1. (a) $[\log_d(nd - n + 1)]$

(b) First remove the root. The root is the minimum element of the heap. Move the last element to the root. Then DOWN-HEAPIFY the root to maintain the property of the heap.

Notice that the running time of DOWN-HEAPIFY is a little different from binary heap. Each time a node has to compare with all its $d$ children and exchanges with the smallest one, taking $O(d)$ time. So the running time is $O(dH) = O(d \log_d n)$

(c) Insert the new node to the end of the heap. Then UP-HEAPIFY the new node.

The running time is still $O(H) = O(\log_d n)$ because in each step a node only has to compare with its parent.

2. (a) The recurrence relation for running time is $T(n) = 2T(n/2) + O(1)$. The running time is $O(n)$.


3. This problem can be easily proved by induction.

4. A single tree with 2 nodes can be constructed by first insert 3 nodes consecutively, then remove the minimum node.

If we have a Fibonacci heap consisting of a single tree of $n - 1$ nodes, we can construct a Fibonacci heap consisting of a single tree of $n$ nodes. First insert 3 new nodes to the heap, 2 of them should be smaller than the current min. Then DELETE-MIN. The two single node in the root cycle are both of degree 0, so they will be linked together. Now
the Fibonacci heap contains two trees, one is a linear chain of \( n - 1 \) nodes, the other is a linear chain of 2 nodes. The degrees of the two roots are both 1, so they will also be linked together. Now delete the single node branch of the new tree. Now we have a Fibonacci heap consisting of a single tree of \( n \) nodes.

5. (a) \( O(\log \log n) \)
   
   Let \( n = 2^m \). The recursive relation becomes \( T(2^m) = T(2^{m/2}) + 1 \). Letting \( S(m) = T(2^m) \), we have \( S(m) = S(m/2) + 1 \). Thus \( S(m) = O(\log m) \), and \( T(n) = O(\log \log n) \).

(b) \( O(n^2) \)
   
   Use the Master Method.

(c) \( O(n \log n) \)
   
   The easiest way is to look at the recursion tree.

(d) \( O(n \log \log n) \)
   
   Divide both sides of the equation by \( n \), we get \( \frac{T(n)}{n} = \frac{T(\sqrt{n})}{\sqrt{n}} + 1 \). Now let \( S(n) = \frac{T(n)}{n} \), \( S(n) = S(\sqrt{n}) + 1 \). By problem (a) we know that \( S(n) = O(\log \log n) \), so \( T(n) = O(n \log \log n) \).