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

Given an index \( i \), its parent and children can be reached in O(1) time:
- \( \text{leftChild} := 2i \)
- \( \text{rightChild} := 2i + 1 \)
- \( \text{parent} := \lfloor i / 2 \rfloor \)

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

How do we complete this code?

Running time of insert?
Q: How do we construct a heap given data?

```
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(1);
10    // Return the minimum value
11    return minValue;
12 }
```

```
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 }
```

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. mp_traversals EC due today.
2. Daily POTDs are ongoing!
```