University of Waterloo
CS240 - Spring 2017
Assignment 3

Due Date: Wednesday, June 14th, 5pm

Please refer to the course webpage for guidelines on submission. Submit your written solutions electronically as a PDF with file name `a03wp.pdf` using MarkUs. We will also accept individual question files named `a03q1w.pdf`, `a03q2w.pdf`, etc., if you wish to submit questions as you complete them.

Unless the base is explicitly specified, we assume in all questions that the logarithm operation is base 2.

Problem 1 AVL trees [5+5=10 marks]

This question is about insertion and deletion of elements in an AVL Tree.

(a) Consider the AVL tree shown in Figure 1. Draw the tree again while replacing $b_0, b_1, \ldots, b_{11}$ with the respective balance factors on each node (similar to slide 17 of Module 4).

Perform the operation Insert(5) on the tree. Draw the tree before each rotation (including the intermediate tree in a double rotation), if any, and draw the final tree. Keep the balance factors updated up until any call to fix is required.

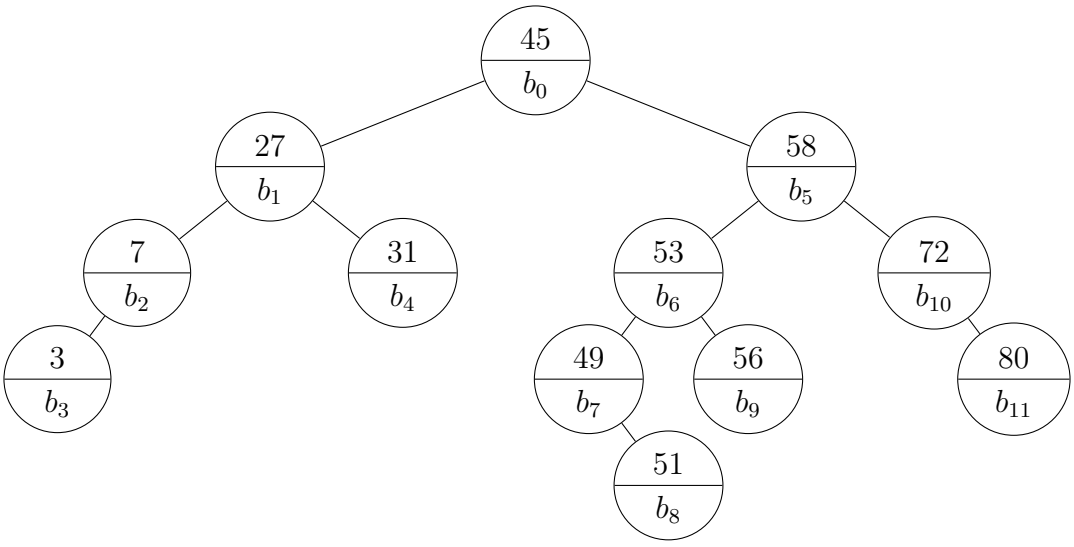

Figure 1: AVL tree of problem 1.

(b) Consider the tree in Figure 1 (*not* the tree obtained after the insertion in part a). Perform the operation delete(27) on the tree, swapping with the inorder successor. Draw the tree before each rotation (including the intermediate tree in a double rotation), if any, and draw the final tree. Keep the balance factors updated up until any call to fix is required. Draw the final tree with balance factors.

Problem 2 AVL-2 trees [4+3+4=11 marks]

We consider a modified version of AVL trees, where the height difference between the left and right subtrees of any node is in $\{-2, -1, 0, 1, 2\}$ instead of $\{-1, 0, 1\}$. These are called AVL-2 trees. We let m_i be the minimum number of nodes of an AVL-2 tree with height i for $i \geq 0$. For example, $m_1 = 2$, since there must be at least two nodes in a tree with height 1, such as the tree in Figure 2.

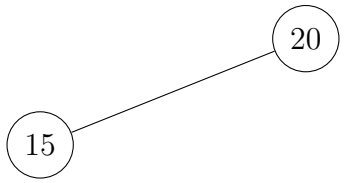

Figure 2: AVL-2 tree with 2 nodes and height 1

(a) For $i = 2, \ldots, 5$, determine m_i and give an example of an AVL-2 tree with m_i nodes. Balance factors do not need to be indicated.

(b) Find a recurrence relation for m_i and give its initial conditions.

(c) Using your recurrence, prove that $m_i \geq 2^{i/3}$ by induction on i.

Problem 3 Interpolation Search [4+4+4=12 marks]

We wish to improve upon binary search for the sorted array A with n distinct values. Instead of just blindly selecting the median index $i = \frac{\ell+r}{2}$ from the range of indices $A[\ell, r]$, *interpolation search* guesses the index of key k by estimating "how far away it should be from ℓ". The guess is calculated by interpolating between the left (ℓ) and right (r) boundary values.

```
InterpSearch(A[ℓ, r], k)
A: an array
ℓ: index of the left boundary
r: index of the right boundary
k: key to search for
1.   if A[ℓ] > k  ||  A[r] < k
2.        return false
3.   i ← ℓ + ⌊(r − ℓ) (k−A[ℓ])/(A[r]−A[ℓ]) ⌋
4.   if A[i] = k
5.        return true
6.   else if A[i] < k
7.        return InterpSearch(A[i + 1, r], k)
8.   else
9.        return InterpSearch(A[ℓ, i − 1], k)
```

For example, let $A = (51, 58, 81, 87, 99)$ and $k = 87$. Starting the search in the range $\ell = 0$ and $r = 4$, then $i = 3$, and $A[i] = 87 = k$ so k is in A.

(a) Interpolation search tends to perform best when $A[i] = f(i) = ai + b$ for $0 \leq i \leq n-1$, $a \neq 0$, and $a, b \in \mathbb{R}$. Show that a search always terminates in $O(1)$ time for these arrays, regardless of whether the key being searched for is stored in the array or not.

(b) What is the worst case search time for interpolation search? Give an example of a single array A with n distinct values (i.e., by defining $A[i] = f(i)$ for some function f) as well as a search key k that demonstrates this worst case search, and analyze its runtime. There are infinitely many such examples, but any one of them will suffice.

(c) Suppose $A[i] = f(i) = t\sqrt{i}$ for $0 \leq i \leq n-1$ and some positive number t. Show that the runtime to search for t is $O(\log \log n)$.

Hint: Let m be the smallest integer such that $n \leq 2^{2^m}$. How does m relate to the depth of recursion?

Interestingly, the average-case runtime for interpolation search is $O(\log \log n)$ if the elements are uniformly distributed, though this question is not related to that.

Problem 4 Self-organizing search [3+2+3+3+3+3=17 marks]

In this problem, we analyze the performance of the move-to-front heuristic for linear search. After each search with this heuristic, the element being searched is moved to the front of the linked list. As input, suppose we are given a linked list with keys $x_1, \ldots, x_n$, but we do not know in which order they are. Let the probability of accessing x_i be p_i, where $p_i > 0$ for all i.

(a) We do not know the order of the keys in the linked list, so let us define $X_{i,j}^{(N)}$ the probability that x_i is before x_j after N queries, where $X_{i,j}^{(0)}$ is the initial probability before any query. Note that $X_{j,i}^{(N)} = 1 - X_{i,j}^{(N)}$. Show that after one query, the probability that x_i is before x_j is

$$X_{i,j}^{(1)} = p_i + (1 - p_i - p_j)X_{i,j}^{(0)}.$$

(b) After two queries, show that the probability becomes

$$X_{i,j}^{(2)} = p_i + (1 - p_i - p_j)\left(p_i + (1 - p_i - p_j)X_{i,j}^{(0)}\right).$$

(c) Show by induction that after N queries, the probability is

$$X_{i,j}^{(N)} = p_i\left(1 + (1 - p_i - p_j) + (1 - p_i - p_j)^2 + \cdots + (1 - p_i - p_j)^{N-1}\right) + (1-p_i-p_j)^N X_{i,j}^{(0)}.$$

(d) Show that the limits of $X_{i,j}^{(N)}$ and $X_{j,i}^{(N)}$, for $N \to \infty$, are

$$\frac{p_i}{p_i + p_j} \quad \text{and} \quad \frac{p_j}{p_i + p_j}$$

respectively.

We will now assume that N is large, and use the limit probabilities.

(e) Show that the expected value of the position of x_i is

$$1 + \sum_{j \neq i} \frac{p_j}{p_i + p_j}.$$

Hint: Define $Y_{i,j}$ as an indicator variable that is equal to 1 if x_i is before x_j, or 0 otherwise. How do the values of $Y_{j,i}$ relate to the position of x_i in the list?

(f) What is the expected number E of links visited in a search using this heuristic? (Do not try to simplify the expression)

Problem 5 (Bonus) Skip lists with two levels [2+3=5 marks]

Suppose you have a skip list with only two levels: the lower one has n entries $a_0, \ldots, a_{n-1}$, and the top one has k entries. For simplicity, we assume k divides n, so that $n = km$, for some integer m. We assume that the k top entries are evenly spread out, so they correspond to $a_0, a_m, a_{2m}, \ldots, a_{(k-1)m}$.

(a) What is the worst case runtime for a query? Give a $\Theta(\ )$ expression for this runtime in terms of k and n.

(b) Given n, what choice of k will minimize this worst case? Give a $\Theta(\ )$ expression for this worst case runtime. It is not necessary to justify the choice of k here.