Data Structures TD3: Trees

Amandine Decker & Marie Cousin

October 2023

INFORMATION: We will correct the following exercises today:

- All the exercises from the introduction section;
- In section 2, exercise 1;
- In section 3, exercise 1;

The other ones are for you to practice, if you want to.

1 Introduction

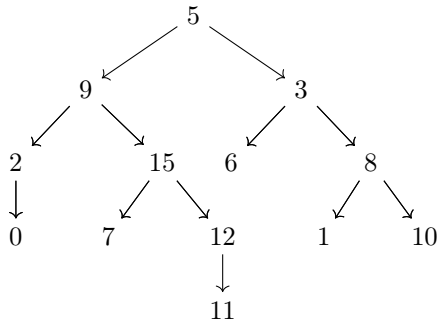
1. **Research algorithm** (the one from the CM):

```
Require: T, value
S ← S_0
add(T, S)
while !isEmpty(S) do
    tree ← get(S)
    remove(S)
    if !isEmpty(tree) then
        if tree['root'] == value then
            return true
        end if
        if !isEmpty(tree['right child']) then
            add(tree['right child'], S)
        end if
        if !isEmpty(tree['left child']) then
            add(tree['left child'], S)
        end if
    end if
end while
return false
```

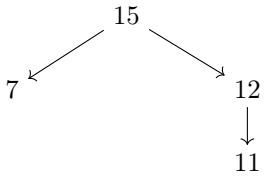
Apply the algorithm and write down the different states of the stack S after each iteration of the while loop:

(a) for the following tree and the value 13;

(b) for the following tree and the value 15;



2. In the tree (let it be T) from the previous question:
   (a) What is the value of T['left child']['right child']['right child']['root']?
   (b) What is the value of T['right child']['left child']['left child']?
   (c) What is the value of T['right child']?
   (d) What is the value of the root of T['left child']['left child']['left child']?
   (e) What is the value of T['right child']['right child']['left child']['root']?
   (f) What is the value of T['left child']['left child']?
   (g) How do you access the node of value 6?
   (h) How do you access the node of value 7?
   (i) How do you access the node of value 11?
   (j) How do you access the following subtree?



   (k) How do you change the value of the node of value 8 to 4 ?

2 Reading an Algorithm

1. What does the following algorithm do? *Hint: You can draw a small tree and apply the algorithm to it to get an idea of what it does.*

```
Require: T
if !isEmpty(T) then
    Q ← Q_0
    add(T, Q)
    c ← 0
else
    return 0
end if
while !isEmpty(Q) do
    tree ← get(Q)
    remove(Q)
    c ← c + tree['root']
    if !isEmpty(tree['left child']) then
        add(tree['left child'], Q)
    end if
    if !isEmpty(tree['right child']) then
        add(tree['right child'], Q)
    end if
end while
return c
```

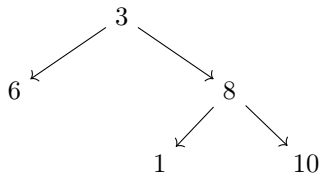
2. What does the following algorithm do? *Hint: You can draw a small tree and apply the algorithm to it to get an idea of what it does.*

```
Require: T
1: if !isEmpty(T) then
2:     Q ← Q_0
3:     add(T, Q)
4:     c ← 0
5: else
6:     return 0
7: end if
8: while !isEmpty(Q) do
9:     tree ← get(Q)
10:    remove(Q)
11:    c ← c + 1
12:    if !isEmpty(tree['left child']) then
13:        add(tree['left child'], Q)
14:    end if
15:    if !isEmpty(tree['right child']) then
16:        add(tree['right child'], Q)
17:    end if
18: end while
19: return c
```

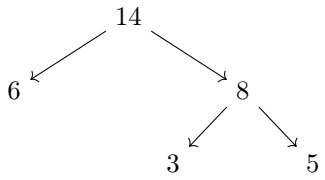
3 Creating an Algorithm

1. Describe an algorithm that would verify if all the nodes of a tree are different pairwise (*i.e.*, the same value never appears twice). Describe the input(s), output(s), and what this algorithm would do.

2. Describe an algorithm that would verify if a tree is a *sum tree*. What we call a sum tree here is a tree where each node is equal to the sum of its direct children. The leafs (*i.e.*, the nodes where both the left and right children are empty) can have any values. For example below, (a) is not a sum tree but (b) is.



(a) Not a sum tree

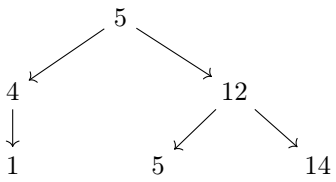


(b) Sum tree ($3 + 5 = 8$ and $6 + 8 = 14$)

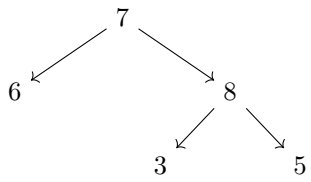
4 Optional Exercise - Due December 8th

```
Require: T
 1: if !isEmpty(T) then
 2:     Q ← Q_0
 3:     add(T, Q)
 4: else
 5:     return True
 6: end if
 7: while !isEmpty(Q) do
 8:     tree ← get(Q)
 9:     remove(Q)
10:     if !isEmpty(tree['left child']) then
11:         if tree['left child']['root'] > tree['root'] then
12:             return False
13:         end if
14:         add(tree['left child'], Q)
15:     end if
16:     if !isEmpty(tree['right child']) then
17:         if tree['right child']['root'] < tree['root'] then
18:             return False
19:         end if
20:         add(tree['right child'], Q)
21:     end if
22: end while
23: return True
```

1. Apply the algorithm above to the trees (a) and (b);



(a)



(b)

2. What does this algorithm do? *Hint: you can try on other trees to get a better idea.*