Keshav Narang

September 15, 2021

Data Structures and Algorithms in Java

**Assignment 9 - Symbol Tables and BSTs**

1. Give a trace of the process of inserting the keys E A S Y Q U E S T I O N into an initially empty table using SequentialSearchST. How many compares are involved?

   In a sequential search symbol table, every element is compared with all the keys linearly until it finds itself or reaches the end of the table. In this example, there are:

   0 + 1 + 2 + 3 + 4 + 5 + 1 + 3 + 6 + 7 + 8 + 9 comparisons = 49 comparisons.

   a. Queue: EASYQUESTION

      0 Comparisons: E has nothing to compare with in the Symbol Table

      Symbol Table:

| **Key** | E |
|---|---|
| **Value** | 0 |

   b. Queue: ASYQUESTION

      1 Comparison: A is compared with the E element in the Symbol Table

      Symbol Table:

| **Key** | E | A |
|---|---|---|
| **Value** | 0 | 1 |

   c. Queue: SYQUESTION

2 Comparisons: S is compared with E and then A in the Symbol Table

Symbol Table:

| **Key** | E | A | S |
|---|---|---|---|
| **Value** | 0 | 1 | 2 |

d. Queue: YQUESTION

3 Comparisons: Y is first compared with E, then A, then S, in the Symbol Table

Symbol Table:

| **Key** | E | A | S | Y |
|---|---|---|---|---|
| **Value** | 0 | 1 | 2 | 3 |

e. Queue: QUESTION

4 Comparisons: Q is compared with E, A, S, and Y in the Symbol Table

Symbol Table:

| **Key** | E | A | S | Y | Q |
|---|---|---|---|---|---|
| **Value** | 0 | 1 | 2 | 3 | 4 |

f. Queue: UESTION

5 Comparisons: U is compared with E, A, S, Y, and Q in the Symbol Table

Symbol Table:

| **Key** | E | A | S | Y | Q | U |
|---|---|---|---|---|---|---|
| **Value** | 0 | 1 | 2 | 3 | 4 | 5 |

g. Queue: ESTION

1 Comparison: E is compared with E and stops

Symbol Table:

| **Key** | E | A | S | Y | Q | U |
|---|---|---|---|---|---|---|
| **Value** | 6 | 1 | 2 | 3 | 4 | 5 |

h. Queue: STION

3 Comparisons: S is compared with E, A, and S, and then stops

Symbol Table:

| **Key** | E | A | S | Y | Q | U |
|---|---|---|---|---|---|---|
| **Value** | 6 | 1 | 7 | 3 | 4 | 5 |

i. Queue: TION

6 Comparisons: T is compared with all elements already in the Symbol Table

Symbol Table:

| **Key** | E | A | S | Y | Q | U | T |
|---|---|---|---|---|---|---|---|
| **Value** | 6 | 1 | 7 | 3 | 4 | 5 | 8 |

j. Queue: ION

7 Comparisons: I is compared with all elements already in the Symbol Table

Symbol Table:

| **Key** | E | A | S | Y | Q | U | T | I |
|---|---|---|---|---|---|---|---|---|
| **Value** | 6 | 1 | 7 | 3 | 4 | 5 | 8 | 9 |

k. Queue: ON

8 Comparisons: O is compared with all elements already in the Symbol Table

Symbol Table:

| **Key** | E | A | S | Y | Q | U | T | I | O |
|---|---|---|---|---|---|---|---|---|---|
| **Value** | 6 | 1 | 7 | 3 | 4 | 5 | 8 | 9 | 10 |

l. Queue: N

9 Comparisons: N is compared with all elements already in the Symbol Table

Symbol Table:

| **Key** | E | A | S | Y | Q | U | T | I | O | N |
|---|---|---|---|---|---|---|---|---|---|---|
| **Value** | 6 | 1 | 7 | 3 | 4 | 5 | 8 | 9 | 10 | 11 |

2. Give a trace of the process of inserting the keys E A S Y Q U E S T I O N into an initially empty table using BinarySearchST. How many compares are involved?

In a Binary Search Symbol Table, the symbol table is in order. Therefore, when a new element needs to be added to the Symbol Table, a binary search operation can be performed to find where the element should be inserted. In the best case, a sequential search will take 1 comparison and update a key, and in the worst case, sequential search will take N comparisons and add a new element. On the other hand, a binary search will take about $\log_2 N$ comparisons regardless of whether it is updating or adding a key. In this particular example, a Binary Search Symbol Table takes:

0 + 1 + 2 + 2 + 2 + 3 + 3 + 3 + 3 + 3 + 3 + 3 comparisons = 28 comparisons

a. Queue: EASYQUESTION

Comparisons (0):

1) Find the middle element (null). Nothing to compare.

Symbol Table:

| **Key** | E |
|---|---|
| **Value** | 0 |

b. Queue: ASYQUESTION

Comparisons (1):

1) Find middle middle element (A) and compare. It is greater.

    a) So find middle element on the right (null). Nothing to compare.

Symbol Table:

| **Key** | A | E |
|---|---|---|
| **Value** | 1 | 0 |

c. Queue: SYQUESTION

Comparisons (2):

1) Compare with middle element (A). It is greater.

    a) So find the middle element on the right side (E).

2) Compare with new middle element (E). It is greater.

    a) So find middle element on the right (null). Nothing to compare.

Symbol Table:

| **Key** | A | E | S |
|---|---|---|---|
| **Value** | 1 | 0 | 2 |

d. Queue: YQUESTION

Comparisons (2):

1) Compare with middle element (E). It is greater.
    a) So find middle element on right side (S).
2) Compare with S. It is greater
    a) So find middle element on right (null). Nothing to compare.

Symbol Table:

| **Key** | A | E | S | Y |
|---|---|---|---|---|
| **Value** | 1 | 0 | 2 | 3 |

3. Queue: QUESTION

Comparisons (2):

1) Compare with middle element (E). It is greater.
    a) So find the middle element on the right side (S).
2) Compare with new middle element (S). It is left.
    a) So find middle element on the left side (null). Nothing to compare.

Symbol Table:

| **Key** | A | E | Q | S | Y |
|---|---|---|---|---|---|
| **Value** | 1 | 0 | 4 | 2 | 3 |

4. Queue: UESTION

Comparisons (3)

1) Compare with middle element (Q). It is greater.
    a) So find middle element on right side (S).
2) Compare with middle element (S). It is greater.

a) So find middle element on right side (Y).

3) Compare with middle element (Y). It is less.

a) So find middle element on the left side(null). Nothing to compare.

Symbol Table:

| **Key** | A | E | Q | S | U | Y |
|---|---|---|---|---|---|---|
| **Value** | 1 | 0 | 4 | 2 | 5 | 3 |

5. Queue: ESTION

Comparisons (3):

1) Compare with middle element (Q). It is less.

a) Find middle element on the left side (A).

2) Compare with new middle element (A). It is greater.

a) Find middle element between A and Q (E).

3) Compare with new middle element (E). They are equal

a) Update table.

Symbol Table:

| **Key** | A | E | Q | S | U | Y |
|---|---|---|---|---|---|---|
| **Value** | 1 | 6 | 4 | 2 | 5 | 3 |

6. Queue: STION

Comparisons (3):

1) Compare with middle element (Q). It is greater.

a) Find middle element on the right side (U).

2) Compare with new middle element (U). It is less.

a) Find middle element between Q and U (S).

3) Compare with new middle element (S). They are equal

a) Update table.

Symbol Table:

| **Key** | A | E | Q | S | U | Y |
|---|---|---|---|---|---|---|
| **Value** | 1 | 6 | 4 | 7 | 5 | 3 |

7. Queue: TION

Comparisons (3):

1) Compare with middle element (Q). It is greater.

a) Find middle element on the right side (U).

2) Compare with new middle element (U). It is less.

a) Find middle element between Q and U (S).

3) Compare with new middle element (S). It is greater

a) Nothing left to compare. Add to table

Symbol Table:

| **Key** | A | E | Q | S | T | U | Y |
|---|---|---|---|---|---|---|---|
| **Value** | 1 | 6 | 4 | 7 | 8 | 5 | 3 |

8. Queue: ION

Comparisons (3):

1) Compare with middle element (S). It is less.

a) Find middle element on the left side (E).

2) Compare with new middle element (E). It is more.

a) Find middle element between E and S (Q).

3) Compare with new middle element (Q). It is less

a) Nothing left to compare. Add to table

Symbol Table:

| **Key** | A | E | I | Q | S | T | U | Y |
|---|---|---|---|---|---|---|---|---|
| **Value** | 1 | 6 | 9 | 4 | 7 | 8 | 5 | 3 |

9. Queue: ON

Comparisons (3):

1) Compare with middle element (Q). It is less.

a) Find middle element on the left side (E).

2) Compare with new middle element (E). It is more.

a) Find middle element between E and Q (I).

3) Compare with new middle element (I). It is more

a) Nothing left to compare. Add to table

Symbol Table:

| **Key** | A | E | I | O | Q | S | T | U | Y |
|---|---|---|---|---|---|---|---|---|---|
| **Value** | 1 | 6 | 9 | 10 | 4 | 7 | 8 | 5 | 3 |

10. Queue: N

Comparisons (3):

1) Compare with middle element (Q). It is less.

a) Find middle element on the left side (I).

2) Compare with new middle element (I). It is more.

a) Find middle element between I and Q (O).

3) Compare with new middle element (O). It is less

a) Nothing left to compare. Add to table

Symbol Table:

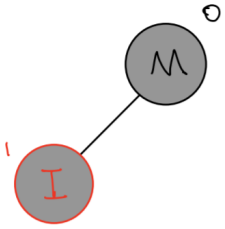
| **Key** | A | E | I | N | O | Q | S | T | U | Y |
|---|---|---|---|---|---|---|---|---|---|---|
| **Value** | 1 | 6 | 9 | 11 | 10 | 4 | 7 | 8 | 5 | 3 |

Both symbol tables have the same final values for each key. However, the Binary Search Symbol Table has fewer comparisons and is sorted.

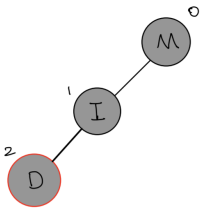
3. Draw the BST that results when you insert the keys M I D T E R M Q U E S T I O N, in that order (associating the value i with the ith key, as per the convention discussed in class) into an initially empty tree. How many compares are needed to build the tree?

a) Adding M - 0 comparisons

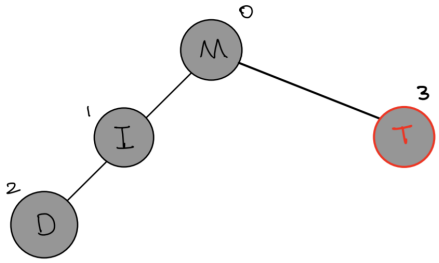

b) Adding I - 1 Comparison



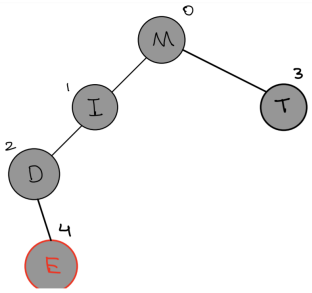

c) Adding D - 2 Comparisons



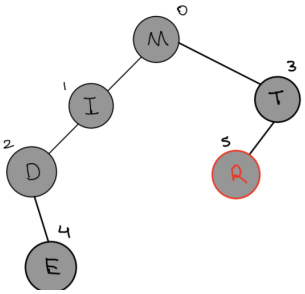

d) Adding T - 1 Comparison



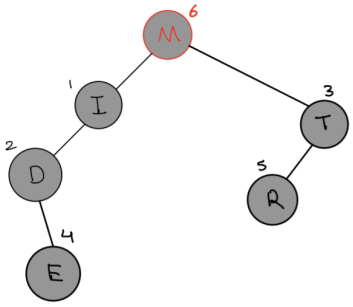

e) Adding E - 3 Comparisons



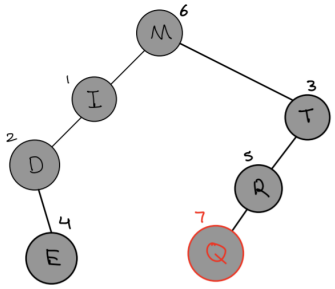

f) Adding R - 2 Comparisons



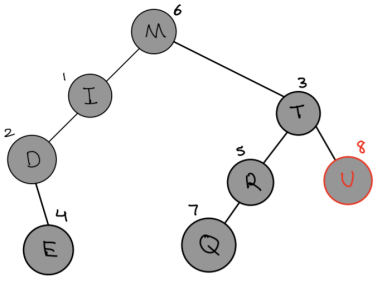

g) Adding M - 1 Comparison



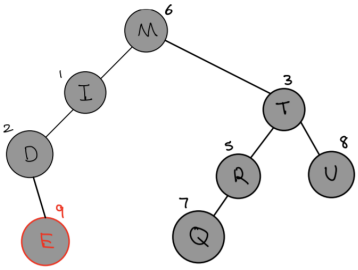

h) Adding Q - 3 Comparisons



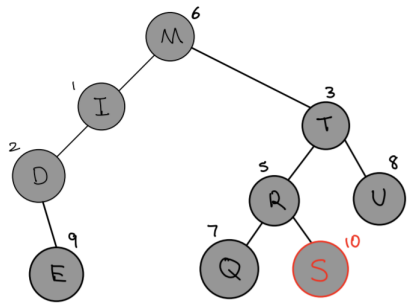

i) Adding U - 2 Comparisons



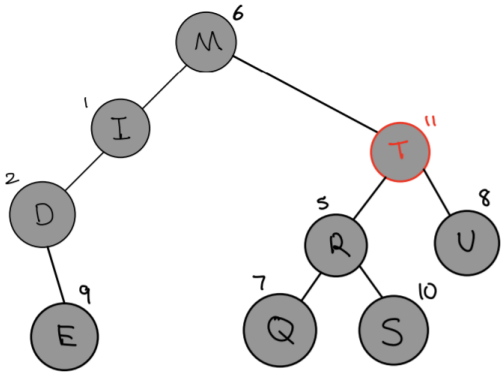

j) Adding E - 4 Comparisons



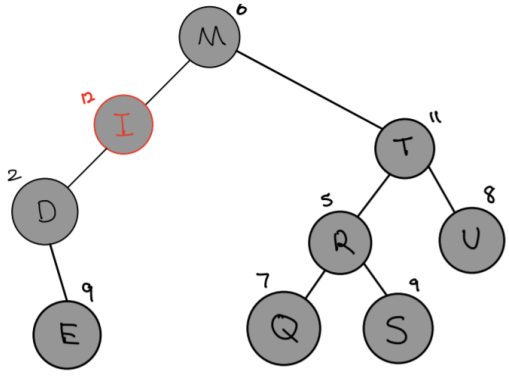

k) Adding S - 3 Comparisons



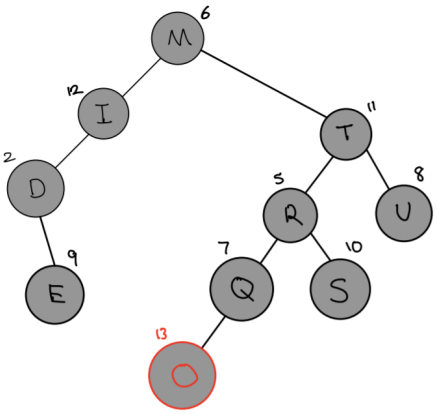

l) Adding T - 2 Comparisons



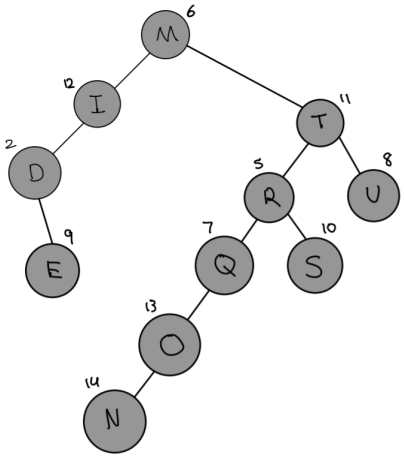

m) Adding I - 2 Comparisons




n) Adding O - 4 Comparisons



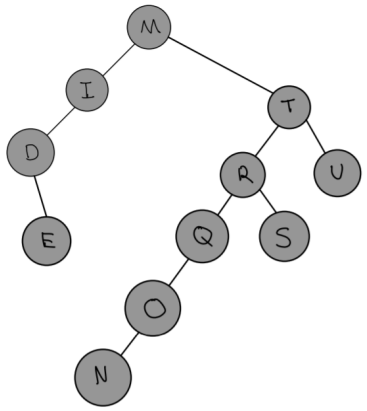

o) Adding N - 5 Comparisons



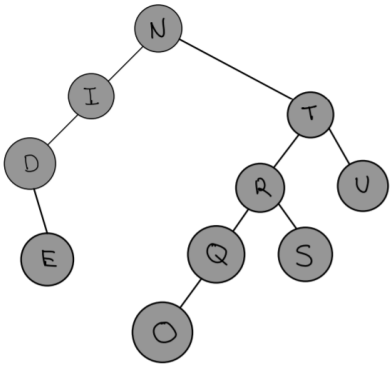

In total, 0 + 1 + 2 + 1 + 3 + 2 + 1 + 3 + 2 + 4 + 3 + 2 + 2 + 4 + 5 comparisons = 35 comparisons were needed to add every element to the tree.

4. Draw the sequence of BSTs that result when you delete the keys from the tree in question 2.a, one by one, in the order they were inserted.

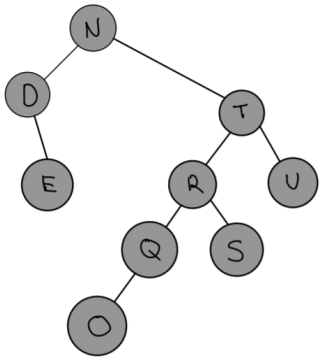
a) Original



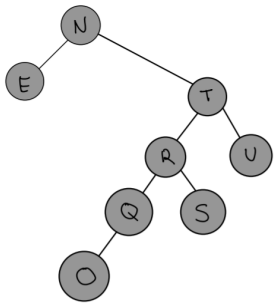

b) Deleting M



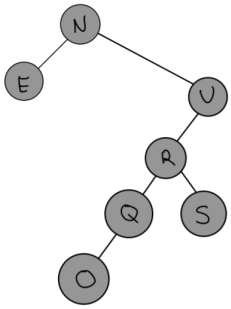

c) Deleting I



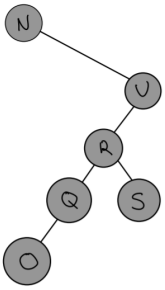

d) Deleting D



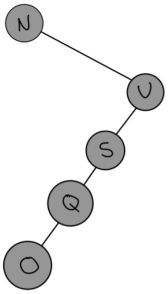

e) Deleting T




f) Deleting E

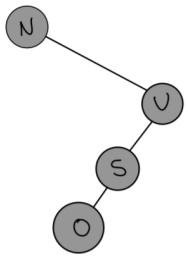



g) Deleting R

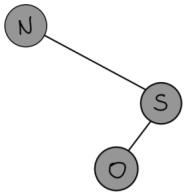



h) Deleting M - Already Deleted

i) Deleting Q

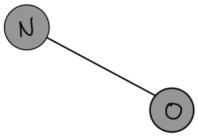



j) Deleting U




k) Deleting E - Already Deleted

l) Deleting S



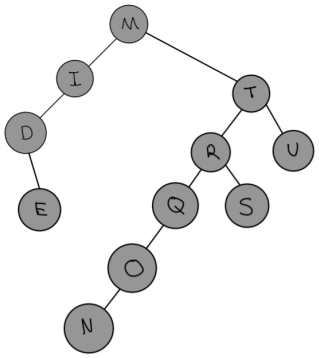

m) Deleting T - Already Deleted

n) Deleting I - Already Deleted

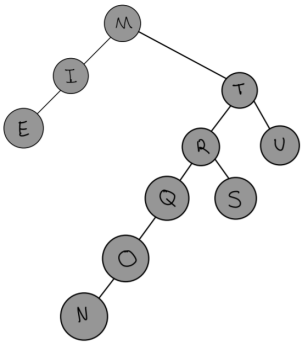
o) Deleting O


5. Draw the sequence of BSTs that result when you delete the keys from the tree in question 2.a, one by one, in alphabetical order.

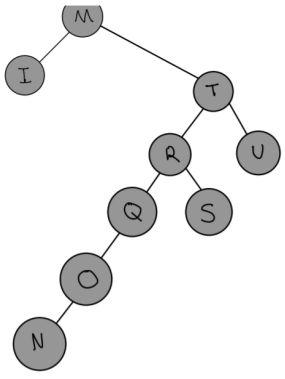
a) Original



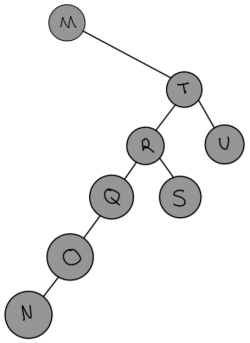

b) Deleting D



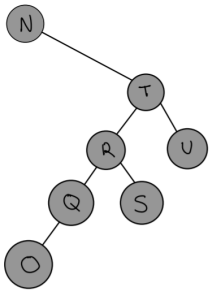

c) Deleting E



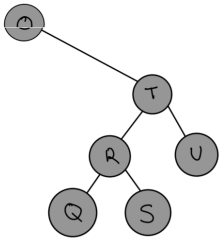

d) Deleting I



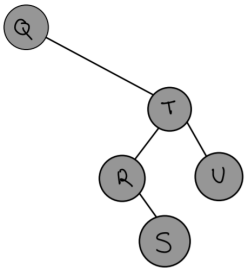

e) Deleting M



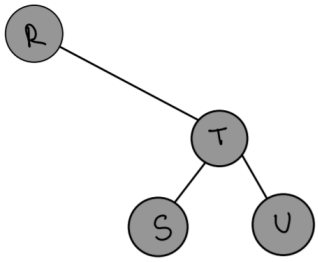

f) Deleting N



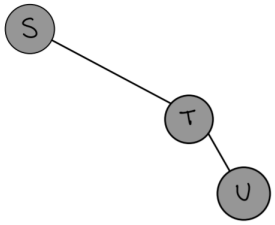

g) Deleting O



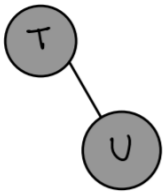

h) Deleting Q




i) Deleting R



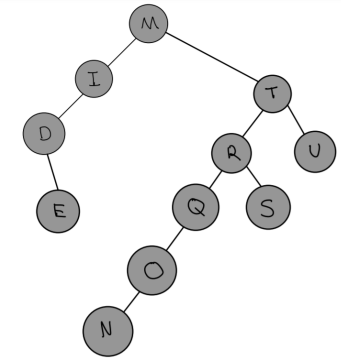

j) Deleting S



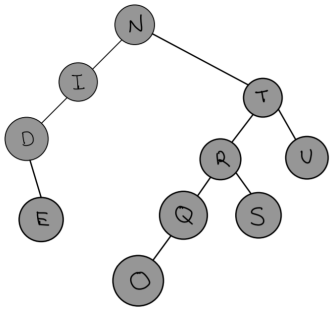

k) Deleting T


6. Draw the sequence of BSTs that result when you delete the keys from the tree in question 2.a, one by one, by successively deleting the key at the root.

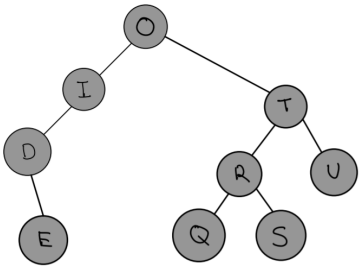
a) Original



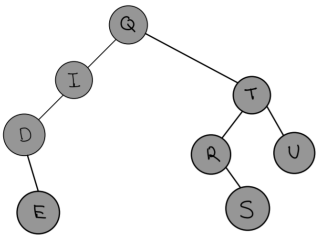

b) Deleting M



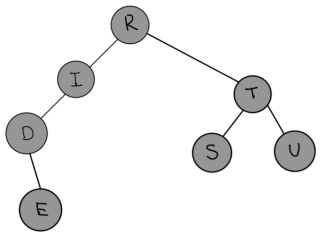

c) Deleting N



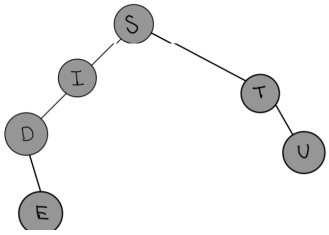

d) Deleting O



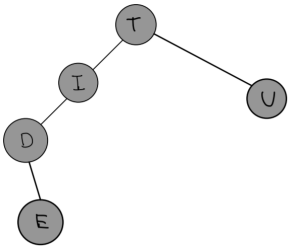

e) Deleting Q



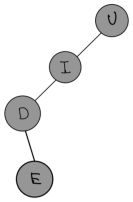

f) Deleting R



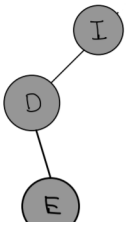

g) Deleting S



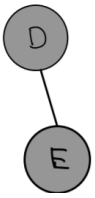

h) Deleting T




i) Deleting U



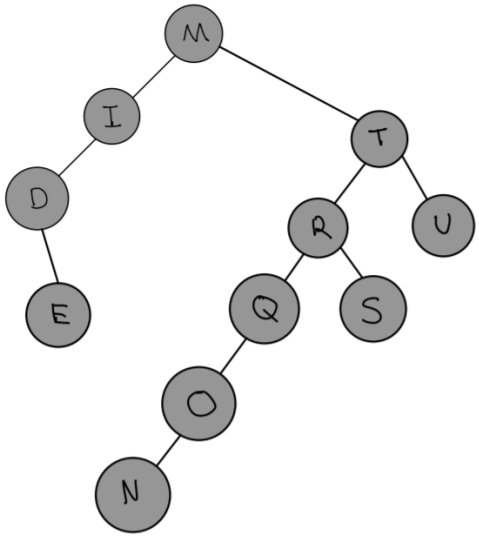
j) Deleting I




k) Deleting D


6. Give the sequences of nodes examined when the methods in BST are used to compute each of the following quantities for the tree in question 2.a

Original:




a. floor("P")

The root (M) is less than the target (P), so we search the right subtree. Now, since the root of this subtree (T) is more than the target (P), we search its left subtree. Since the root of

the new subtree (S) is still more than the target (P), we search its left subtree. The root of the new subtree (Q) is now less than the target, so we search its right subtree. But Q does not have a right subtree, so the floor of P is Q. The sequence of nodes is M→T→R→Q.

b. select(5)

Select(5) returns the fifth element of the tree in ascending order, which is, in this case, N. We need to start off by ranking each element by the number of children in its left subtree plus one. The rank of the root, M, is only 4, and that is less than the 5 we are looking for. So we go to its right subtree. The rank of the root of this subtree, T, is 5, and adding the rank of M gives a rank of 9 which is greater than 5. So we need to go to the left subtree of T. The root R has a rank of four, and adding the rank of four from M gives a rank of eight, which is still greater than the five we are looking for. We keep repeating this process until we reach N. The rank of N is one, and the rank of M is four, and adding them together gives a rank of 5, the rank we are looking to select. The sequence of nodes we followed to find select(5) was M → T → R → Q → O → N.

c. ceiling("V")

The root (M) is less than the target, so we search the right subtree. The root of this subtree (T) is still less than the target, so we search its right subtree. The root of this new subtree (U) is still less than the target, so we search its right subtree. But U does not have a right subtree. So there is no element greater than or equal to V. So there is no ceiling of V. The sequence of nodes was: M → T → U → null.

d. rank("S")

S has no children on its left side, so its rank is just 1.