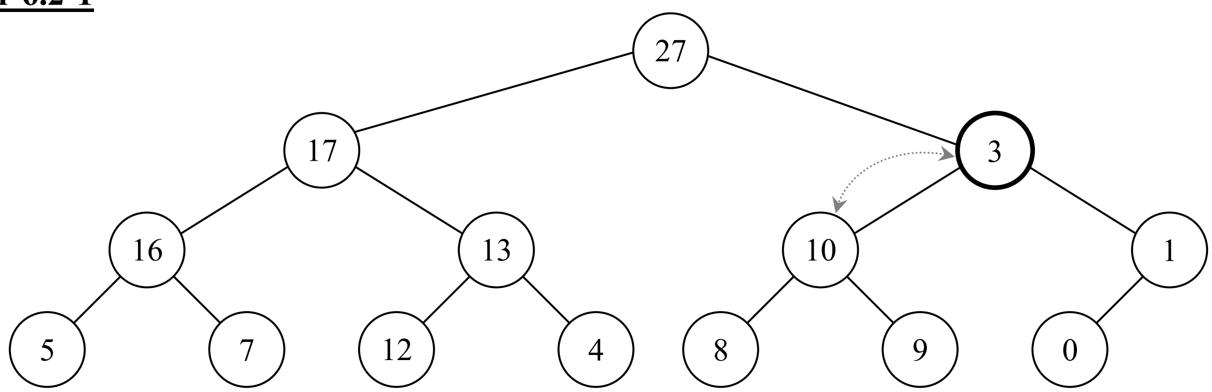
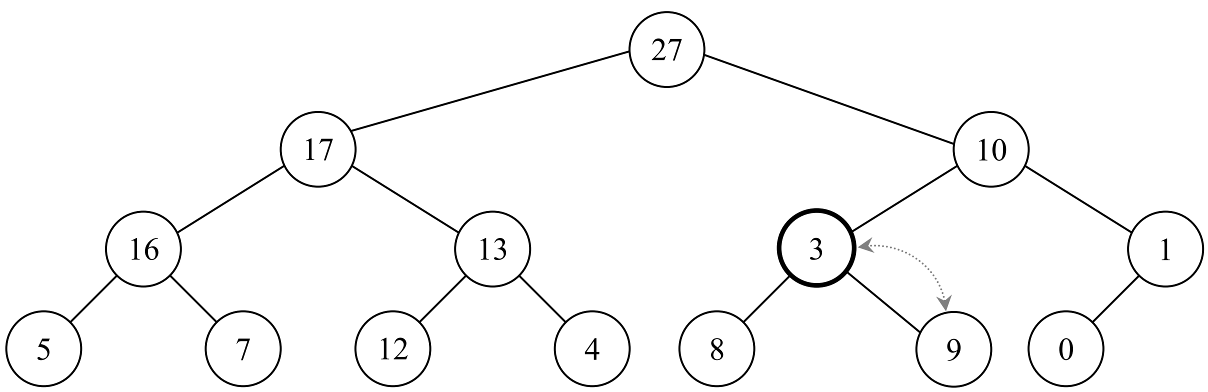
Solutions to CS/MCS 401 Exercise Set #3 (Summer 2007)

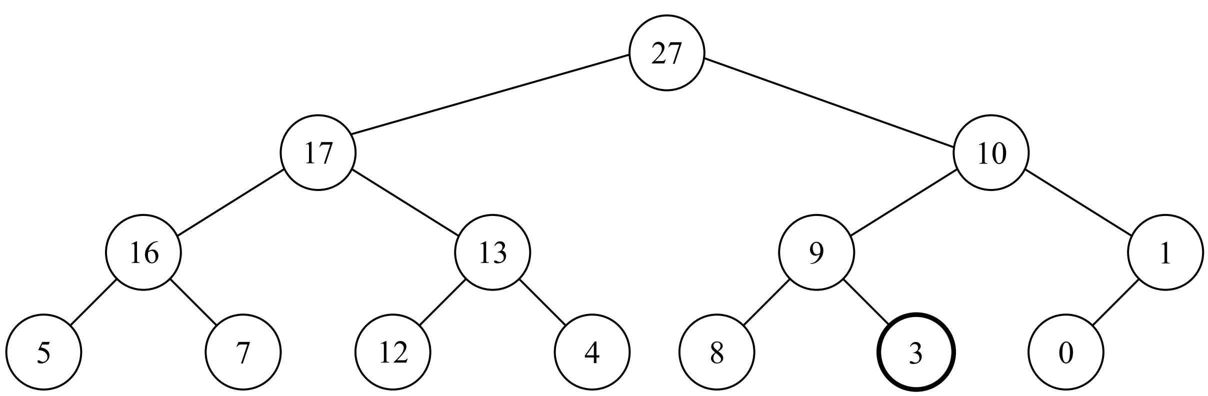
Exer 6.1-6

No. Note $a[4] = 6$ and $a[9] = 7$, so the element in node 4 fails to be greater than the element in its right child.

Exer 6.2-1

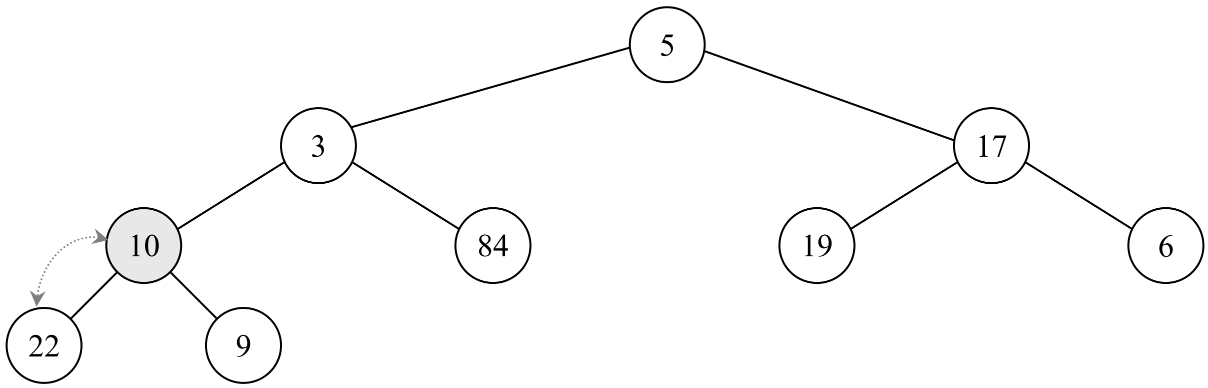


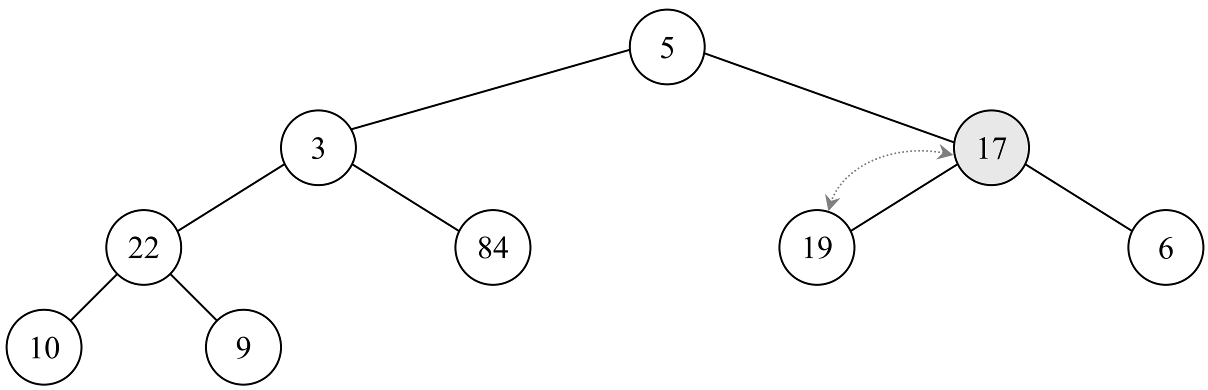


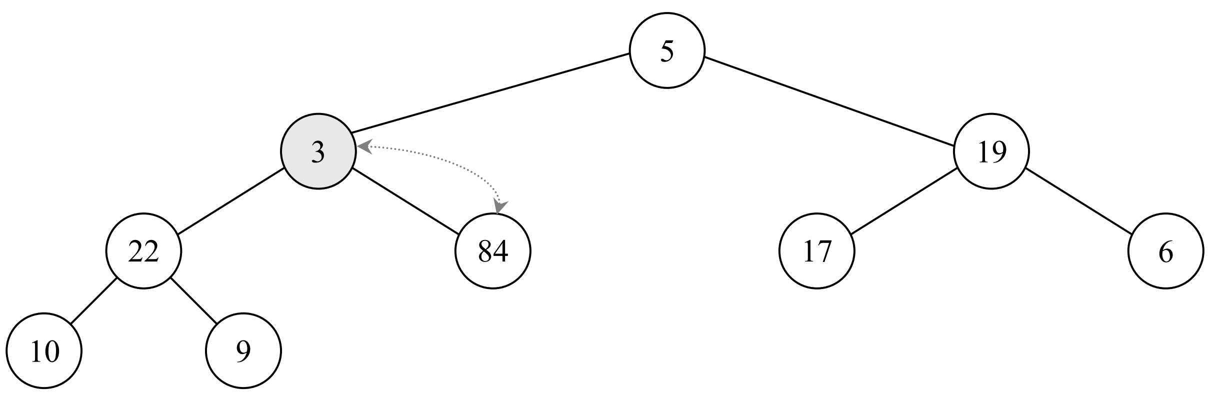


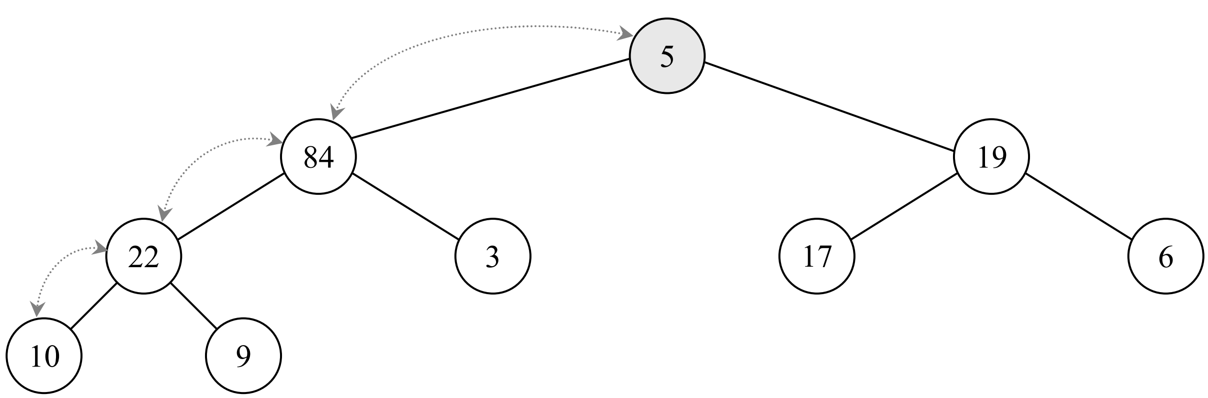
Exer 6.3-1

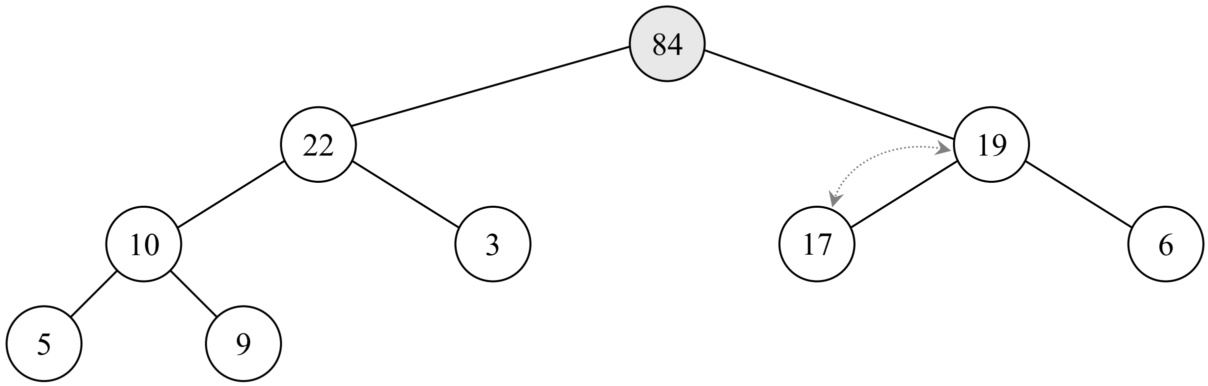
A = | 5 3 17 10 84 19 6 22 9 |











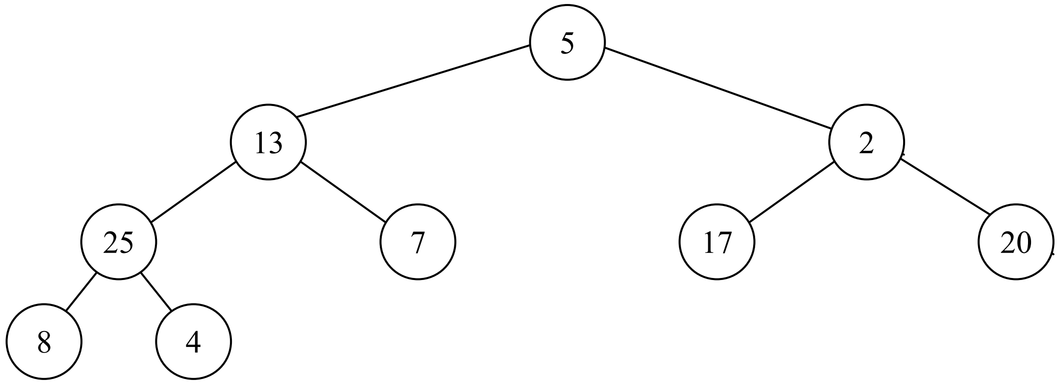
Exer 6.3-2

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.

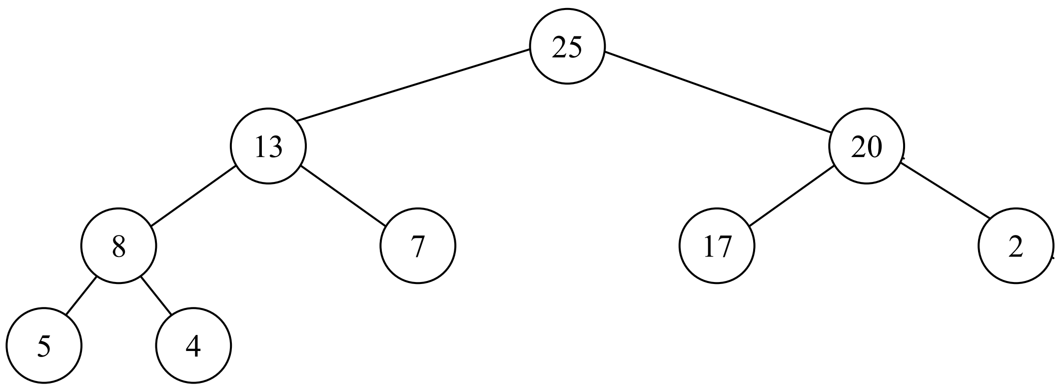
Exer 6.4-1

A = | 5 13 2 25 7 17 20 8 4 |

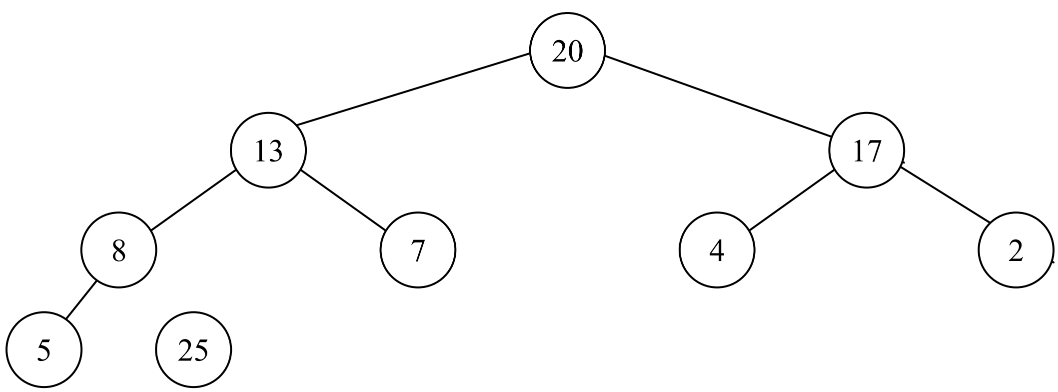
The diagrams below show the original array, the array after *build_Max_Heap*(), and the array after each step of converting the heap into a sorted array. Note you need show only the first three steps (first five diagrams).



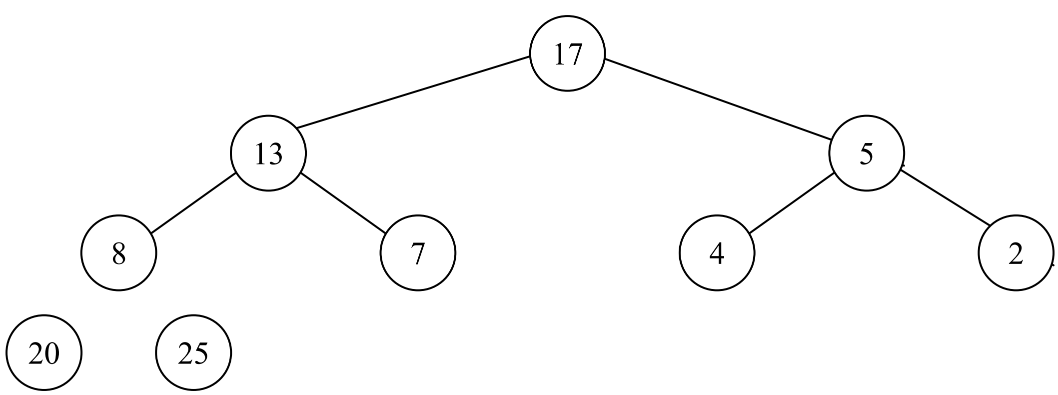
5
13
2
25
7
17
20
8
4



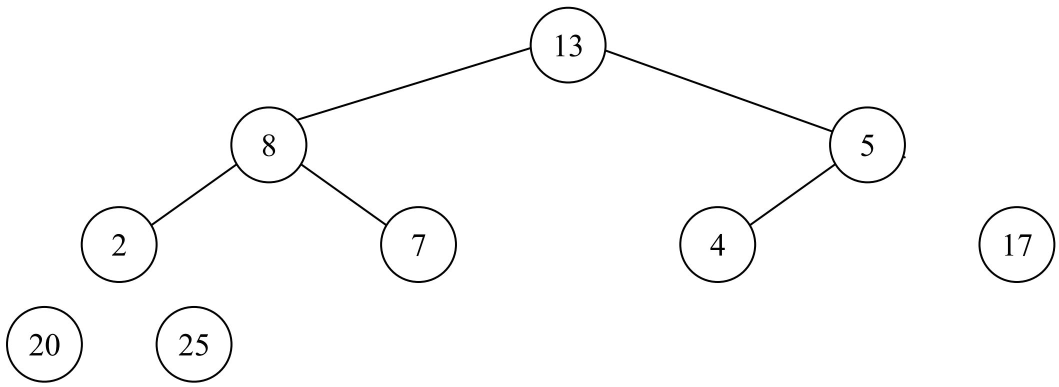
25
13
20
8
7
17
2
5
4

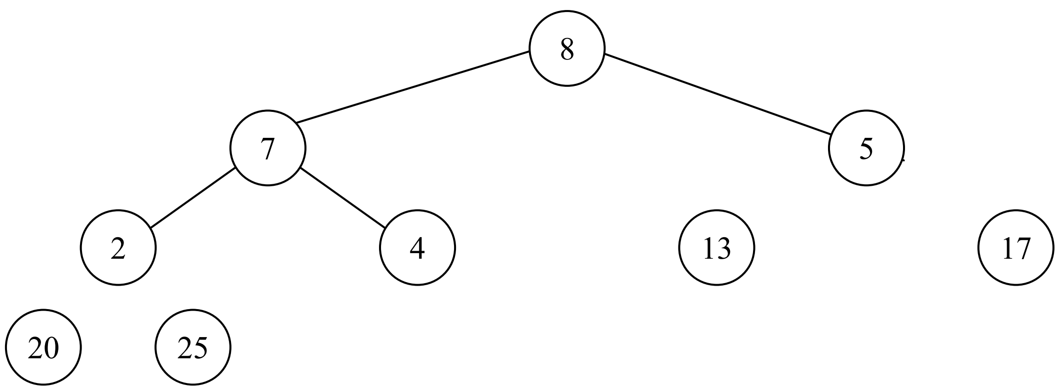


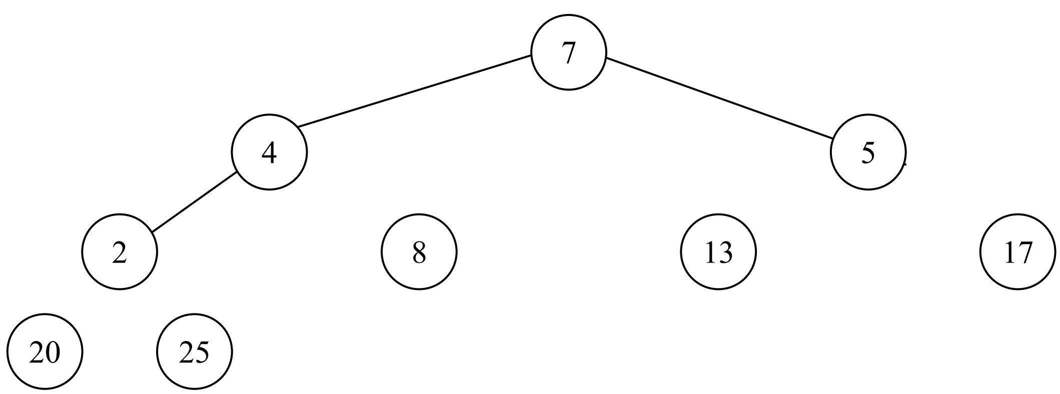
20
13
17
8
7
4
2
5
25

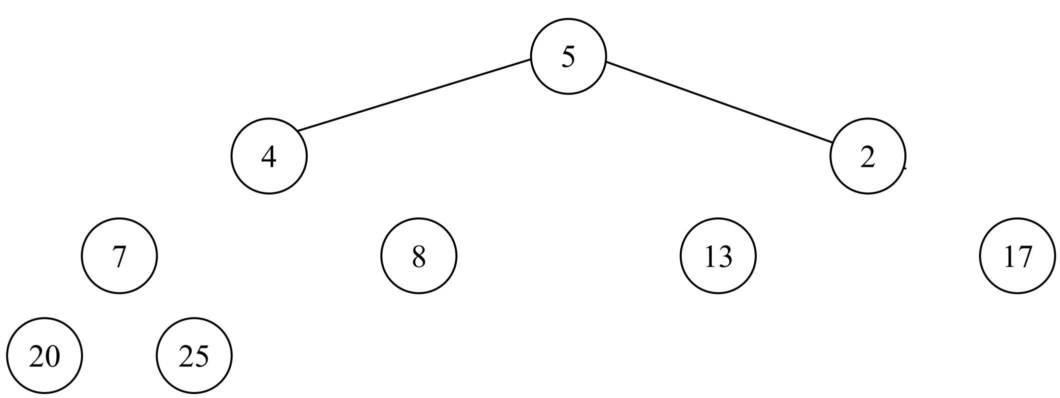


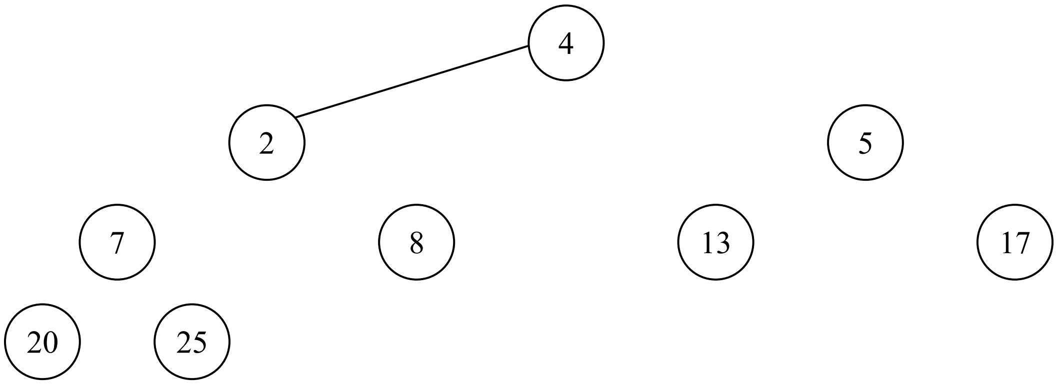
17
13
5
8
7
4
2
20
25

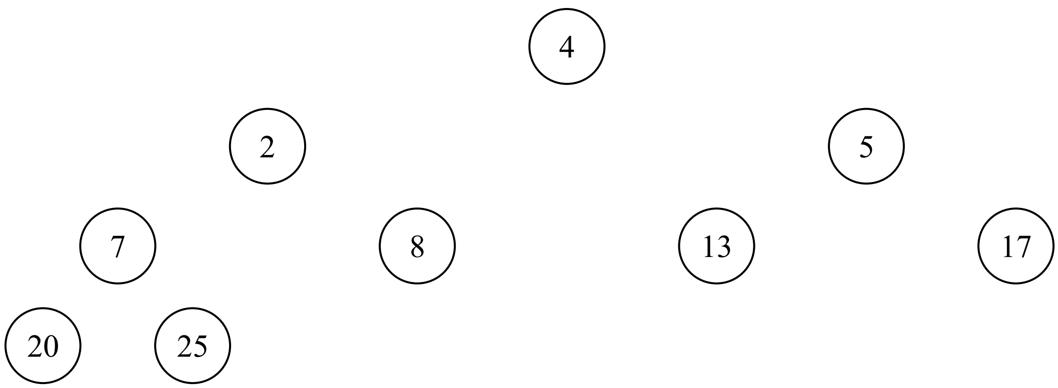

13
8
5
2
7
4
17
20
25


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





<u>6.5-7</u>

heap-delete(A, *i*, *n*);
  swap(A[*i*], A[*n*]);
  $n = n - 1$;
  while ($i > 1$ **and** A[*i*] > A[$\lfloor i/2 \rfloor$])
    swap(A[*i*], A[$\lfloor i/2 \rfloor$]);
    $i = \lfloor i/2 \rfloor$;
  while ($2i \le n$)
    if ($2i+1 \le n$ and A[2*i*+1] > A[2*i*])
      $p = 2i+1$;
    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 F. The inversions in the array **a** = (41, 16, 74, 33, 66, 54) are:

(41,16), (41,33), (74,33), (74,66), (74,54), (66,54).

The number of inversions is 6. Straight insertion sort would perform 6 comparisons in which it finds the elements out of order (1 for each inversion) and $n-1 = 5$ comparisons in which it finds the elements in order — a total of **11 comparisons**. Each comparison in which the elements are out of order is followed by an exchange, so there are **6 exchanges**.

Exersise G. Assume the recurrence

$$C(n) = C(0.8n) + C(0.5n) + C(0.2n) + n, \quad C(1) = 1$$

has a solution

$$C(n) = an^b + cn + d$$

for some real numbers a, b, c, and d. We ignore the fact that $0.2n$, $0.5n$, and $0.8n$ are not in general integers. Substituting the proposed solution into the recurrence, we obtain

$$\begin{aligned} an^b + cn + d \quad = \quad & a(0.2n)^b + c(0.2n) + d + \\ & a(0.5n)^b + c(0.5n) + d + \\ & a(0.8n)^b + c(0.8n) + d + \\ & + n \end{aligned}$$

or

$$a(1 - 0.2^b - 0.5^b - 0.8^b)n^b + (c - 0.2c - 0.5c - 0.8c - 1)n - 2d = 0.$$

Since this must hold for all n, the coefficients of n^b, n, and 1 each must equal 0.

$$a(1 - 0.2^b - 0.5^b - 0.8^b) = 0 \quad \Rightarrow \quad 1 - 0.2^b - 0.5^b - 0.8^b = 0 \quad (\text{since } a \neq 0),$$

$$c - 0.2c - 0.5c - 0.8c - 1 = 0 \quad \Rightarrow \quad c = -2,$$

$$2d = 0 \quad \Rightarrow \quad d = 0.$$

Now the condition $C(1) = 1$ implies $a + c + d = 1$, which together with the values of c and d above gives $a = 3$.

Let $f(x) = 1 - 0.2^x - 0.5^x - 0.8^x$. b is a root of $f(x)$. $f(x)$ is clearly an increasing function of x, since each of 0.2^x, 0.5^x, and 0.8^x decrease as x increases. $f(1) = -0.5$ and $f(2) = 0.07$. Thus $f(x)$ has a unique root (equal to b), which must lie between 1 and 2 (probably closer to 2), and it can be approximated by bisection or Newton's method. With an initial guess of 2, Newton's method with 4 iterations gives $b = 1.8267247$, or to two decimal places $b \approx 1.83$.

Note: The complete solution is $C(n) = 3\,n^{1.83} - 2n$. We saw earlier that division into 3 equal-sized subproblems of total size $1.5n$, with linear divide/combine time, led to a $\Theta(n^{1.59})$ time algorithm. Here the division into three unequal-sized subproblems with the same total size raises the time to $\Theta(n^{1.82})$.

Exercise H.

Let L, M, and R be sorted arrays of length $n/3$ (possibly $\lfloor n/3 \rfloor$ or $\lceil n/3 \rceil$, so the sum of the lengths is n). Assume that each array has an extra element ∞ at the end. We can merge L, M, and R into a single sorted array A of length n using the algorithm below. Here i, j, and k

represent the positions of the current elements in L, M, and R respectively; and x represents the smallest element not yet merged from M or R, provided $xValid$ is true. As usual, indentation indicates nesting of blocks.

```
i = 1;  j = 1;  k = 1;
xValid = false;
for ( q = 1,2, ..., n )
    if ( not xValid )
        if ( M[j] ≤ R[k] )                (*)
            x = M[j];
            j = j + 1;
        else
            x = R[k];
            k = k + 1;
        xValid = true;
    if ( L[i] ≤ x )                       (**)
        A[q] = L[i];
        i = i + 1;
    else
        A[q] = x;
        xValid = false;
```

Comparisons are performed in the lines (*) and (**). The comparison in line (**) is performed on each pass through the loop — a total of n times. The comparison on line (*) is performed on the first pass; on the remaining passes, it is performed unless the element merged to A on the previous pass came from L — a total of

$$n - (\text{number of elements merged from L on the first } n{-}1 \text{ passes})$$

times. Thus the comparison on line (*) is performed $n-\text{length(L)}$ or $n-(\text{length(L)}-1)$ times. Since L has length $n/3$, the number of comparisons on line (*) is $2n/3$ or $2n/3 + 1$. The total number of comparisons is $5n/3$ or $5n/3 + 1$.

__Exercise I.__

$C(n) = 3C(n/3) + 5/3n$, $C(1) = 0$. We assume $n = 3^k$, so $k = \log_3(n)$.

This recurrence has the correct form for the Master Theorem with $a = 3$, $b = 3$, $E = 1$, $n^E = n$, $f(\mathrm{n}) = 5/3n$. However, the Master Theorem tells us only that the solution is $\Theta(n\log_b(n))$, whereas we are asked for an exact solution. In the proof of the Master Theorem, we showed that

$$C(n) = f(n) + af(n/b) + a^2f(n/b^2) + ... + a^{k-1}f(n/b^{k-1}) + a^kd, \text{ where } d = C(1).$$

Substituting the appropriate values for a, b, $f(n)$, and d, we obtain

$$C(n) = 5/3n + 3(5/3)(n/3) + 3^2(5/3)(n/3^2) + ... + 3^{k-1}(5/3)(n/3^{k-1}).$$

This sum contains k terms, each equal to $5/3n$, so the sum is $5/3nk = 5/3n\log_3(n)$.

The exact solution when n is a power of 3 is $\mathbf{C(n) = 5/3\,n\log_3(n) \approx 1.052\,n\lg(n)}$.