Q: An optimal buildHeap operation:

![Heap Diagram]

Heap.hpp (partial)

```cpp
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):

$S(h) :=$

$S(0) =$

$S(1) =$

$S(h) =$

Disjoint Sets

Let $R$ be an equivalence relation. We represent $R$ as several disjoint sets. Two key ideas from Monday:

- Each element exists in exactly one set.
- Every set is an equitant representation.
  - Mathematically: $4 \in [0]_R \Rightarrow 8 \in [0]_R$
  - Programmatically: find(4) == find(8)

Building Disjoint Sets:

- Maintain a collection $S = \{s_0, s_1, \ldots, s_k\}$
- Each set has a representative member.
- ADT:
  ```cpp
  void makeSet(const T & t);
  void union(const T & k1, const T & k2);
  T & find(const T & k);
  ```
Implementation #1: Representative Member Array

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

**Operation:** find(k)
...running time?

**Operation:** union(k1, k2)
...running time?

Implementation #2: UpTrees

- Continue to use an array where the index is the key
- The value of the array is:
  - -1, if we have found the representative element
  - The index of the parent, if we haven’t found the rep. element

Step-by-step construction of UpTrees:

Example:

<table>
<thead>
<tr>
<th>2</th>
<th>5</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

...where is the error(s) in this table?

**Implementation – DisjointSets::find**

```cpp
int DisjointSets::find(int i) {
    if ( s[i] < 0 ) { return i; }
    else { return _find( s[i] ); }
}
```

What is the running time of find?

What is the ideal UpTree?

**Implementation – DisjointSets::union**

```cpp
void DisjointSets::union(int r1, int r2) {
}
```

How do we want to union the two UpTrees?

**CS 225 – Things To Be Doing:**

1. Theory Exam 3 starts Thursday; **Practice Exam Available!**
2. MP5 due tonight at 11:59pm
3. Lab Section: lab_puzzles coming up this week in lab!
4. Daily POTDs are ongoing!