

How might we build a queue where our 'front' is always the min element?

Priority Queue ADT:

- insert
- remove
- isEmpty

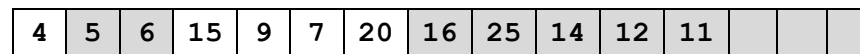
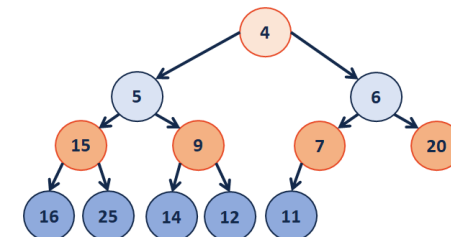
Implementation of PriorityQueue

insert	removeMin	Implementation
O(n)	O(n)	Unsorted Array
O(1)	O(n)	Unsorted List
O(lg(n))	O(1)	Sorted Array
O(lg(n))	O(1)	Sorted List

Q1: What errors exist in this table? (Fix them!)

Q2: Which algorithm would we use?

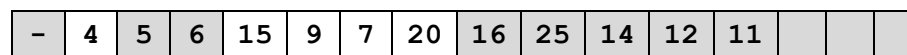
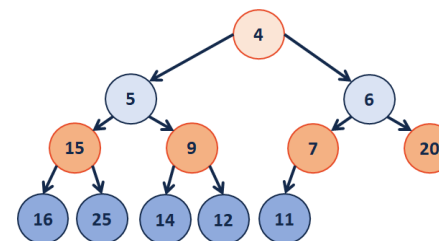
Implementing a (min)Heap as an Array



Operations:

- leftChild(index) :=
- rightChild(index) :=
- parent(index) :=

Inserting into a Heap



...running time?

```

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         };
37         _heapifyUp( _____ );
38     }
39 }
40 }

```

What's wrong with this code?

```

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

```

Q: How do we construct a heap given data?

Heap Operation: removeMin / heapifyDown:

