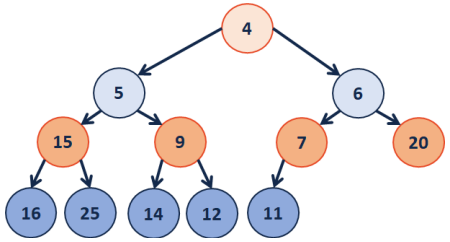
CS 225

#29: Heap Operations

November 1, 2019 · *G Carl Evans*

A Heap Data Structure

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



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

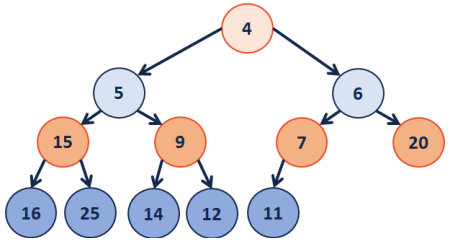
Given an index *i*, it's parent and children can be reached in O(1) time:

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

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

- **T = {} or**
- **T = {r, T_L, T_R} and r is less than the roots of T_L, T_R and T_L, T_R are minHeaps**

Inserting into a Heap



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

Heap.hpp (partial)

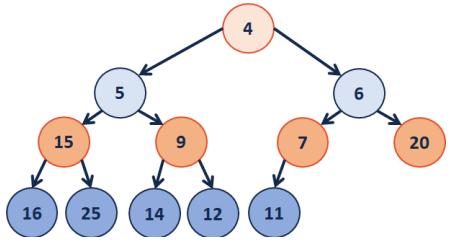
```
 1  template <class T>
 2  void Heap<T>::_insert(const T & key) {
 3    // Check to ensure there's space to insert an element
 4    // ...if not, grow the array
 5    if ( size_ == capacity_ ) { _growArray(); }
 6
 7    // Insert the new element at the end of the array
 8    item_[++size] = key;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

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

How do we complete this code?

Running time of insert?

Heap Operation: removeMin / heapifyDown:



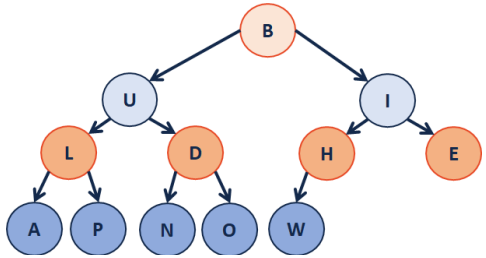
-	4	5	6	15	9	7	20	16	25	14	12	11			

Heap.hpp (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  }
```

```
51  template <class T>
52  void Heap<T>::_heapifyDown(int index) {
53    if ( !_isLeaf(index) ) {
54      T minChildIndex = _minChild(index);
55      if ( item_[index] ___ item_[minChildIndex] ) {
56         std::swap( item_[index], item_[minChildIndex] );
57         _heapifyDown( _________________ );
58      }
59    }
60  }
```

Q: How do we construct a heap given data?



–	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?

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:

Finally, finding the running time:

CS 225 – Things To Be Doing:

1. Theory Exam 3 starts on Thursday (topic list online, more soon!)
2. MP5 due date: Monday, November 4th
3. lab_hash is due Sunday, November 3th
4. Daily POTDs are ongoing!