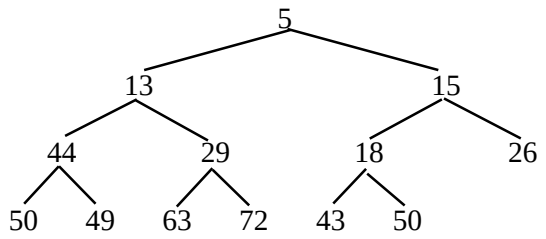
ECS 60 Written Homework #5: Priority Queues and Disjoint Sets (19 points) Fall 2015

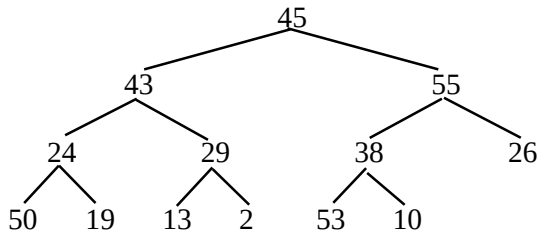
Due Monday, November 9th, 4:00 pm in 2131 Kemper

1. (4 points, 2 points each) With the following binary heap as a starting point for each of the following, show the resulting heap after completing both of the specified operations. You should draw only one heap for each part.



   a. Insert 20, insert 4
   b. DeleteMin, deleteMin.

2. (2 points) Assuming that the following values are in an array, use buildHeap described in section 6.3 (pp. 255 -257) in the text to create a binary heap. You need only show the final binary heap.



3. (2 points) Weiss 6.15a which says, "Suppose we need to perform *M* percolateUps(), and *N* deleteMins() on a *d*-heap that initially has *N* elements.
   a. What is the total running time of all operations in terms of *M*, *N*, and *d*

4. (8 points, 2 points each) Given the following set of operations, show the final disjoint set tree for each of the following union strategies.
   union(A, D), union(C, B), union (F, E), union (G, D), union(F, A), find(A), union(H, E), union(E, G), find(E)

   a. Arbitrary, where the set specified first in the union will always be the root of the combined set.
   b. Union by size. When the sets are the same size, the set with a root that is closer to the beginning of the alphabet should be the root of the merged set.
   c. Union by height. When the sets are the same height, the set with a root that is closer to the beginning of the alphabet should be the root of the merged set.
   d. Arbitrary with path compression. The set specified first in the union will always be the root of the merged set.

5. (3 points) Is there a heap *T* storing seven distinct elements such that a preorder traversal of *T* yields the elements of *T* in sorted order? How about an inorder traversal? How about a postorder traversal? For each question, if you answer yes, then provide a drawing of the heap. (from Michael T. Goodrich, Roberto Tamassia, and David Mount, *Data Structures & Algorithms, Second Edition*, Hoboken, NJ, John Wiley & Sons, 201., p.362.)