Homework 4
(Chosen from various homeworks, thanks!)

4.1: Assuming it's a max heap
1. Second largest element is left or right child of root (max element), i.e. second level
2. Third largest is either left or right child of root, or on the third level as a child of the second largest element
3. In the last level as a leaf of the tree

4.8

4.12

4.12 Array: 1 4 3 2 5 7 6 8
1 4 3 8 5 7 6 2
1 4 7 8 5 3 6 2
1 8 7 4 5 3 6 2
8 1 7 4 5 3 6 2
8 5 7 4 1 3 6 2

Final Heap:
8
5 1
4 1 3 6
4.17
Heapsort involves $\Theta(n)$ calls to Sift-down procedure that takes $\Theta(\log i)$ time in all cases because height of heap with $i$ elements is $\Theta(\log i)$. Therefore, in all cases, the time is

$$\sum_{i=1}^{n-1} \log i = \log((n - 1)!) = \Theta(n \log n)$$

4.21

A $d$-heap is a generalization of the binary heap discussed in this chapter. It is represented by an almost complete $d$-ary rooted tree for some $d \geq 3$. Rewrite procedure SIFT-UP for the case of $d$-heaps. What is its time complexity?

It is the same process as for a binary heap.

Example:

```
    21
   / \   \
  17  13
 / \   / \  \
 9  15 4 10
```

Time complexity = $O($height$) = O(\log_d n)$

```python
done = false
if i = 1 then exit
repeat
    if key(H[i]) > key(H[i/d]) then
        interchange $H[i]$ and $H[i/d]$
    else
        done = true
        $i = i/d$
until $i = 1$ or done
```
4.26

The sequence of unions will create some trees, where each tree corresponds to a disjoint set. Each union takes two roots to make a bigger tree, without calling for finds.

The cost of a single find (with path compression) is one plus the number of new children of the root. The cost of m finds will be m plus the number of children (less than n) of the root.

4.28

The sequence of unions will create some trees, where each tree corresponds to a disjoint set. Each union takes two roots to make a bigger tree, without calling for finds.

The cost of a single find (with path compression) is one plus the number of new children of the root. The cost of m finds will be m plus the number of children (less than n) of the root.
Solve exercise 4.26 using the weight-balancing rule and path compression.

**Weight-balancing rule:**

The action of the operation \texttt{Union}(x,y) is to let the root of the tree with fewer nodes point to the root of the tree with a larger number of nodes.

If both trees have the same number of nodes, then let y be the parent of x.

\begin{itemize}
  \item [a)] Initial configuration
    \begin{center}
      \begin{tikzpicture}
        \node (1) at (0,0) {1};
        \node (2) at (1,0) {2};
        \node (3) at (2,0) {3};
        \node (4) at (3,0) {4};
        \node (5) at (4,0) {5};
        \node (6) at (5,0) {6};
        \node (7) at (6,0) {7};
        \node (8) at (7,0) {8};
      \end{tikzpicture}
    \end{center}
  \end{itemize}

\begin{itemize}
  \item [b)] First four \texttt{Union} operations: \texttt{Union}(1,2), \texttt{Union}(3,4), \texttt{Union}(5,6), \texttt{Union}(1,3)
    \begin{center}
      \begin{tikzpicture}
        \node (1) at (0,0) {1};
        \node (2) at (1,0) {2};
        \node (3) at (2,0) {3};
        \node (4) at (3,0) {4};
        \node (5) at (4,0) {5};
        \node (6) at (5,0) {6};
        \node (7) at (6,0) {7};
        \node (8) at (7,0) {8};
      \end{tikzpicture}
    \end{center}
  \end{itemize}

\begin{itemize}
  \item [c)] Next two \texttt{Union} operations: \texttt{Union}(1,5), \texttt{Union}(3,7)
    \begin{center}
      \begin{tikzpicture}
        \node (1) at (0,0) {1};
        \node (2) at (1,0) {2};
        \node (3) at (2,0) {3};
        \node (4) at (3,0) {4};
        \node (5) at (4,0) {5};
        \node (6) at (5,0) {6};
        \node (7) at (6,0) {7};
        \node (8) at (7,0) {8};
      \end{tikzpicture}
    \end{center}
  \end{itemize}

\begin{itemize}
  \item [d)] Next \texttt{Find} operation: \texttt{Find}(1)
    \begin{center}
      \begin{tikzpicture}
        \node (1) at (0,0) {1};
        \node (2) at (1,0) {2};
        \node (3) at (2,0) {3};
        \node (4) at (3,0) {4};
        \node (5) at (4,0) {5};
        \node (6) at (5,0) {6};
        \node (7) at (6,0) {7};
        \node (8) at (7,0) {8};
      \end{tikzpicture}
    \end{center}
  \end{itemize}

\begin{itemize}
  \item [e)] Next \texttt{Union} operation: \texttt{Union}(1,5)
    \begin{center}
      \begin{tikzpicture}
        \node (1) at (0,0) {1};
        \node (2) at (1,0) {2};
        \node (3) at (2,0) {3};
        \node (4) at (3,0) {4};
        \node (5) at (4,0) {5};
        \node (6) at (5,0) {6};
        \node (7) at (6,0) {7};
        \node (8) at (7,0) {8};
      \end{tikzpicture}
    \end{center}
  \end{itemize}

\begin{itemize}
  \item [f)] Last \texttt{Find} operation: \texttt{Find}(1)
    \begin{center}
      \begin{tikzpicture}
        \node (1) at (0,0) {1};
        \node (2) at (1,0) {2};
        \node (3) at (2,0) {3};
        \node (4) at (3,0) {4};
        \node (5) at (4,0) {5};
        \node (6) at (5,0) {6};
        \node (7) at (6,0) {7};
        \node (8) at (7,0) {8};
      \end{tikzpicture}
    \end{center}
  \end{itemize}

**Note:**

Both this and the problem in 4.26 end up being the same because the rank of weight are the same.