply

Anonymous July 5, 2013 at 9:03 PM (http://www.sourcetricks.com/2011/06/c-heaps.html?showComment=1373038401914#c3726603541923838510)

No, Greg is correct. After determining which child is smaller you still have to decide whether or not the parent and child should be swapped.
In the original main(), add a second "200" value and watch the output.

Reply

Enter your comment...

Comment as: Google Accou ▼

Publish Preview

(https://www.blogger.com/comment-iframe.g?blogID=7748177500667831327&postID=6023749418398908479&blogspotRpcToken=614194)

Subscribe to: Post Comments (Atom) (http://www.sourcetricks.com/feeds/6023749418398908479/comments/default)

Tutorial Pages

Java Tutorials (http://www.sourcetricks.com/p/java-tutorials.html)

Java Interview Questions (http://www.sourcetricks.com/2014/06/core-java-interview-questions.html)

Scala Tutorials (http://www.sourcetricks.com/p/scala-tutorials.html)

Programming in C++ (http://www.sourcetricks.com/p/programming-in-c.html)

Programming interview questions and answers in C++ (http://www.sourcetricks.com/p/programming-interview-questions-and.html)

Data Structures using C++ (http://www.sourcetricks.com/p/data-structures-using-c.html)

Algorithms in C++ (http://www.sourcetricks.com/p/algorithms-in-c.html)

Design Patterns using C++ (http://www.sourcetricks.com/p/design-patterns-using-c.html)

Android (http://www.sourcetricks.com/p/android.html)

HTML, CSS and Javascript (http://www.sourcetricks.com/p/html-css-and-javascript.html)

UML Notations (http://www.sourcetricks.com/2008/05/uml-generalization.html)

Tag Cloud

Java (http://www.sourcetricks.com/search/label/Java) CPP (http://www.sourcetricks.com/search/label/CPP) Programming interview questions and answers

(http://www.sourcetricks.com/search/label/Programming%20interview%20questions%20and%20ans
Design Patterns (http://www.sourcetricks.com/search/label/Design%20Patterns) Scala (http://www.sourcetricks.com/search/label/Scala) Android (http://www.sourcetricks.com/search/label/Android) Algorithms (http://www.sourcetricks.com/search/label/Algorithms) Data Structures (http://www.sourcetricks.com/search/label/Data%20Structures) JavaScript (http://www.sourcetricks.com/search/label/JavaScript) tools (http://www.sourcetricks.com/search/label/tools) UML (http://www.sourcetricks.com/search/label/UML) html (http://www.sourcetricks.com/search/label/html)

Small Web Tools New (http://www.smallwebtools.com/)

C++ Singly Linked Lists (http://www.sourcetricks.com/2008/07/c-singly-linked-lists.html)

What is a singly linked list? How to implement singly linked lists in C++? This article explains the concept of singly linked list data ...

C++ Pre-Order, In-Order, Post-Order Traversal of Binary Search Trees (http://www.sourcetricks.com/2011/05/c-pre-order-in-order-post-order.html)

Pre-Order, In-Order, Post-Order traversal of Binary Search Trees (BST) This article explains the depth first search (DFS) traversal meth...

C++ Queues (http://www.sourcetricks.com/2008/07/c-queues.html)

What is a queue? How to implement queues using C++? This article explains the queue data structure and demonstrates sample implementati...

Javascript scroll to bottom of page (http://www.sourcetricks.com/2010/07/javascript-scroll-to-bottom-of-page.html)

This code snippet provides a JavaScript function to scroll a HTML page programatically to the bottom on page load. Make the browser height...

C++ Factory Pattern (http://www.sourcetricks.com/2008/05/c-factory-pattern.html)

Factory Pattern Factory pattern is a creational design pattern. Idea of the factory patterns is to localize the object creation code. ...

Posts

Comments

Contact Us (http://www.sourcetricks.com/p/contact.html)

Email address... Submit