### Build Heap: Complexity Analysis

<table>
<thead>
<tr>
<th>Level</th>
<th># of nodes</th>
<th># of steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>h</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>h – 1</td>
</tr>
<tr>
<td>2</td>
<td>2^2</td>
<td>h – 2</td>
</tr>
<tr>
<td>3</td>
<td>2^3</td>
<td>h – 3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>i</td>
<td>2^i</td>
<td>h – i</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>h-1</td>
<td>2^h-1</td>
<td>h – (h – 1)</td>
</tr>
</tbody>
</table>

- **Total # of steps:** \( S = h + 2(h-1) + 2^2(h-2) + \ldots + 2^{h-1}(1) \)
- **Multiplying by 2:** \( 2S = 2h + 2^2(h-1) + 2^3(h-2) + \ldots + 2^h(1) \)
- **Subtracting:**
  \[
  S = h + 2 + 2^2 + 2^3 + \ldots + 2^{h-1} + 2^h \\
  = (2^{h+1} – 1) – (h + 1) = O(N)
  \]

### Binary Heaps: Other Operations

- **Heap operations**
  - \( \text{min}: O(1) \)
  - \( \text{insert}: O(\log N) \)
  - \( \text{deleteMin}: O(\log N) \)

- **Additional operations:**
  - \( \text{decreaseKey}(p, \Delta): O(\log N) \)
  - \( \text{increaseKey}(p, \Delta): O(\log N) \)
  - \( \text{delete}(p): O(\log N) \)
  - \( \text{buildHeap}: O(N) \)
  - \( \text{find}(x): \text{find element } x \)
  - \( \text{findMax}: \text{find the maximum element} \)
  - \( \text{merge}(H1, H2): \text{merge two heaps, each of size } O(N) \)
    - \( O(N) \) inserts => \( O(N \log N) \) time
    - \( \text{Copy } H2 \text{ at end of } H1 \text{ and use buildHeap } => O(N) \text{ time!} \)
Heapsort  

```java
public static <AnyType extends Comparable<? Super AnyType>> void heapSort(AnyType[] arr) {
    BinaryHeap<AnyType> pq =
        new <BinaryHeap><AnyType>(arr.length);
    for(int i=0; i<arr.length; i++)
        pq.insert(arr[i]);
    for(int i=0; i<arr.length; i++)
        arr[i] = pq.deleteMin();
}
```

In-class Problem: Binary Trees

- A complete binary tree with N elements
  - Uses array positions 1 to N (array of size N)

- If tree is not complete, how large must the array be:
  - If the tree has two extra levels (slightly imbalanced)?
  - If deepest node is at depth 4.1 log N?
  - In the worst-case?
In-class Problem: Using Linked Lists

- Suppose a binary heap is represented using explicit links
  ```java
  public static class binaryNode<AnyType> {
      AnyType element;
      binaryNode<AnyType> left;
      binaryNode<AnyType> right;
      ...}
  ```

- Give a simple algorithm to find the tree node at implicit position k (array position k)
  - Hint: consider the binary representation of k (e.g., 10011)

In-class Problem: Merging Heaps

- Suppose a binary heap is represented using explicit links
  ```java
  public static class binaryNode<AnyType> {
      AnyType element;
      binaryNode<AnyType> left;
      binaryNode<AnyType> right;
      ...}
  ```

- Consider two heaps L, R that have last levels complete
  - Give an O(log N) algorithm to merge if l = r

  - Give an O(log N) algorithm if r = l + 1