1. A binary *max heap* is a complete binary tree with the additional property that …

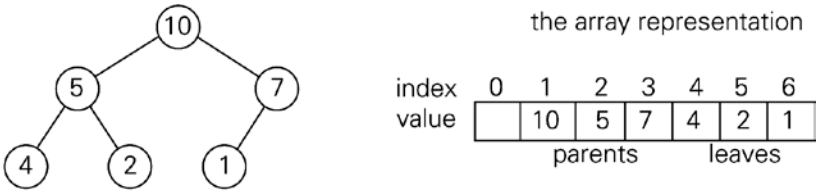

FIGURE 6.10 Heap and its array representation

2. Describe the details of the representation of a binary (max) heap by an array.

   Runtime for `insert` and `removeMax`: (why?)

3. Give a brief overview of how heapsort works.

4. **Next three questions:** In a full binary heap (thus N is one less than a power of 2) represented as an array, let *depth(i)* be the depth of the node whose subscript in the array is i, and let *height(i)* be the height of the node whose subscript in the array is i.

5. Give a simple formula (as a function of i) for *depth(i).*

6. Show that the sum of the depths of all of the nodes in the tree is Θ (N log N).

7. depth(i) + height(i) = _____

8. Which is the faster of the two approaches for building the initial heap for HeapSort, and why?

9. How can we use a “precalculation” with the adjacency matrix of a graph to count paths of length 2 in that graph?

10. List several examples of algorithms where using additional space allows us to solve a problem faster.

11. (with partners) Quickly discuss these issues.
    What problem do we try to solve by hashing?
    What is the general idea of how hashing works?
    Why does it fit into Chapter 7 (space-time tradeoffs)?
    What are the main issues to be addressed when discussing hashing implementation?
    How to choose between a hash table and a binary search tree?

12. Discuss with partners to be sure that you can define these terms:
    hash function, collision, load factor (α) , perfect hash function, open addressing, linear probing, cluster, quadratic probing, rehashing, separate chaining. When doing an insertion into the table, how does the value of α affect expected number of probes. If you need hints, see the PowerPoint slides at
    http://www.rose-hulman.edu/class/csse/csse230/201230/Slides/17-Graphs-HashTables.pdf