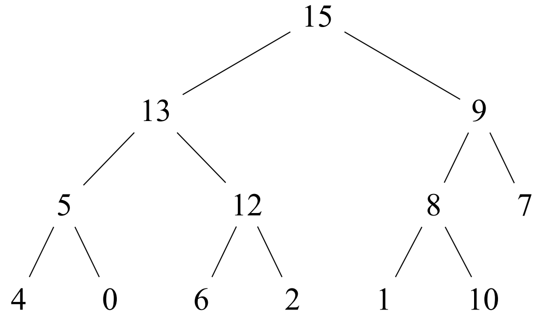
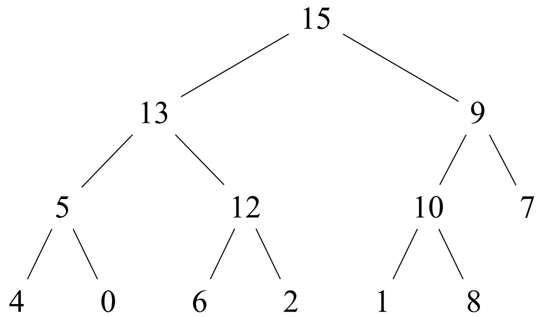
Problem 1

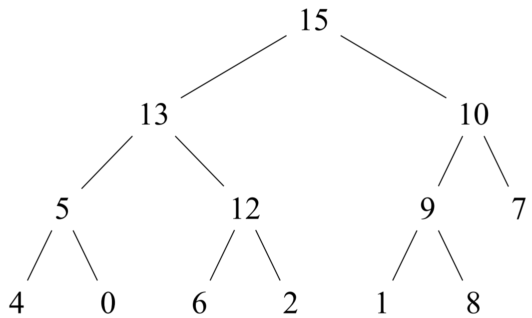
1. Increase Heap Size
   $A = \{15, 13, 9, 5, 12, 8, 7, 4, 0, 6, 2, 1, 10\}$



2. `Max-Heap-Increase-Key`
   - Array after first swap: $A = \{15, 13, 9, 5, 12, 10, 7, 4, 0, 6, 2, 1, 8\}$



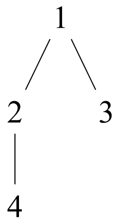
   - Array after second swap: $A = \{15, 13, 10, 5, 12, 9, 7, 4, 0, 6, 2, 1, 8\}$



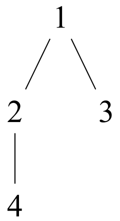
Problem 2

1. They do not always create the same heap. Let $A = \{1, 2, 3, 4\}$
   - `Build-Max-Heap`: $A = \{4, 1, 3, 2\}$

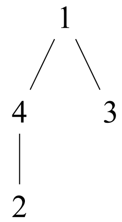
     (a) `Max-Heapify` $(A, 4)$

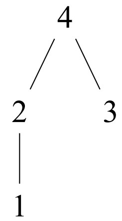
     (b) `Max-Heapify` $(A, 3)$

     (c) `Max-Heapify` $(A, 2)$

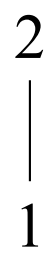
     

     (d) `Max-Heapify` $(A, 1)$

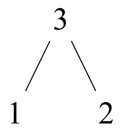
     

   - `Build-Max-Heap'`: $A = \{4, 3, 2, 1\}$

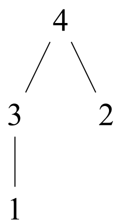
     (a) `Max-Heap-Insert` $(A, A[2])$

     (b) `Max-Heap-Insert` $(A, A[3])$

     (c) `Max-Heap-Insert` $(A, A[4])$

2. (a) Each `Max-Heap-Insert` operation takes $\Theta(\log k)$ time, where k is the current heap size.
   (b) The worst case happens when the array is sorted, and the `Max-Heap-Insert` operation will get called $n - 1$ times.
   (c) Each time it should pull the element to the beginning of the heap, and the total time $T(n) = \sum\limits_{k=2}^{n} \log k = \log(n!) = \Theta(n \log n)$

Problem 3

Since constructing a binary search tree from an unordered list involves the same comparison requirements as sorting, it must take $\Omega(n \log n)$ time in the worst case. This is a fundamental lower bound for any comparison-based algorithm that constructs a BST.

Problem 4

The black-height of a node x, denoted $bh(x)$, is the number of black nodes on any path from x. The shortest path from x to a descendant leaf is a path where red and black nodes alternate as much as possible. This shortest path contains only $bh(x)$ black nodes and at least $bh(x)$ nodes in total. The longest path occurs when every black node is followed by a red node, doubling the path length. Therefore, the longest path from x to a descendant leaf has length at most $2 \times bh(x)$.

Problem 5

- Base case
  For $n = 1$, the matrix is $\begin{pmatrix}1\end{pmatrix}$, so the determinant is 1.

- Inductive Step
  Assume the statement holds for $n-1$, we will prove it for n.

  1. For $i = n-1$ down to 1, replace column $i+1$ with $col(i+1) - x_0 \times col(i)$. We will get

$$A = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & x_1 - x_0 & x_1(x_1 - x_0) & \dots & x_1^{n-2}(x_1 - x_0) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1} - x_0 & x_{n-1}(x_{n-1} - x_0) & \dots & x_{n-1}^{n-2}(x_{n-1} - x_0) \end{pmatrix}$$

  2. Since the first row has zeros except for the first element, we can expand along the first row

$$\det(V) = 1 \times \det(V_1')$$

  where the matrix V_1' is

$$V_1' = \begin{pmatrix} x_1 - x_0 & x_1(x_1 - x_0) & \cdots & x_1^{n-2}(x_1 - x_0) \\ x_2 - x_0 & x_2(x_2 - x_0) & \cdots & x_2^{n-2}(x_2 - x_0) \\ \vdots & \vdots & \ddots & \vdots \\ x_{n-1} - x_0 & x_{n-1}(x_{n-1} - x_0) & \cdots & x_{n-1}^{n-2}(x_{n-1} - x_0) \end{pmatrix}$$

  From each row i, we factor out $(x_i - x_0)$

$$\det(V_1') = \left(\prod_{i=1}^{n-1}(x_i - x_0)\right)\det(W)$$

  where W is the matrix

$$W = \begin{pmatrix} 1 & x_1 & \cdots & x_1^{n-2} \\ 1 & x_2 & \cdots & x_2^{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1} & \cdots & x_{n-1}^{n-2} \end{pmatrix}$$

  3. By the inductive hypothesis, we have

$$\det(W) = \prod_{1 \le j < k \le n-1}(x_k - x_j)$$

  Therefore

$$\det(V) = \prod_{i=1}^{n-1}(x_i - x_0) \times \prod_{1 \le j < k \le n-1}(x_k - x_j) = \prod_{0 \le j < k \le n-1}(x_k - x_j)$$