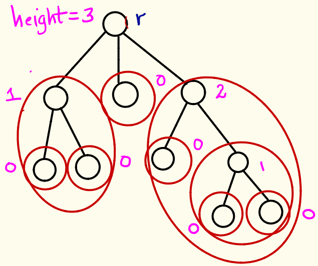


Recurrence equation for the height of a tree

height(r) = 0 if r is a leaf

height(r) = max{height of subtrees} + 1 if r internal

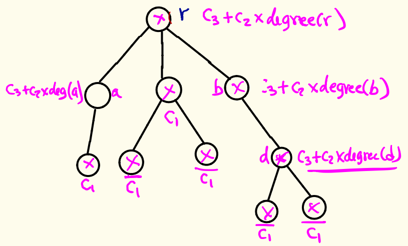
Algorithm height(r)

In: Root r of a tree

Out: Height of tree

```
if r is a leaf then return 0
else {
    mh ← -1   // max height
    for each child c of r do {
        h ← height(c)
        if h > mh then mh ← h
    }
    return mh+1
}
```

Time Complexity Analysis



Algorithm height(r)
In: Root r of a tree
Out: Height of tree

c_1 { **if** r is a leaf **then** **return** 0
else {
mh ← -1 // max height
$\#iterations = degree(r)$
for each child c of r **do** {
h ← $\boxed{height(c)}$ ignore
if h > mh **then** mh ← h
}
return mh+1
}

c_2; $c_2\times degree(r)$; $c_3+c_2\times degree(r)$; c_3

1. First, analyze algorithm ignoring recursive calls.
   c_1 operations in base case; $c_3+c_2\times degree(r)$ in recursive case
2. Determine the number of calls
   One call is performed per node.
3. Count total number of operations.

$$\underbrace{\sum_{leaves} c_1} + \sum_{\substack{internal\\ node(u)}} (c_3+c_2\times degree(u)) = c_1\times \#leaves + c_3\times \#internal + c_2 \sum_{\substack{internal\\ node(u)}} degree(u)$$

$\#edges = n-1$

$\cancel{c_1}\times \#leaves(n) + \cancel{c_3}\times \#internal + \cancel{c_2}(n-\cancel{1})$ is $O(n)$

Algorithm TOC(r, indentation)

In: Root r of a tree representing the structure of a book,
integer indentation (in the initial call to the algorithm
the value of indentation is zero).

Out: {Print table of contents properly indented.

```
for i←1 to indentation do
        print(' ')
print r.data
for each child u of r do
        TOC(u, indentation+1)
```

Space Complexity: amount of memory needed for execution stack and for data structures.

$c_1 n +$

Max # of activation records simultaneously in the execution stack is equal to the height of the tree+1

$c_2 \times$ height

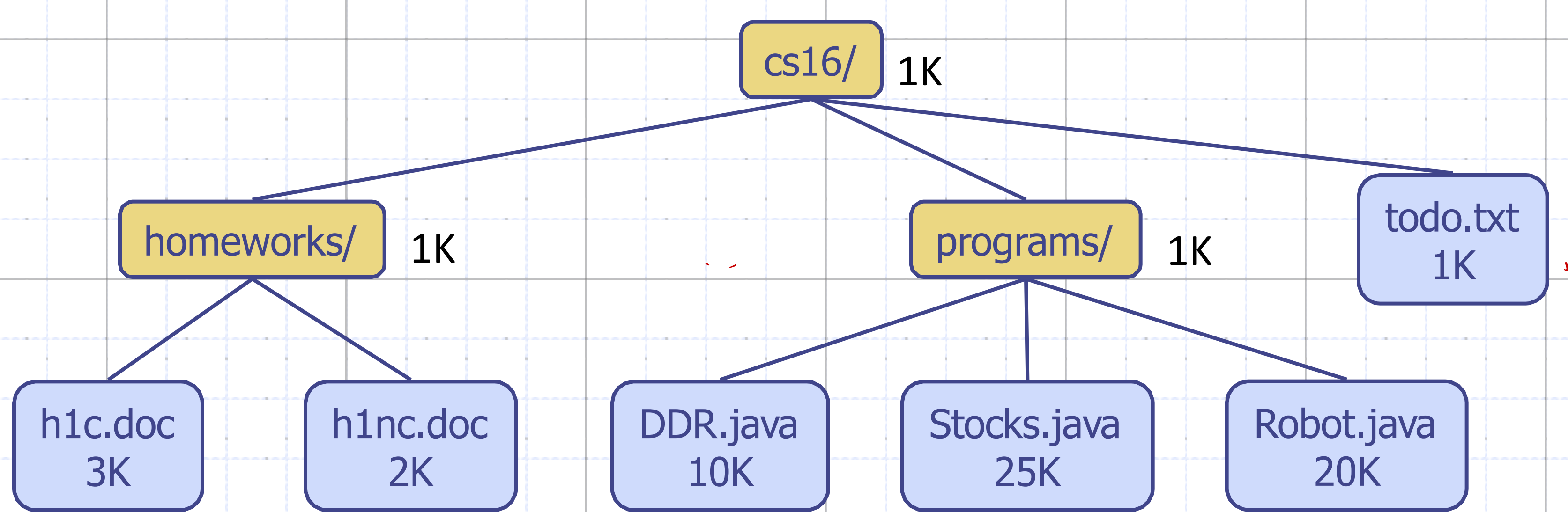
$$f_s(n) = \cancel{c_1} n + \cancel{c_2} \times \text{height}$$

is $O(n)$

Postorder Traversal

Application

Compute space used by the files in a directory and its subdirectories



Algorithm diskSpace(r)
In: root r of a file system tree
Out: Total space used by the file system

$$C_1 \left\{ \begin{array}{l} S \leftarrow 0 \\ \underline{\text{for}} \text{ each child } u \text{ of } r \ \underline{\text{do}} \\ \qquad S \leftarrow \boxed{\text{diskSpace}(u)} + S \; \} \; C_2 \; \} \; C_2 \times \text{degree}(r) \\ \underline{\text{return}} \; S + r.\text{space} \end{array} \right.$$

Ignoring recursion

$C_2 \times \text{degree}(r) + C_1$

How many calls?

One per node

$$f(n) = \sum_{\substack{\text{nodes}\\ u}} (c_1 + c_2 \times \text{degree}(u))$$

$$= \underbrace{\sum_{\substack{\text{nodes}\\ u}} c_1}_{} + c_2 \underbrace{\sum_{\substack{\text{nodes}\\ u}} \text{degree}(u)}_{n-1}$$

$$= \cancel{c_1} n + \cancel{c_2} (n - \cancel{1}) \text{ is } O(n)$$