University of Waterloo
CS240 Fall 2021
Assignment 3

Due Date: Wednesday, Nov 3 at 5:00pm

The integrity of the grade you receive in this course is very important to you and the University of Waterloo. As part of every assessment in this course you must read and sign an Academic Integrity Declaration before you start working on the assessment and submit it **before the deadline of November 3rd** along with your answers to the assignment; i.e. **read, sign and submit A03-AID.txt now or as soon as possible**. The agreement will indicate what you must do to ensure the integrity of your grade. If you are having difficulties with the assignment, course staff are there to help (provided it isn't last minute).

The Academic Integrity Declaration must be signed and submitted on time or the assessment will not be marked.

Please read `http://www.student.cs.uwaterloo.ca/~cs240/f20/guidelines.pdf` for guidelines on submission. **Each question must be submitted individually to MarkUs as a PDF** with the corresponding file names: a3q1.pdf, a3q2.pdf, ... , a3q6.pdf .

It is a good idea to submit questions as you go so you aren't trying to create several PDF files at the last minute.

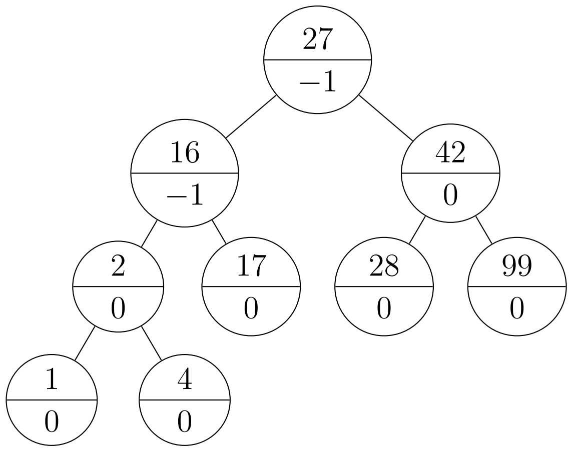
Late Policy: Assignments are due at 5:00pm. To accommodate any small time differences with our submission server, there is a grace period of 5 minutes; i.e. we will accept assignments until 5:05pm without penalty. Assignments submitted after 5:05pm will incur a penalty of 1 mark per minute to a maximum of 10 marks.
Assignments submitted after 5:15pm will not be accepted but may be reviewed (by request) for feedback purposes only.

Problem 1 [0+2+2=4 marks]

a) **Practice** (not worth any marks): Starting with an empty AVL tree, insert the following keys in order: 27 99 17 28 42 16 1 2 4.

You should obtain the AVL tree given in the next part.

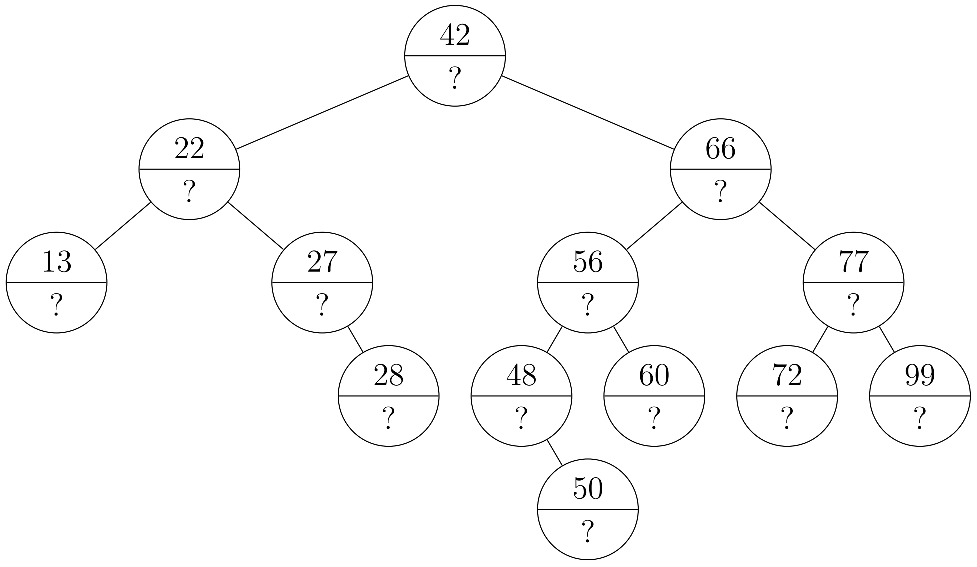
b) Given the following AVL tree:
Note: this tree shows balance factors instead of height.



Insert the following keys in order: $8^\star$, 22, 21, $18^\star$.

Show the resulting AVL trees with **balance factors** (not height) for each node after the elements marked with star ($\star$) are inserted.
Note: you should only show 2 trees.

c) Consider the following AVL tree:



Given the above tree, delete the following keys in order:

66, $13^\star$, 72, 77, $56^\star$, $42^\star$

Show the resulting AVL trees with **balance factors** (not height) for each node after the elements marked with star ($\star$) are deleted. If you have a choice of which element to move up, pick the inorder successor.
Note: you should only show 3 trees.

Problem 2 AVL Trees [4+6 marks]

In this question, we want to add support for an operation `ithSuccessor` on AVL trees (in addition to the standard operations `insert, delete, find`). The operation `ithSuccessor` has two parameters, x and $i \geq 0$, and returns the ith inorder successor of the node x. If $i = 0$, then the node x itself is returned. You may assume that all input is valid; i.e. the successor exists (but may not be in the subtree rooted at x).

We assume that the nodes have the following fields:

- `key` – the key of the node;
- `left` – pointer to the left child;
- `right` – pointer to the right child;
- `balance` – balance factor of the node;
- `parent` – pointer to the parent of the node;
- `isLeft` – is true if the node is a left child of its parent;
- `isRight` – is true if the node is a right child of its parent;
- `numLeft` – holds the number of nodes in the left subtree of the node;
- `numRight` – holds the number of nodes in the right subtree of the node.

a) Give an algorithm `ithNode(x, i)` which returns the ith inorder node in the subtree rooted at x. For example, suppose the subtree contains m nodes, when $i = 1$, the minimum element in the subtree is returned and when $i = m$ the maximum element in the subtree is returned. You may assume that the subtree has at least i elements. Your algorithm should take worst-case $O(\log(m))$ time. Briefly justify that your algorithm achieves this runtime.

b) Give the algorithm for `ithSuccessor(x, i)` for an AVL tree with n nodes. Your algorithm should take worst-case $O(\log(n))$ time and must use `ithNode(x, i)` from above. Briefly justify that your algorithm achieves this runtime.

Problem 3 [3+3+4=10 marks]

Which of the following binary trees must have height $O(\log n)$? Justify your answer.

a) There is a constant $c > 0$ such that for all nodes z in T, $z.left.height \leq z.right.height + c$.

b) Every node z that is not a leaf in T has exactly two children.

c) Given a BST T, let $N(z)$ be the number of nodes in the subtree rooted at z. If z is an empty subtree, then $N(z) = 0$.
There is a constant $0 < c \leq 1$ such that for every node z in T, $N(z.left) \geq c \times N(z.right) - 1$ and $N(z.right) \geq c \times N(z.left) - 1$.

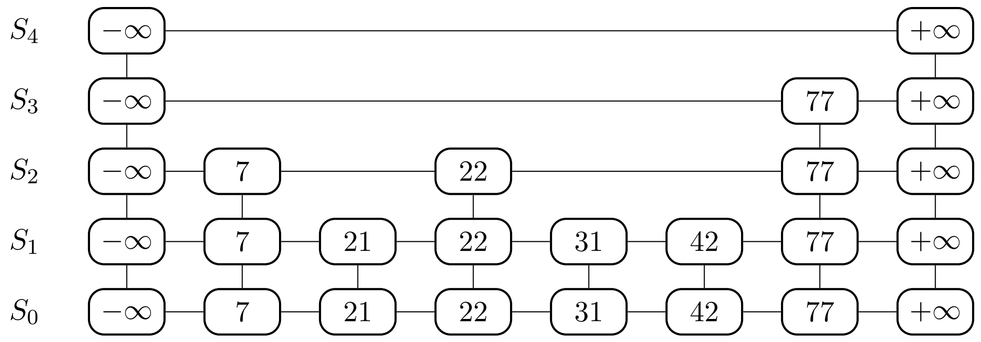
Problem 4 [0+2+1+2+2+2=9 marks]

a) Practice (not worth any marks): Starting with an empty skip list, insert the following keys in order: 22 31 77 7 21 42 using the following flip sequence:

HHTHTHHHTHHTHTHT

You should obtain the skip list given in the next part.

b) Given the following skip list:



Show the skip list that is created by inserting the keys: 37, 66, 13, 4, 27, 99 into the given skip list. You must use the following coin flip sequence:

$$HHHHTTHTHTHHTHTTHHHHHTHHTT$$

Note that each coin flip in the sequence will only be used once, in order and there may be some unused coin flips.

After inserting all keys, determine the exact number of comparisons (between 2 keys) required to search for the keys inserted in this part (that is, indicate what the search cost would be for 37, and then the search cost for 66, and so on).

For example, a search for 18 in the given skip list above, requires 6 comparisons.

Key	37	66	13	4	27	99
Comparisons						

For the remaining parts, assume that the probability of adding a level to a tower is p $(0 < p < 1)$, as opposed to $\frac{1}{2}$.

c) Explain why the probability that a key in the skip list has height at least i is p^i.

d) Show that the expected number of extra nodes (i.e., the total number of nodes in skip list not including the nodes in S_0) is $O(n)$. Therefore, the space requirements for this skip list are linear in the number of keys being stored.

e) Similar to what was done in lecture, let $C(k)$ be the expected total path length which rises k levels when working back to the top-most, left-most node. Find the recurrence relation for $C(k)$ in terms of k and p.

f) Assuming $C(0) = 0$, show that your recurrence relation from the previous part has closed form $C(k) = k/p$.

Problem 5 [2+2+2=6 marks]

Consider the list of keys:

[1 2 3 4 5 6 7 8 9 10]

and assume we perform the following searches:

$$8, 6, 3, 7, 5^\star, 5, 2, 6, 2, 8^\star, 3, 9, 3, 8^\star$$

a) Using the move-to-front heuristic, give the list ordering after the starred ($\star$) searches are performed. Additionally, record the number of comparisons between keys after each search, as well as, the total number of comparisons.

8	6	3	7	5	5	2	6	2	8	3	9	3	8	Total

b) Repeat part (a), using the transpose heuristic instead of the move-to-front heuristic.

c) Another heuristic is *move-to-front2 (MTF2)* that is similar to *move-to-front (MTF)* except that when an element is found at position i it is moved to position $\left\lfloor \frac{i}{2} \right\rfloor$. Repeat part (a), using this heuristic.

Problem 6 [2+2+2+2=8 marks]

a) Insert the following binary keys into an initially empty (uncompressed) binary trie:

0110\$, 110\$, 0000\$, 01001\$, 01\$, 00\$, 01011\$, 0\$, 111\$, 010\$, 11\$

b) From your answer to part (a), delete the following keys, and show the trie after each deletion:

01011\$, 00\$, 0110\$

c) Repeat part (a), except use a compressed trie.

d) Repeat part (b), starting from your answer to part (c).