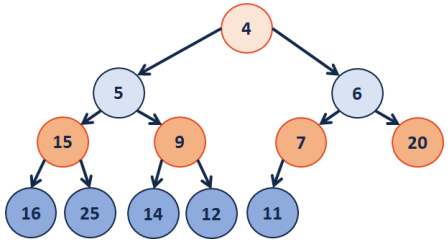
CS 225

#28: Heap Operations

March 30, 2018 · *Wade Fagen-Ulmschneider*

A Heap Data Structure

(specifically a minHeap in this exampl, as the minimum element is at the root)



-	4	5	6	15	9	7	20	16	25	14	12	11			

Given an index *i*, it's parent and children can be reached in O(1) time:

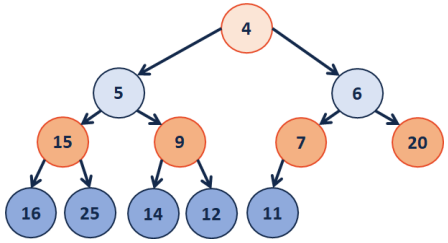
- leftChild := 2i
- rightChild := 2i + 1
- parent := floor(i / 2)

Formally, a complete binary tree T is a minHeap if:

- T = {}
- T = {r, T_l, T_r} where T_l and T_r are minHeap and r is smaller than the root of T_l and T_r

---

Inserting into a Heap



-	4	5	6	15	9	7	20	16	25	14	12	11			

Heap.cpp (partial)

```
1  template <class T>
2  void Heap<T>::_insert(const T & key) {
3    // Check to ensure there's space to insert an element
4    // ...if not, grow the array
5    if ( size_ == capacity_ ) { _growArray(); }
6
7    // Insert the new element at the end of the array
8    item_[++size] = key;
9
10   // Restore the heap property
11   _heapifyUp(size);
12 }
```

O(1)

O(lg(n))

```
31 template <class T>
32 void Heap<T>::_heapifyUp( unsigned index ) {
33   if ( index > 1 ) {
34     if ( item_[index] < item_[ parent(index) ] ) {
35       std::swap( item_[index], item_[ parent(index) ]
36 );
37       _heapifyUp( parent(index) );
38     }
39   }
40 }
```

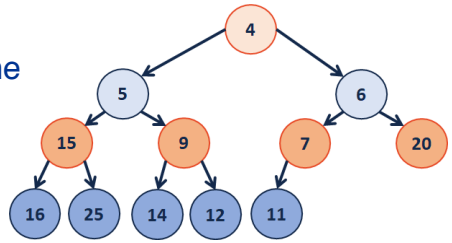
What's wrong with this code?

Running time of insert?

O(lg(n))

Heap Operation: removeMin / heapifyDown:

1. remove the root
2. replace the root with the last element in the array



-	4	5	6	15	9	7	20	16	25	14	12	11			

Heap.cpp (partial)

```
1  template <class T>
2  void Heap<T>::_removeMin() {
3    // Swap with the last value
4    T minValue = item_[1];
5    item_[1] = item_[size_];
6    size--;
7
8    // Restore the heap property
9    heapifyDown();
10
11   // Return the minimum value
12   return minValue;
13 }
```

```
1  template <class T>
2  void Heap<T>::_heapifyDown(int index) {
3    if ( !_isLeaf(index) ) {        // base case
4      T minChildIndex = _minChild(index);   // select smaller element
5      if ( item_[index] > item_[minChildIndex] ) {
6        std::swap( item_[index], item_[minChildIndex] );
7        _heapifyDown( minChildIndex );
8      }
9    }
10 }
```

Theorem: The running time of buildHeap on array of size n is: _____.

Strategy:

Define S(h):
Let **S(h)** denote the sum of the heights of all nodes in a complete tree of height **h**.

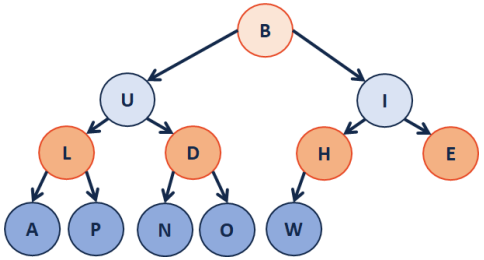
S(0) =

S(1) =

S(h) =

Proof of S(h) by Induction:

Q: How do we construct a heap given data?



1. sort the array -> heap O(n lg(n))
2. heapifyUp -> insert every new element O(lg(n)) * O(n) = n O(lg(n))
3. heapifyDown -> assume already heap, perform heapifyDown

-	B	U	I	L	D	H	E	A	P	N	O	W			

Heap.cpp (partial)

```
1  template <class T>
2  void Heap<T>::buildHeap() {
3    for (unsigned i = parent(size); i > 0; i--) {
4      heapifyDown(i);
5    }
6  }
```

Running Time?

Finally, finding the running time:

CS 225 – Things To Be Doing:

1. Theory Exam 3 starts next week (Tuesday, April 3rd)
2. MP5 deadline is Monday, April 2nd
3. lab_hash is due Sunday, April 1st
4. Daily POTDs are ongoing!