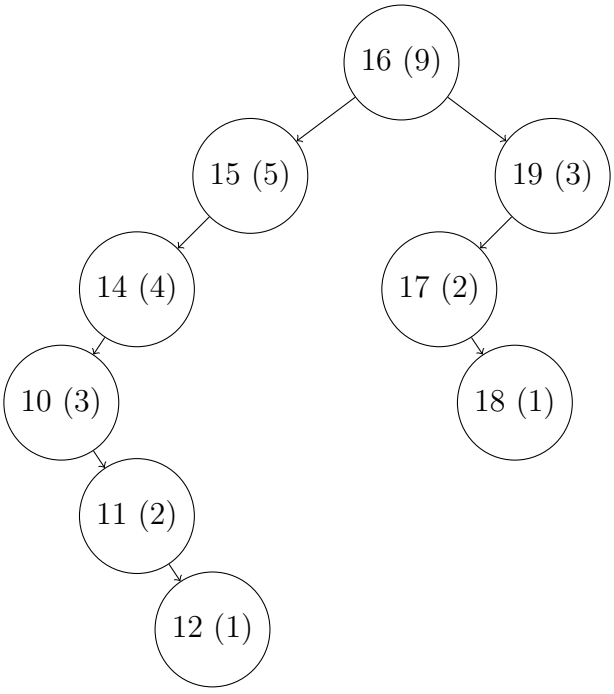
Midterm Practice
CS 240E Winter 2021
University of Waterloo
Monday, March 8, 2021

Scapegoat Tree Insertion

Insert the element 13 in the following scapegoat (2/3)-tree. The numbers in the brackets are the number of nodes in that subtree.



Lower Bound

Suppose you own n electrical devices. Each of them comes with a charger cable, which you tossed into a box when you got it. But now it is time to recharge the devices, and so you must find for each one the correct charger cable. For each device, exactly one charging cable is correct. The charging cables look similar enough that you cannot compare them amongst themselves. The only thing that you can do is plug a cable into a device, which will tell you whether the plug fits, or is too big, or is too small.

Argue that any algorithm to find cables for all devices must use $\Omega(n \log n)$ such operations in the worst case.

Multi-Way Merge

Given a set of k sorted arrays, where the combination of the k arrays has n elements in total, design a worst case $O(n \log k)$ time algorithm that produces a single sorted array containing all n elements.

Double Hashing

a) Give the contents of the hash table that results when you insert items with the keys

$$I, E, A, T, P, R, S, L, Y, G$$

in that order into an initially empty table of size $M = 16$. Use double hashing, with the hash functions given below:

c	A	E	G	I	L	P	R	S	T	Y
ASCII(c)	65	69	71	73	76	80	82	83	84	89
$h_1(c) =$ ASCII(c) mod 16	1	5	7	9	12	0	2	3	4	9
$h_2(c)$=((13 · ASCII(c)) mod 15)+1	6	13	9	5	14	6	2	15	13	3

b) Find one character $c \in \{A, \ldots, Z\}$ that is different from all the ones in part (a) such that inserting c into your result from part (a) with the same hash-function would lead to "failure to insert" (i.e., to re-hashing). Briefly justify your answer.

Hint: Make use of the fact that the table-size M is not well-chosen for double-hashing.

c) Suppose you have an implementation of double-hashing (not with the above hashing functions). Unfortunately there is a bug in your double-hashing code such that one or both of the hash functions always return the same value $x > 0$. Describe what happens in each of these situations:

(i) h_1 has the bug,

(ii) h_2 has the bug,

(iii) both h_1 and h_2 have bugs.

You should comment on what the resulting probe sequence will be and how long you would expect that an Insert will take. (You need not prove the run-time of Insert formally, but compare its behaviour to hashing methods that we have seen in class.)

You may assume that M and the hashing-functions are suitable for double-hashing. In particular the load factor α is kept small ($\alpha < \frac{1}{2}$), M is a prime, $h_2(k) \in \{1, \ldots, M-1\}$, and h_1 and h_2 hash uniformly if they don't have bugs.