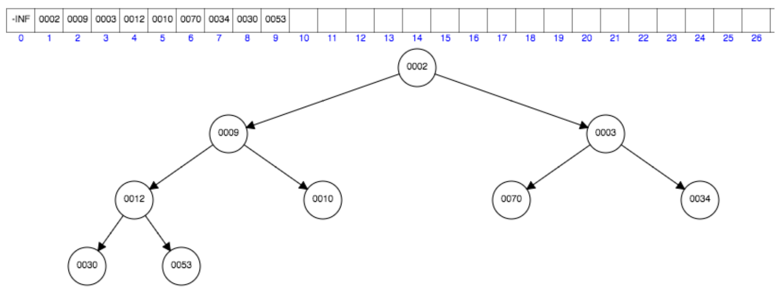
Due Dec. 9, 11.59pm

In this project, you will implement a min-Heap data structure using a USGIS earthquake dataset. You will implement insert and delete operations on the heap. While the heap is represented as an array internally, you will display this as a singly linked list using BRIDGES (and for extra credit, as a binary tree structure (nodes and links) in BRIDGES). An example is shown below: **Requirements.**



1. You have to implement a min heap data structure; for convenience, refer to the max heap implementation provided in the accompanying handout, that you can use as a guideline; **however, note that we are building a min-Heap in this project and this will require adapting this implementation.** The representation of the heap is a 1D array(of size N+1, with the first element left unused - see figure above).

2. **Dataset.** You will use a USGIS Earthquake dataset that is already built into BRIDGES (see the EarthquakeUSGS class for details). To acquire the dataset, issue the following call from BRIDGES (after initializing BRIDGES):

```
        // Set up the earthquake user
USGSaccount name = new USGSaccount( "earthquake" );

        // Retrieve a list of (maxElements) Tweets
List<EarthquakeUSGS> eqlist = Bridges.getAssociations(name, maxElements );
```

   This retrieves maxElement(an integer) number of earthquake records of type *EarthquakeUSGS* from BRIDGES and puts it into a Java List (`http://docs.oracle.com/javase/7/docs/api/java/util/List.html`). You can now use these data items in your heap.

3. **Building the Heap.** Ignore the buildHeap() method in the handout; you will build the min heap using a sequence of insert() calls. Insert the earthquake records into the heap, one at a time.

4. **Delete Operation.** Deleting an element from the heap will remove the root element, followed by a reheap. Test your delete operation a few times , by deleting an element from the heap and then redisplaying (calling visualize()) the heap.

5. **Display.** You can display the contents of the heap in BRIDGES using a singly linked list (using SLelement¡EarthquakeUSGIS¿); simply copy the records from your array into a singly linked list, with the minimum element at the beginning of the list. Use the label field to display the earthquake details(magnitude, location, etc); see the example at `http://bridges-cs.herokuapp.com/assignments/12/bridges_public`

Extra Credit. Rather than display the heap using a list, we want to display its tree(logical) representation, similar to the figure above. For this you will need to traverse the list (its tree structure) starting at the root and build a binary tree. You can use the **BinTreeElement<E>** class (where E will USGSEarthquake) in BRIDGES for this purpose.

Thus, your heap traversal algorithm will start with root node (at index 1), find its two children(at indices 2 and 3 – in general if a parent is at index i, then its children are at 2i and 2i+1 in a heap) and recursively follow its children, and so on. Traversal stops when the index exceeds the array size. Use the setLeft() and setRight() methods of BinTreeElement to assign the children. Once the heap is built, use the setDataStructure() and visualize() methods to display the heap as a tree structure.

You can simply write a method within the Heap class to implement the traversal to construct the binary tree.

Submission Requirements.

Turn in your source code to Canvas by the deadline; ensure it is well documented. An interactive demo will be required for evaluation.