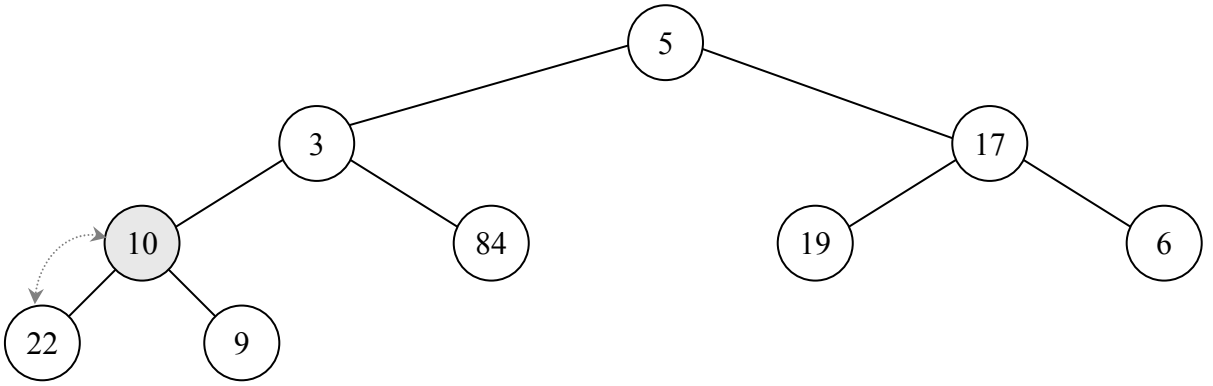
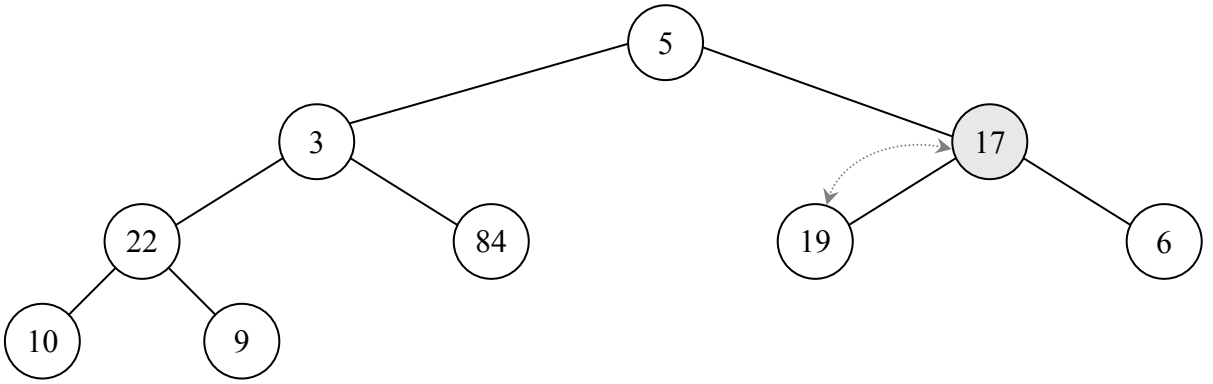
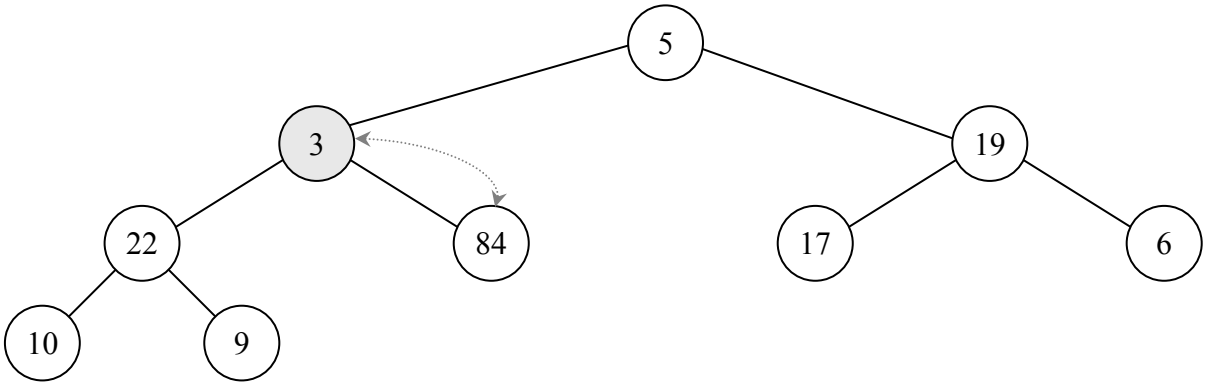
Solutions to CS / MCS 401 Week #6-7 Exercises (Spring 2008)

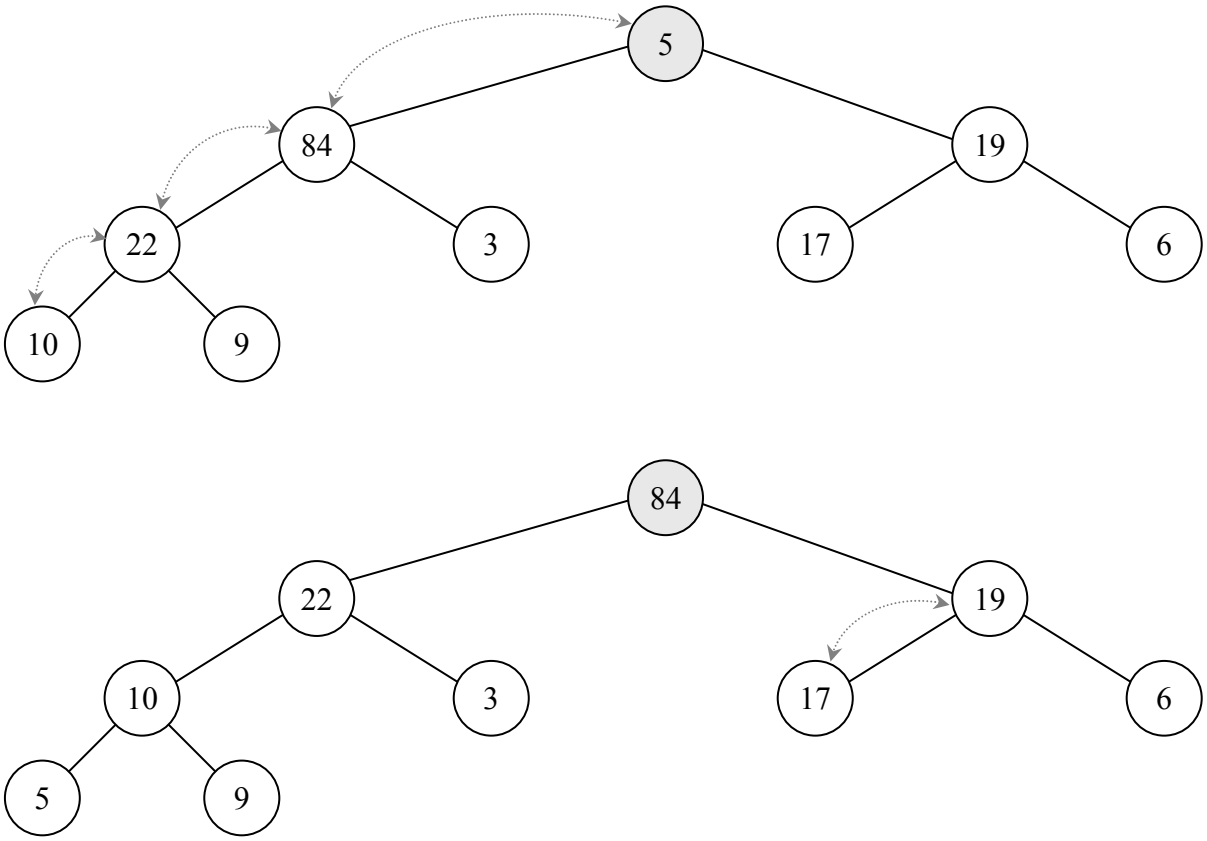
Exercise 6.3-1

A = | 5 | 3 | 17 | 10 | 84 | 19 | 6 | 22 | 9 |









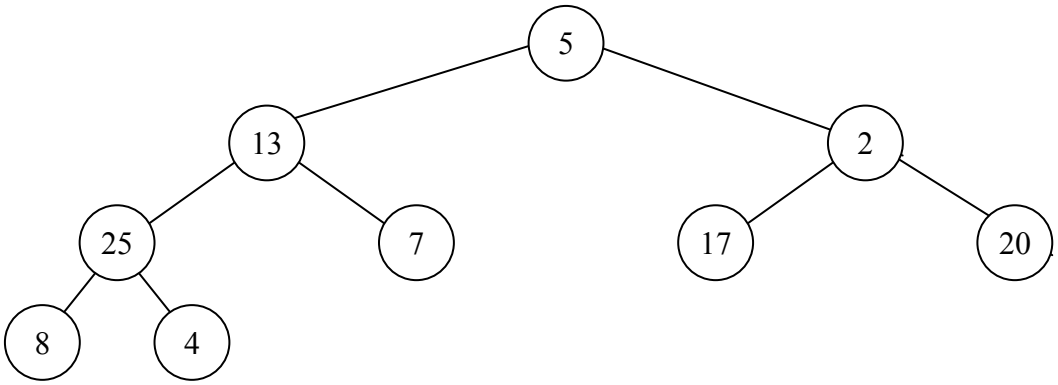
Exer 6.3-2

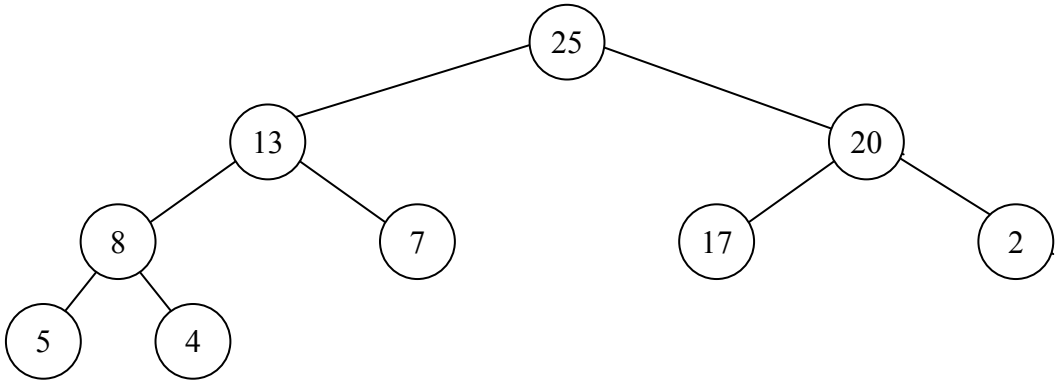
In order to apply *max-heapify*() to node i, the left and right subtrees of node i (i.e., the subtrees rooted at nodes $2i$ and $2i+1$) must already be heaps. By applying *max-heapify*() to nodes in order of decreasing i, this will always be the case.

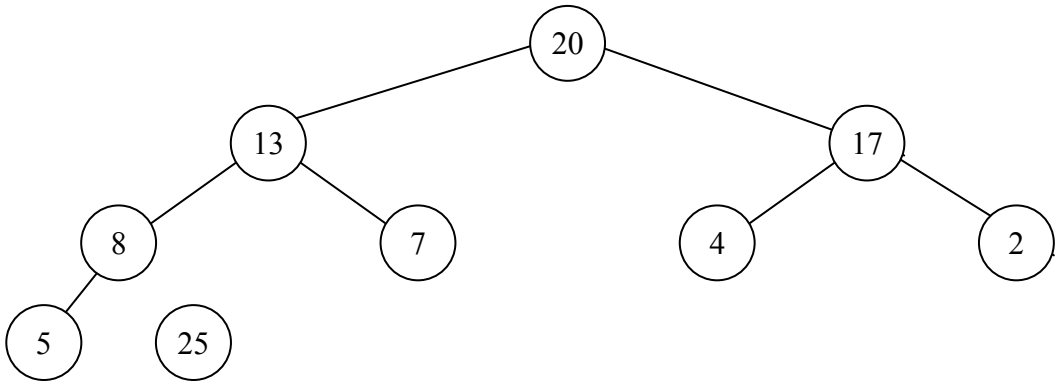
Exer 6.4-1

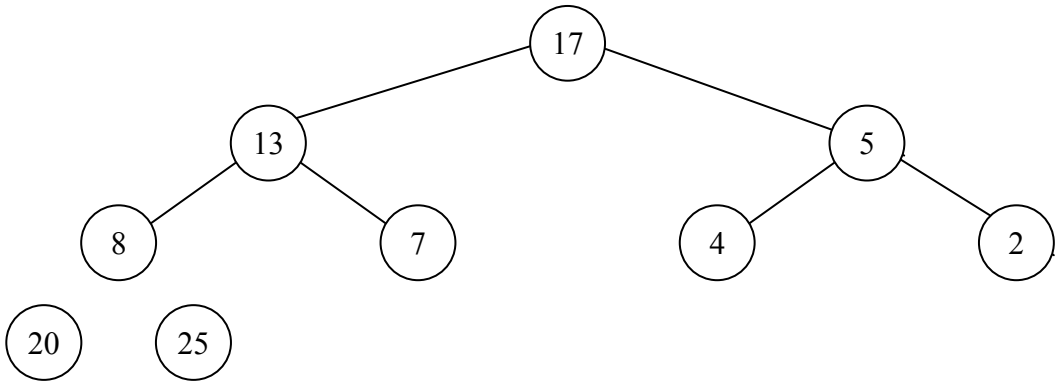
A = 5 13 2 25 7 17 20 8 4

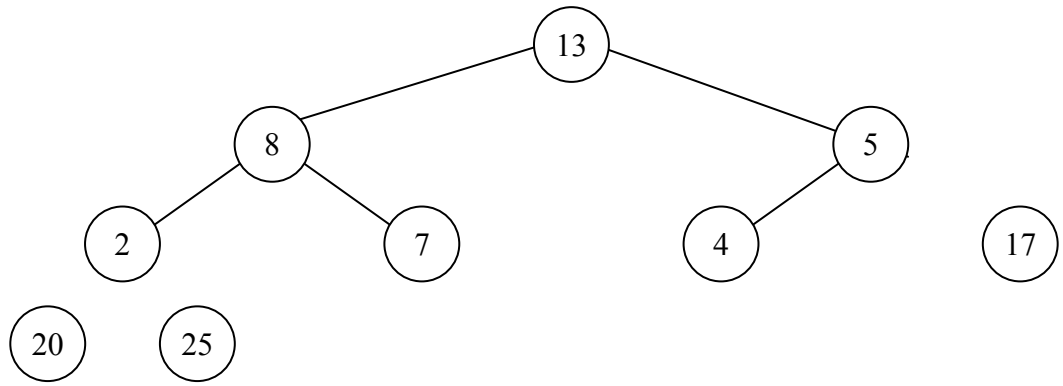
The diagrams below show the original array, the array after *build_Max_Heap*(), and the array after each step of converting the heap into a sorted array. Note you need show only the first two steps (first four diagrams).

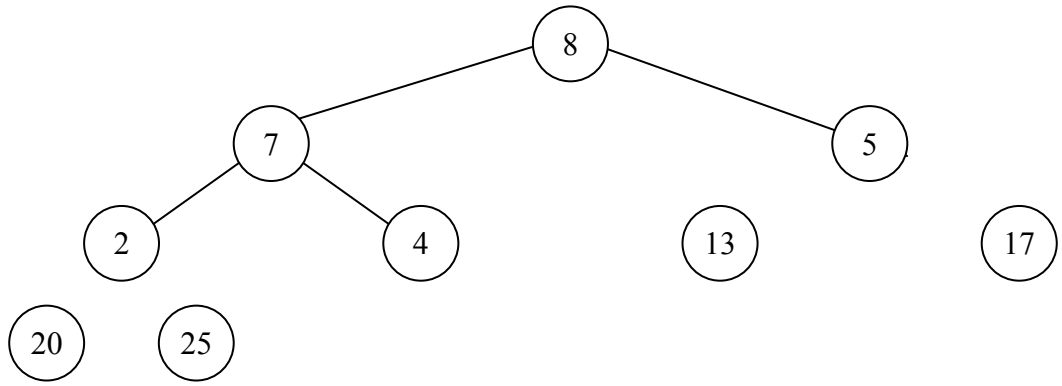

5
13
2
25
7
17
20
8
4

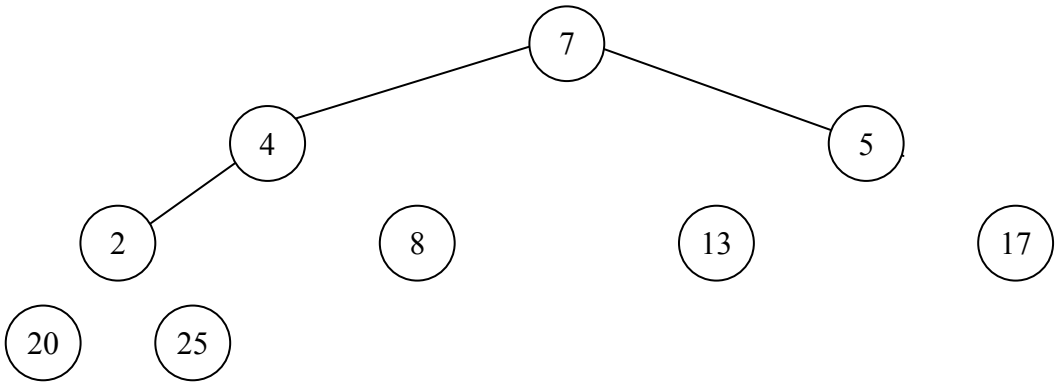

25
13
20
8
7
17
2
5
4

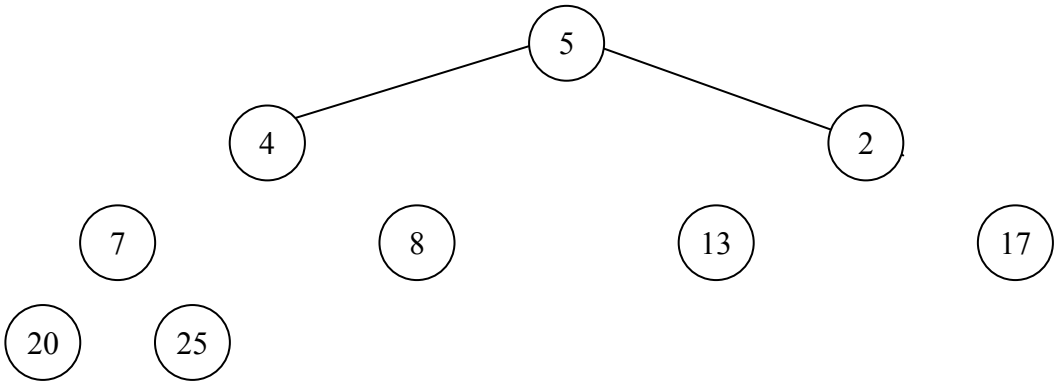

20
13
17
8
7
4
2
5
25

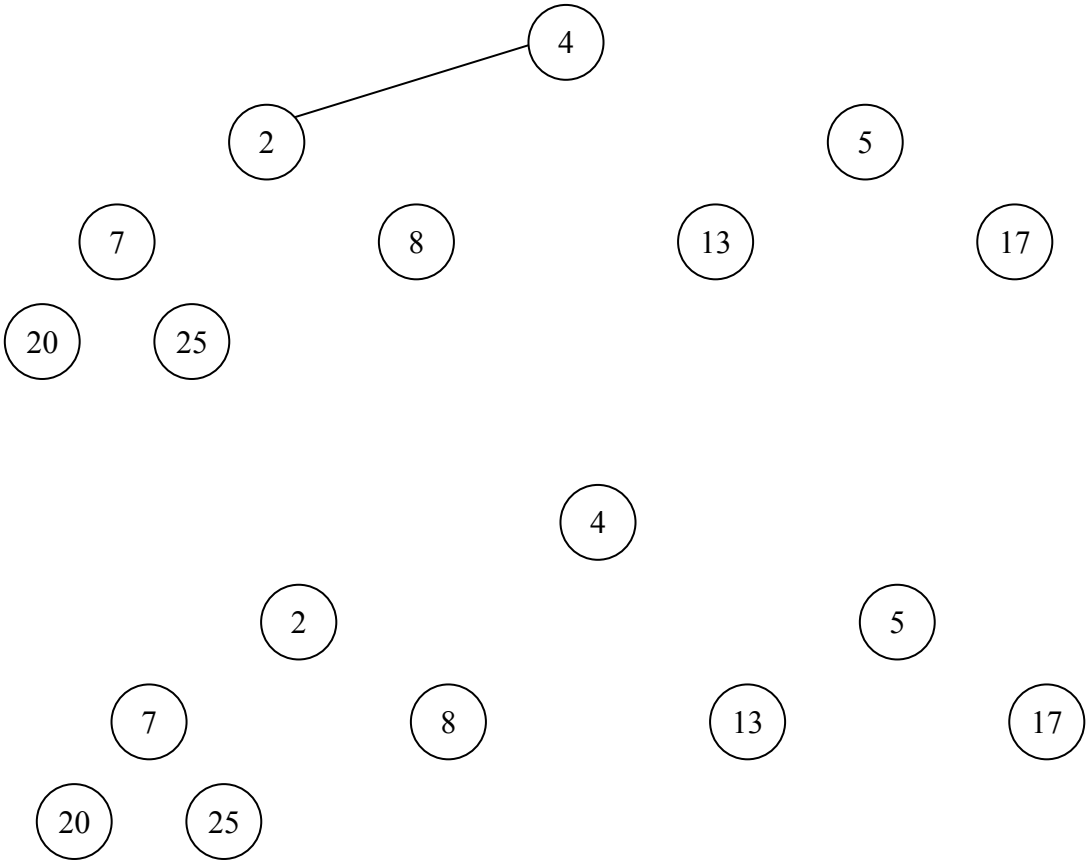

17
13
5
8
7
4
2
20
25


13
8
5
2
7
4
17
20
25


8
7
5
2
4
13
17
20
25


7
4
5
2
8
13
17
20
25


5
4
2
7
8
13
17
20
25



Exercise 6.5-7

```
heap-delete( A, i, n);
    swap(A[i],A[n]);
    n = n – 1;
    while ( i > 1 and A[i] > A[⌊i/2⌋] )
        swap(A[i], A[⌊i/2⌋] );
        i =⌊i/2⌋;
    while ( 2i ≤ n )
        if ( 2i+1 ≤ n and A[2i+1] > A[2i] )
            p = 2i+1;
        else
            p = 2i;
        if ( A[i] < A[p] )
            swap(A[i], A[p]);
            i = p;
        else
            return;
```

Note: The first while loop moves A[i] up if necessary; the second moves it down if necessary. At most one of the loops will actually be traversed.

Exercise 8.1-3 Suppose a comparison sorting algorithm runs in linear time from some fraction $\delta(n)$ of its inputs. This means that there exists a constant c (not depending on n) such that, for all n sufficiently large, the algorithm performs at most cn comparisons for $\delta(n)n!$ of its $n!$ inputs. In the decision tree, there must be at least $\delta(n)n!$ leaves at depth cn or less. But we know that the number of leaves at depth cn or less is bounded by 2^{cn}. So $\delta(n)n! \le 2^{cn}$, or $\delta(n) \le 2^{Cn}/n!$. Approximating $n!$ by Stirling's formula gives

$$\delta(n) \le 2^{cn}/n! \le 2^{cn}/((n/e)^n sqrt(2\pi n)) = (2^c e/n)^n/sqrt(2\pi n).$$

Exercise 8.1-3 asks specifically about the case $\delta(n) = 1/2$, $\delta(n) = 1/n$, and $\delta(n) = 1/2^n$. In none of these cases is $\delta(n) \le (2^c e/n)^n/sqrt(2\pi n)$ for some constant c and all n sufficiently large. If $\delta(n) = 1/2^n$, then $\delta(n)/((2^c e/n)^n/sqrt(2\pi n)) = (n/2^{1+c}e)^n sqrt(2\pi n)$ approaches ∞ as n approaches ∞, since $n/2^{1+c}e > 1$ for all n sufficiently large. So a comparison sorting algorithm cannot run in linear time even for $1/2^n$ of its inputs.

Exercise K

a) The lower bound on $C_{max}(n)$, the worst-case number of comparisons that an algorithm performs in sorting n distinct elements, is $\lceil \lg(n!) \rceil$. When $n = 5$, the lower bound on $C_{max}(n)$ is $\lceil \lg(5!) \rceil = \lceil \lg(120) \rceil =$ **7 comparisons**.

b) If a sorting algorithm achieves the lower bound on $C_{max}(n)$ in (a), the decision tree must have height 7. The number of leaves in the decision tree is at least 120. If in addition the algorithm performs only four comparisons for two or more inputs, it must have 2 leaves at depth 4, call them x and y. If we replace each of x and y by a complete binary tree of height 3, the height of the entire binary tree remains 7, but the number of leaves increases by 14. (Leaf x is replaced by a subtree with 8 leaves, and likewise y is replaced by a subtree with 8 leaves.) This produces a binary tree of height 7 with at least 134 leaves, violating the bound of $2^7 = 128$ on the number of leaves.

Exercise L

$$\binom{n}{n/2} = n! \,/\, ((n/2)!)^2 \approx 2^{n - \lg(n)/2 - 0.326}.$$

$$\approx (n/e)^n \text{ sqrt}(2\pi n) \,/\, \left((n/2e)^{n/2} \text{ sqrt}(2\pi n/2)\right)^2 \quad \text{using Stirling's formula}$$

$$= n^n/e^n \text{ sqrt}(2\pi n) \,/\, \left((n^n/2^n e^n)\, \pi n\right)$$

$$= 2^n \text{ sqrt}(2\pi n) \,/\, \pi n \quad \text{since the } n^n \text{ and } e^n \text{ terms cancel out}$$

$$= 2^n \text{ sqrt}(1/n) \text{ sqrt}(2/\pi)$$

$$= 2^n \, 2^{\lg(\text{sqrt}(1/n))} \, 2^{\lg(\text{sqrt}(2/\pi))}$$

$$\approx 2^n \, 2^{-\lg(n)/2} \, 2^{-0.326} \quad \text{noting } \lg(\text{sqrt}(2/\pi)) \approx -0.326$$

$$= 2^{n - \lg(n)/2 - 0.326}.$$

We know that when two $n/2$-element arrays L and R are merged to an n-element array A, $\binom{n}{n/2}$ outcomes are possible. (Each $n/2$ element S = $\{s_1,...,s_{n/2}\}$ subset of $\{1,...,n\}$ corresponds to unique output in which the elements from L end up in positions $s_1,...,s_{n/2}$ of A.) So the decision tree must have at least $\binom{n}{n/2} \approx 2^{n-\lg(n)/2-0.326}$ leaves, implying it must have height at least $\lceil \lg(2^{n-\lg(n)/2-0.326}) \rceil = \lceil n-\lg(n)/2-0.326 \rceil$. Any merging algorithm must perform at least $\lceil \boldsymbol{n-\lg(n)/2-0.326} \rceil$ **comparisons** in the worst case.

Note: By different arguments, we can obtain a lower bound of $n-1$ comparisons in the worst case, so except for a few very small values of n, the lower bound of $\lceil n-\lg(n)/2-0.326 \rceil$ comparisons cannot actually be achieved.

Exercise M There are 8 heaps of size 5, illustrated below.