TA: Jade Cheng

ICS 241

Recitation Lecture Notes #11

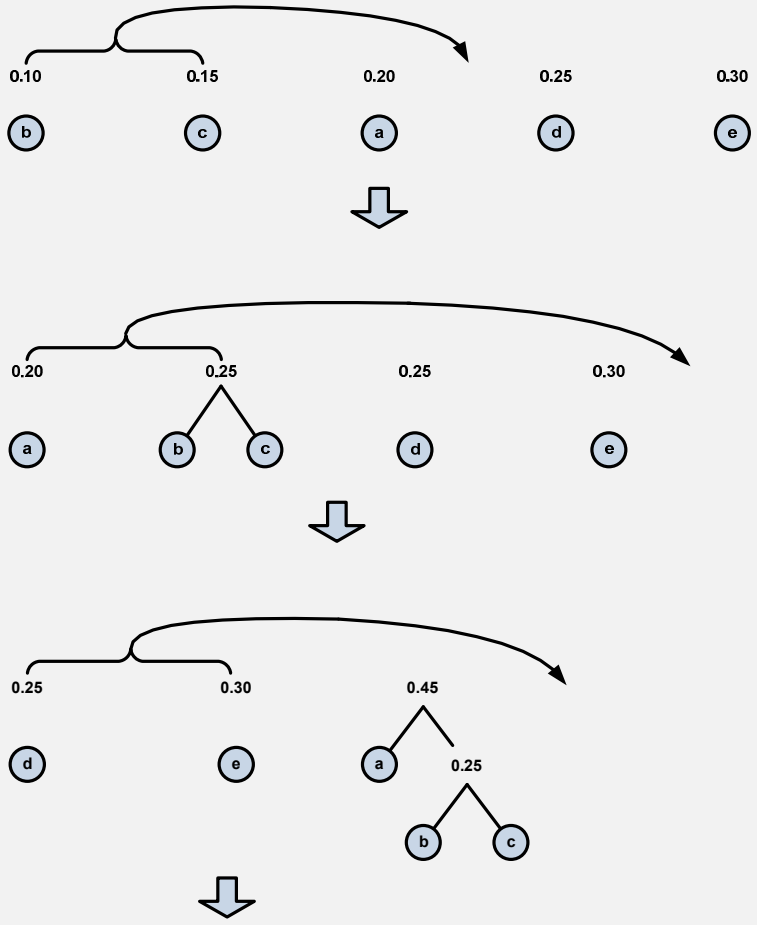
November 06, 2009

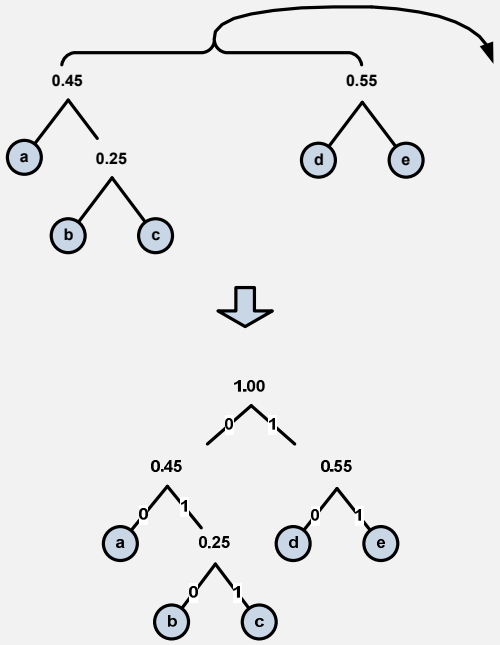
Recitation #11

Question: Use Huffman coding to encode these symbols with given frequencies: a: 0.20, b: 0.10, c: 0.15, d: 0.25, e: 0.30. [Chapter 10.2 Review]

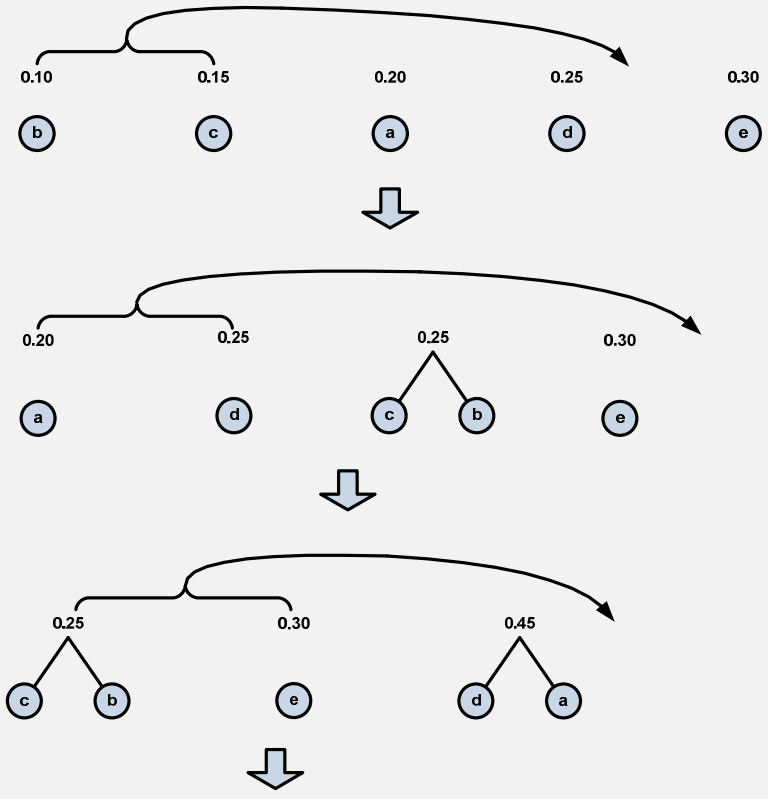
a. Construct two different Huffman codes for these given symbols.

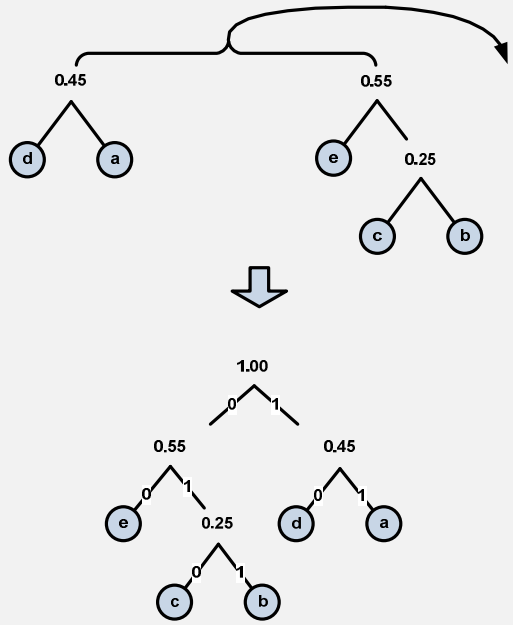
Answer: Since the orders for same frequency parts can be different; the insertion can be different based on implementation, there is more than one way to construct Huffman codes for a particular set of symbols. In this example, one way of doing it is:





So, the Huffman codes for these symbols with the given frequencies can be $a: 00, b: 010, c: 011, d: 10, e: 11$. We can also implement an insertion that inserts after the nodes with the same values. Also, we can group the node from right to left instead of left to right. There is another way of doing it:





We get a different set of encoding this time and they are: a: 11, b: 011, c: 010, d: 10, e: 00. As we saw the shapes of these two Huffman trees happen to be the same.

b. What is the average number of bits required to encode a character?

Answer: Let's use the first way of encoding as an example, a: 00, b: 010, c: 011, d: 10, e: 11. Let's call the average number of bits $n_{average}$, the frequencies of characters f_a, f_b, f_c, f_d, f_e.

$$\begin{aligned} n_{average} &= n_a \cdot f_a + n_b \cdot f_b + n_c \cdot f_c + n_d \cdot f_d + n_e \cdot f_e \\ &= 2 \cdot 0.20 + 3 \cdot 0.10 + 3 \cdot 0.15 + 2 \cdot 0.25 + 2 \cdot 0.30 \\ &= 0.40 + 0.30 + 0.45 + 0.5 + 0.6 \\ &= 2.25\,. \end{aligned}$$

Now let's look at the second way of encoding, a: 11, b: 011, c: 010, d: 10, e: 00. They length of encoding each character is the same as the first way and the average $n_{average}$ bit required to encode a letter is therefore the same 2.25.

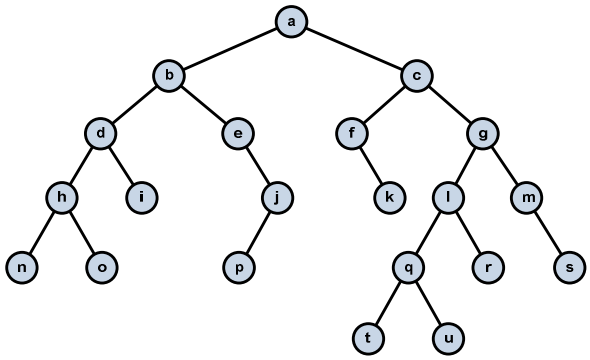
c. How much savings does the Huffman code provide on average compared to a fixed-length encoding on these symbols?

Answer: We have computed the average number of bits needed to encode these symbols using Huffman encoding. Let's consider the number of bits needed to encode these symbols using fixed-length encoding. We have five different symbols, hence we need at least 3 bits long codes. Because $2^3 - 1 = 7 \geq 5$. So we would be using 3 bits per character for fixed-length encoding.

$$n_{fixed\_length} = n_a \cdot f_a + n_b \cdot f_b + n_c \cdot f_c + n_d \cdot f_d + n_e \cdot f_e$$
$$= 3 \cdot 0.20 + 3 \cdot 0.10 + 3 \cdot 0.15 + 3 \cdot 0.25 + 3 \cdot 0.30$$
$$= 0.60 + 0.30 + 0.45 + 0.75 + 0.9$$
$$= 3\,.$$

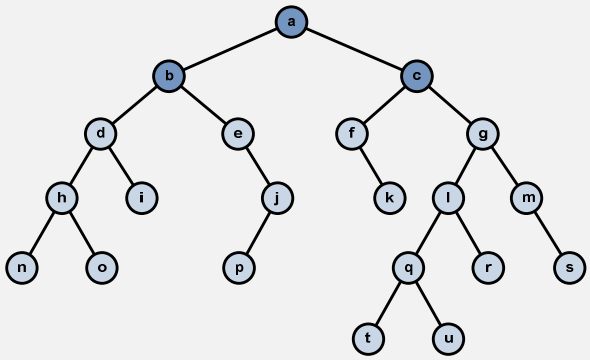
Obviously, the frequencies in this case don't do anything. The size of encoding for each character is 3 all the time. So, we save $(3 - 2.25)/3 = 25\%$ on average.

Question: Determine the order of visits the vertices of the given ordered rooted tree. [Chapter 10.3 Review]

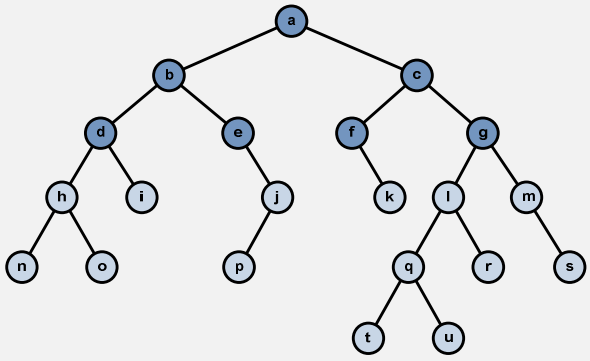


a. Preorder traversal

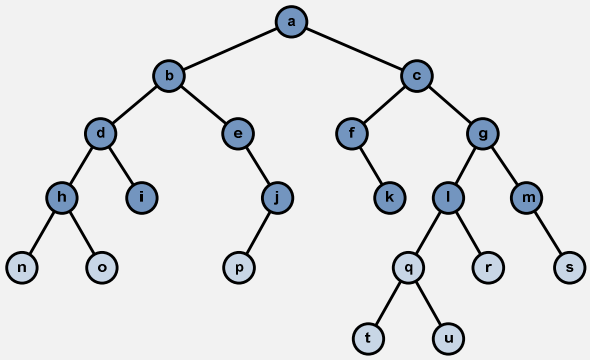
Answer: Step 1: Starting from the root : a, b, c



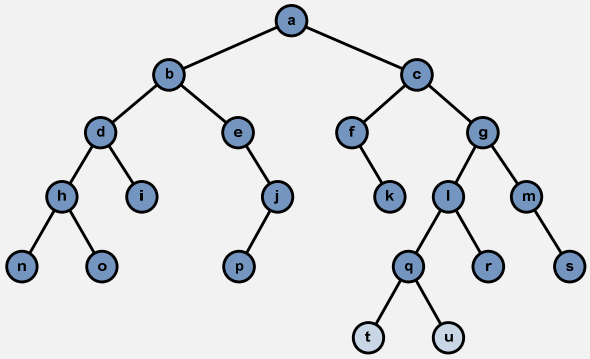
Step 2: Exam every nodes on level 1: a, b, d, e, c, f, g



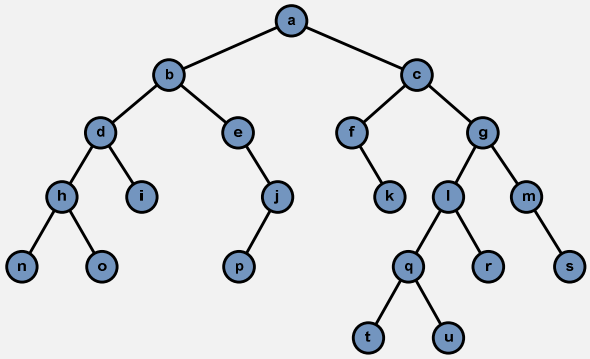
Step 3: Exam every nodes on level 2: $a, b, d, h, i, e, j, c, f, k, g, l, m$



Step 4: Exam every nodes on level 3: $a, b, d, h, n, o, i, e, j, p, c, f, k, g, l, q, r, m, s$

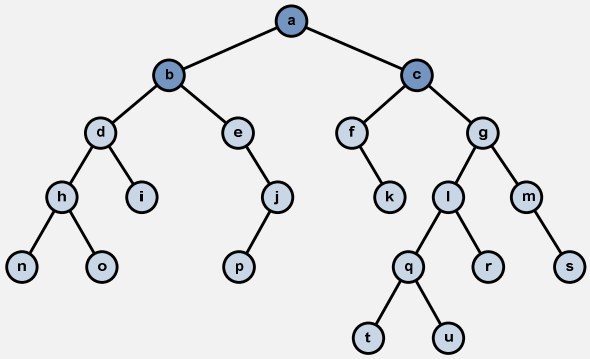


Step 4: Exam every nodes on level 4: $a, b, d, h, n, o, i, e, j, p, c, f, k, g, l, q, t, u, r, m, s$

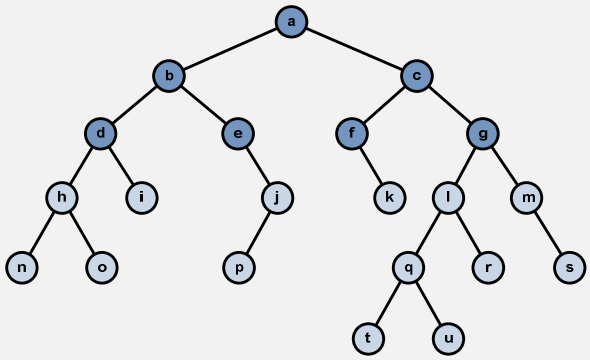


b. Postorder traversal

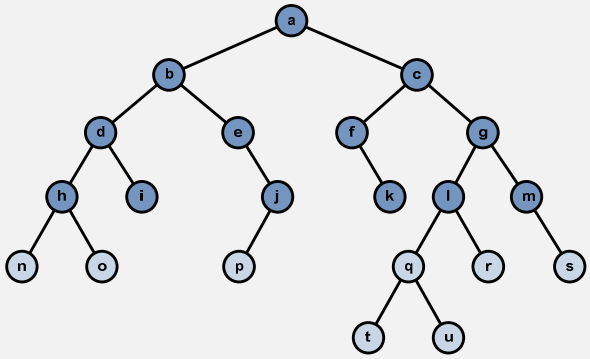
Answer: Step 1: Starting from the root : b, c, a



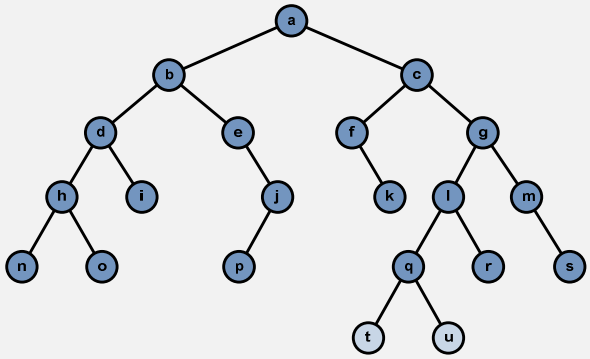
Step 2: Exam every nodes on level 1: d, e, b, f, g, c, a



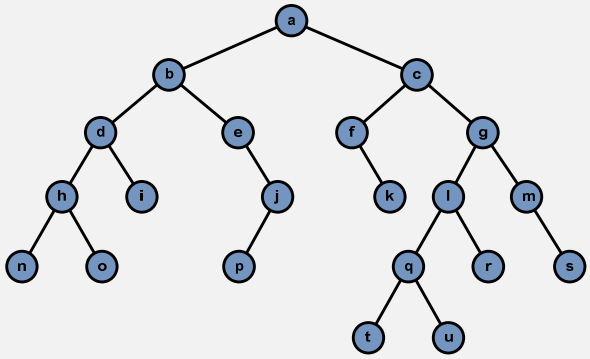
Step 3: Exam every nodes on level 2: $h, i, d, j, e, b, k, f, l, m, g, c, a$



Step 4: Exam every nodes on level 3: $n, o, h, i, d, p, j, e, b, k, f, q, r, l, s, m, g, c, a$

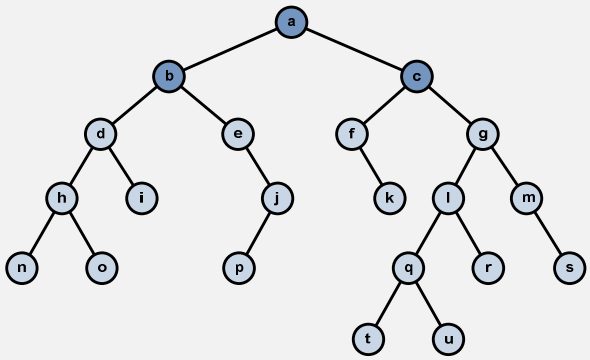


Step 4: Exam every nodes on level 4: $n, o, h, i, d, p, j, e, b, k, f, t, u, q, r, l, s, m, g, c, a$

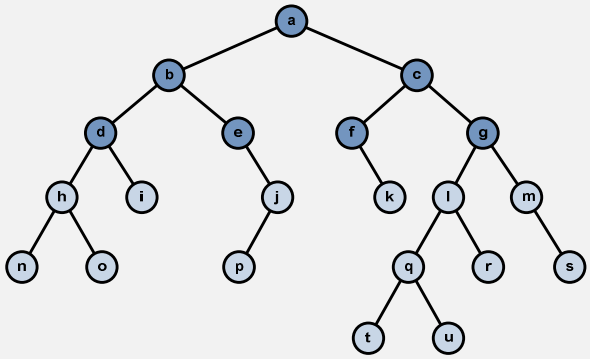


c. Inorder traversal

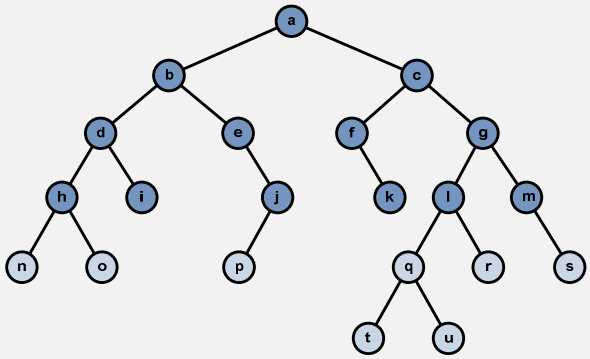
Answer: Step 1: Starting from the root : b, a, c



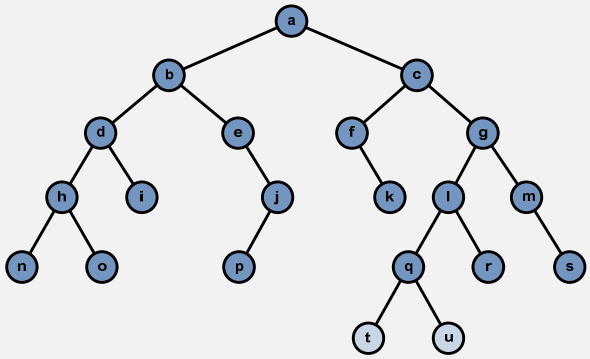
Step 2: Exam every nodes on level 1: d, b, e, a, f, c, g



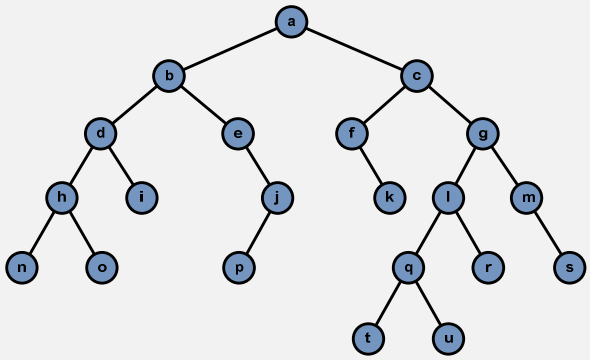
Step 3: Exam every nodes on level 2: $h, d, i, b, e, j, a, f, k, c, l, g, m$



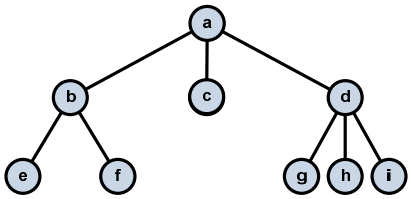
Step 4: Exam every nodes on level 3: $n, h, o, d, i, b, e, p, j, a, f, k, c, q, l, r, g, m, s$



Step 4: Exam every nodes on level 4: $n, h, o, d, i, b, e, p, j, a, f, k, c, t, q, u, l, r, g, m, s$



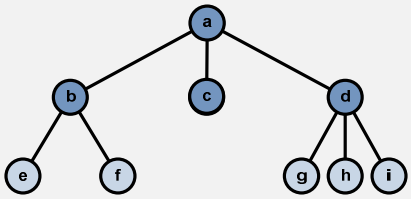
Question: Inorder traverse the following tree. [Chapter 10.3 Review]



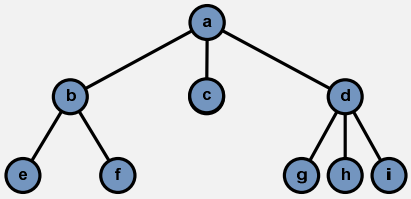
Review: There's no confusing about poastorder and preorder, since the parent goes either at the end of at the beginning and the children nodes get listed out from left to right one by one. But the inorder tree traversal might be confusing. We would need to know where in the middle should we place the parent. According to the inorder traversal algorithm provided on the book, we place the parent right after the first child. In other words, only the left most child goes before the parent.

```
procedure Inorder(T: ordered rooted tree)
        r := root[T]
        if leaf[r]
                then list r
        else
        begin
                l := first child of r from left to right
                T(l) := subtree with l as its root
                Inorder(T(l))                                   // list the first child
                list r                                          // list the parent
                for each child c ≠ l of r from left to right
                        T(c) := subtree with c as its root      // list other children
                        Inorder(T(c))
        end
```

Answer: Step 1: Starting from the root : b, a, c, d



Step 2: Exam every nodes on level 1: $e, b, f, a, c, g, d, h, i$

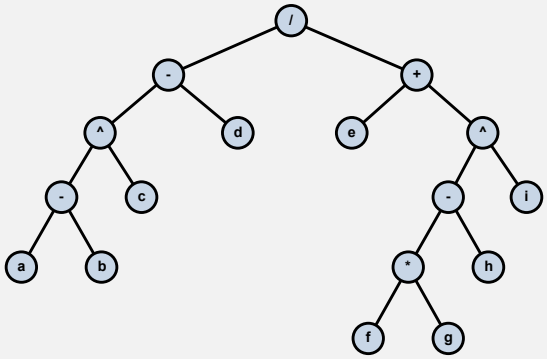


Question: Construct the parsing tree and write the algebraic expression below in the following different notations. [Chapter 10.3 Review]

$$\frac{(a-b)^c - d}{e + (f * g - h)^i}$$

a. Prefix notation.

Answer: Construct the parsing tree, which will be used in all three questions.



Now we can traverse the parse tree following the preorder tree traversal algorithm to obtain the prefix notation of the given algebraic expression

```
/ - ^ - a b c d + e ^ - * f g h i
```

b. Postfix notation.

Answer: Traverse the parse tree following the postorder tree traversal algorithm to obtain the postfix notation of the given algebraic expression

```
a b - c ^ d – e f g * h – i ^ + /
```

c. Infix notation.

Answer: Traverse the parse tree following the postorder tree traversal algorithm to obtain the postfix notation of the given algebraic expression

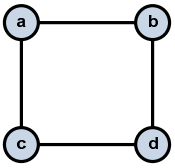
```
a - b ^ c - d / e + f * g - h ^ i
```

Question: How many edges must be removed from a connected graph with n vertices and m edges to produce a spanning tree? [Chapter 10.4 Review]

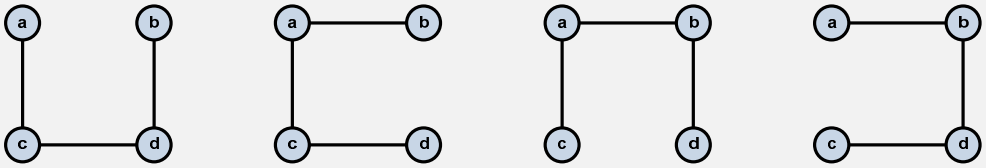
Review: *Theorem 2 of Tree Properties:* A tree with n vertices has $n - 1$ edges

Answer: Converting a connected graph to a spanning tree reduce the edges from the m to $n - 1$. Therefore the edges to remove to produce a spanning tree is $m - (n - 1) = m - n + 1$.

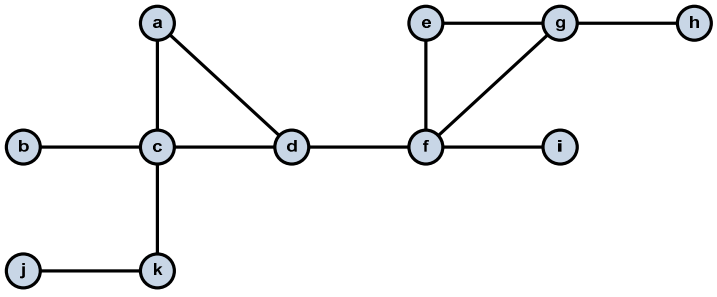
Question: Look for all the spanning trees for the following simple connected graph. [Chapter 10.4 Review]



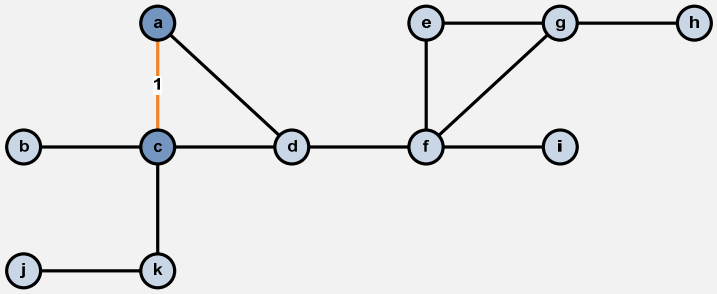
Answer: There are four different spanning trees we can construct from this graph.



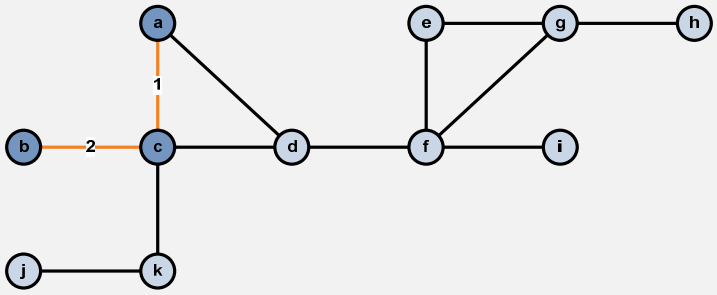
Question: Use a depth-first search to find a spanning tree for the following graph. Use vertex a as the root and use alphabetical ordering to determine in which order to visit the vertices. [Chapter 10.4 Review]



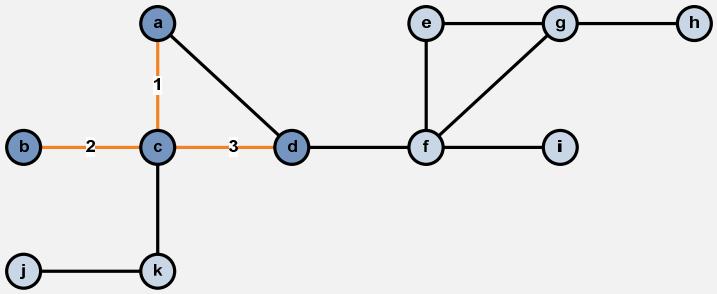
Answer: Step 1. Starting from the root a, $a \to c$.



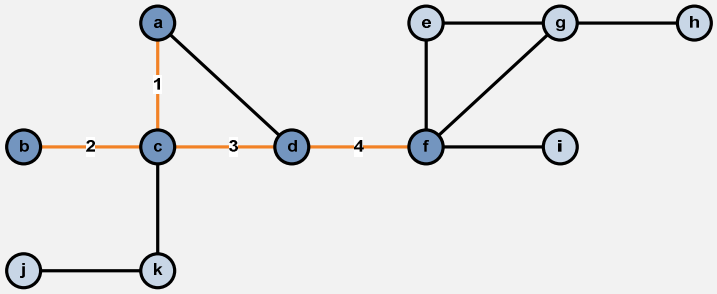
Step 2. $c \to b$



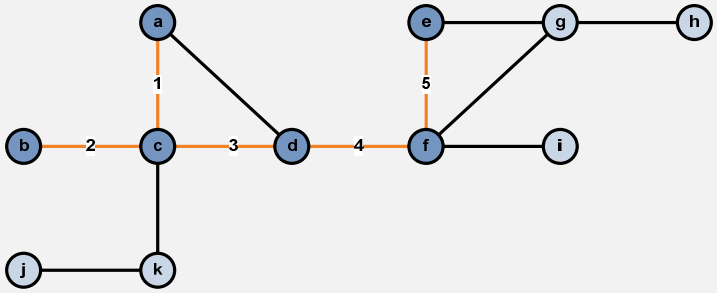
Step 3. Back track $b \to c$, and then $c \to d$



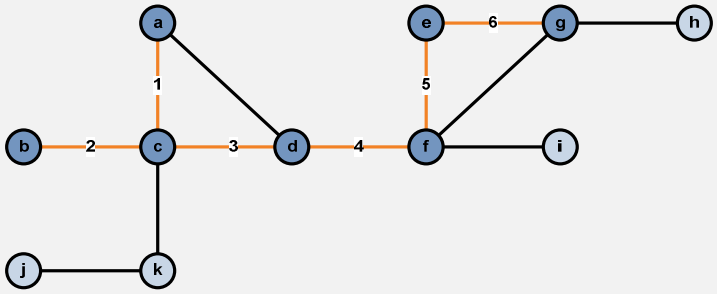
Step 4. $d \to f$



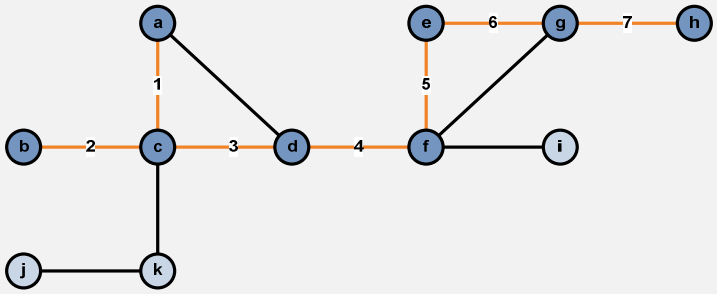
Step 5. $f \to e$



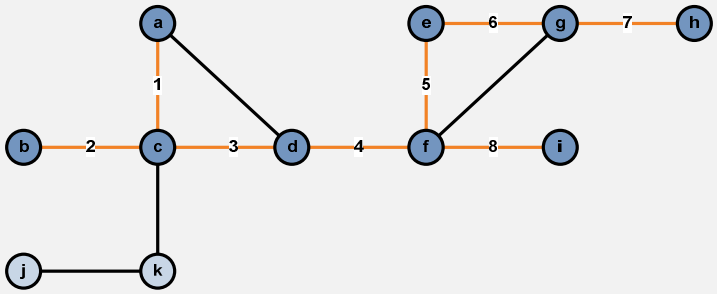
Step 6. $e \to g$



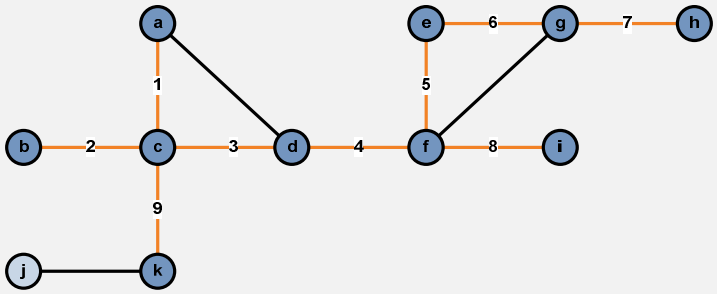
Step 7. $g \to h$



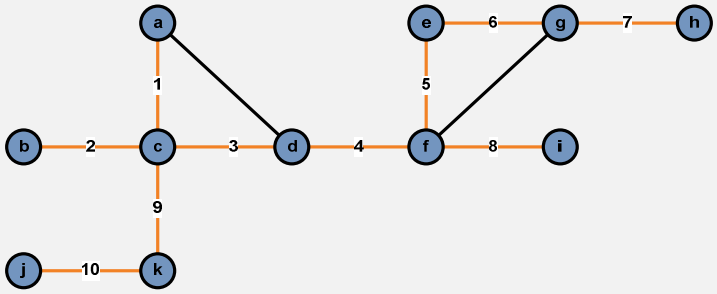
Step 8. Back track $h \to g, g \to e, e \to f$, and then $f \to i$



Step 9. Back track $i \to f, f \to d, d \to c$, and then $c \to k$

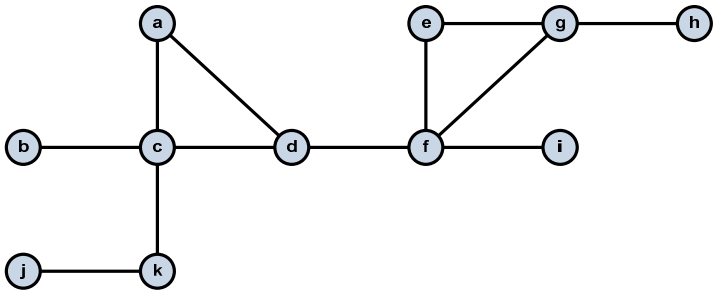


Step 9. $k \to j$

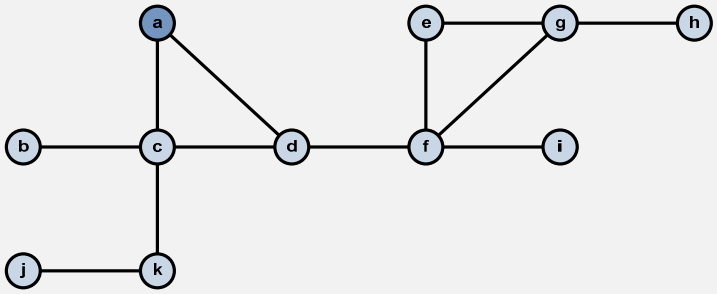


At this point, we've traversed the entire tree and visited all nodes. Algorithm terminates and we obtain the order of nodes we are going to visit using depth-first search. The order is $a, b, d, f, e, g, h, i, k, j$.

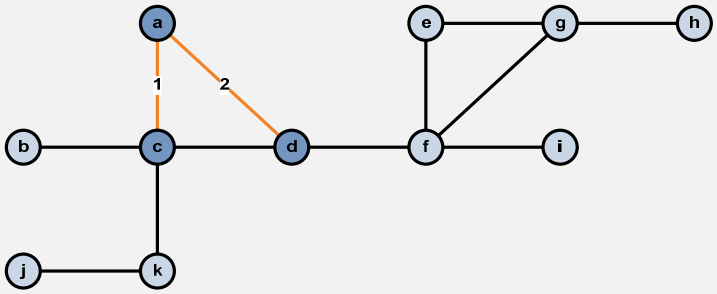
Question: Use a breadth-first search to find a spanning tree for the following graph. Use vertex a as the root and use alphabetical ordering to determine in which order to visit the vertices. [Chapter 10.4 Review]



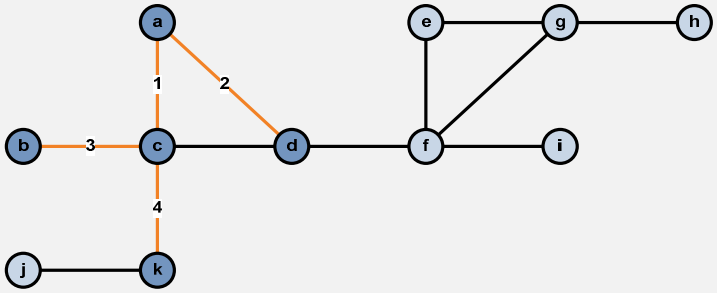
Answer: Step 1. Starting from the root a.



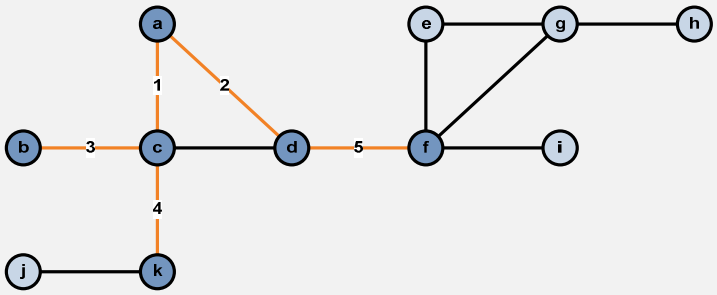
Step 2. Explore all paths from a, $a \to c, a \to d$



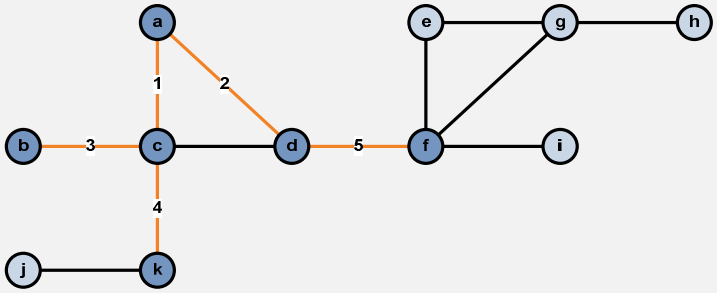
Step 3. Explore all paths from c, $c \to b, c \to k$



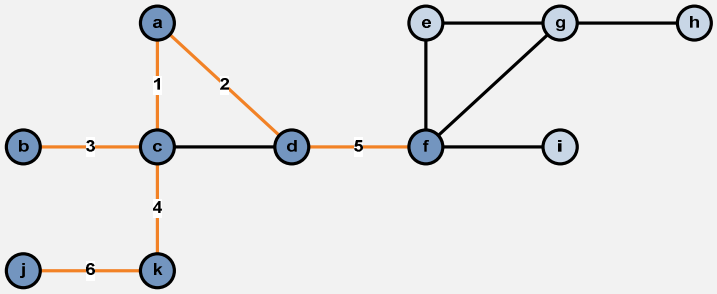
Step 4. Explore all paths from d, $d \to f$



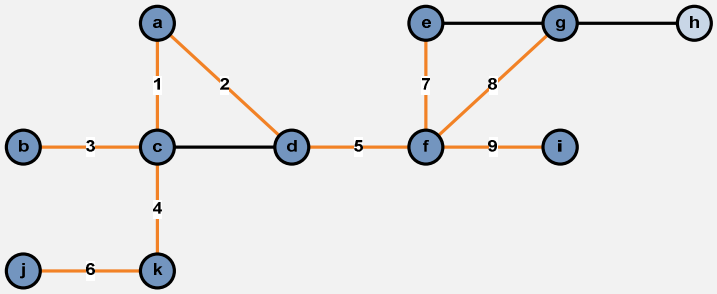
Step 5. Explore all paths from b, no new path. Node b is a leaf node.



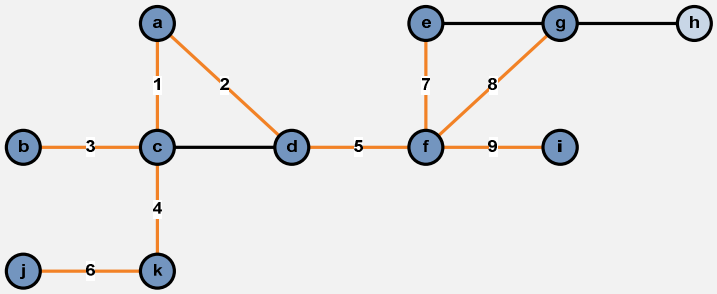
Step 6. Explore all paths from k, $k \to j$



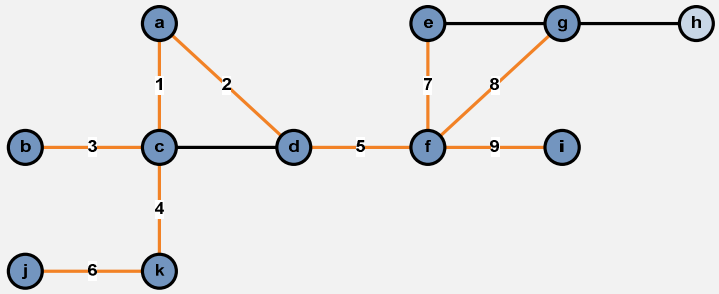
Step 7. Explore all paths from f, $f \to e, f \to g, f \to i$



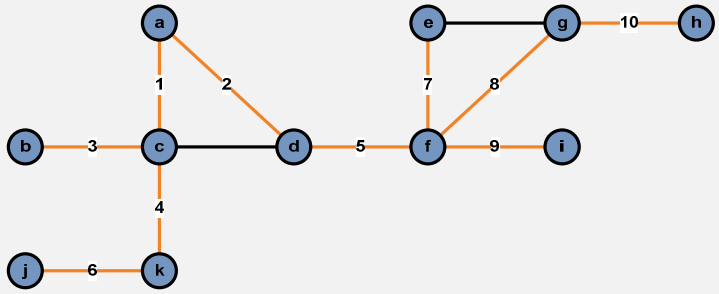
Step 8. Explore all paths from j, no new path. Node j is a leaf node



Step 9. Explore all paths from e, no new path. Node e is a leaf node



Step 10. Explore all paths from g, $g \rightarrow h$



At this point, we've traversed the entire tree and visited all nodes. Algorithm terminates and we obtain the order of nodes we are going to visit using breadth-first search. The order is $a, c, d, b, k, f, j, e, g, i, h$.