University of Waterloo
CS240 Winter 2020
Assignment 3

Due Date: Wednesday, March 4 at 5:00pm

Please read `http://www.student.cs.uwaterloo.ca/~cs240/w20/guidelines.pdf` for guidelines on submission. This assignment contains only written problems. Submit your written solutions electronically as a PDF with file name a3Submit.pdf using MarkUs. We will also accept individual question files named a3q1.pdf, a3q2.pdf, ... , a3q5.pdf if you wish to submit questions as you complete them.

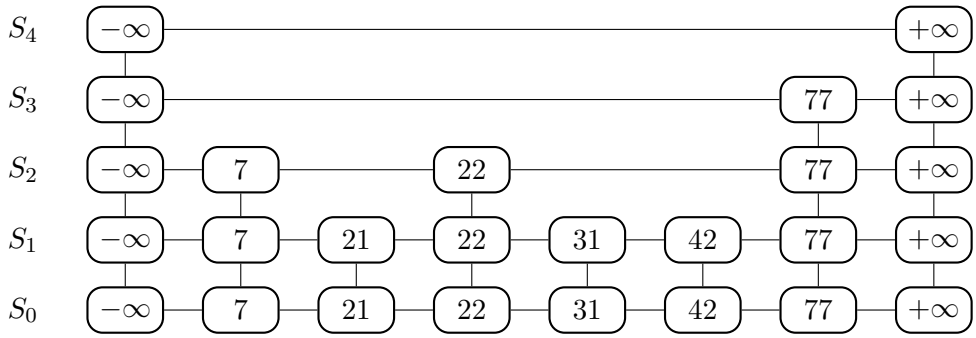
Problem 1 [0+5+2+3+3+3=16 marks]

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 sum of all tower heights not including 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 2 [3+3=6 marks]

Suppose you have a skip list with only three levels: the lower one has n entries $a_0, \ldots, a_{n-1}$, the middle level has k entries and the top level has only the two sentinel values. For simplicity, we assume k divides n, so that $n = km$, for some integer m. We assume that the k entries on the middle level 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.

Problem 3 [3+3+3=9 marks]

Consider the list of keys:

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

and assume we perform the following searches:

9, 5, 2, 6, 4*, 4, 1, 5, 3, 9*, 2, 8, 2, 9*

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.

9	5	2	6	4	4	1	5	3	9	2	8	2	9	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 $\lfloor \frac{i}{2} \rfloor$. Repeat part (a), using this heuristic.

Problem 4 [4+4=8 marks]

The worst case of interpolation search results in $\Theta(n)$ search time. Let i be the interpolated index where the next comparison in the search will be made (formula from lecture):

$$i = \ell + \left\lfloor \frac{k - A[\ell]}{A[r] - A[\ell]} \times (r - \ell) \right\rfloor$$

This splits the array into two parts, a smaller "good side" and a larger "bad side". A recursion on the "bad side" is called an *interpolation failure*.

To improve recursion on the "bad side", a counter `numFailures`, that keeps track of the number of interpolation failures, can be used to modify the interpolation index i (computed above) so that the "good side" and "bad side" are more balanced - hopefully avoiding some future interpolation failures.

a) Give pseudocode for a modified interpolation search in which the two parts are made more balanced by reducing the size of the larger side by up to `numFailures`. Make sure to handle boundary cases.

b) Give the worst case search time for this modified algorithm.

For example, consider the following:

```
k = 66, A = { 6, 35, 42, 58, 59, 67, 68, 82, 85, 96 }
Iteration 1: l = 0, r = 9, i = 6
Iteration 2: l = 0, r = 5, i = 4
k = 66 not found
```

The first iteration of the search compares the key to `A[6] = 68` and then recurses/iterates on the left since `k < A[6]`. Since there are 3 indices on the right of `A[6]` and 6 indices on the left, the right is the good side and the left is the bad side, so recursing on the left in this case is an interpolation failure.

Problem 5 [2+3+2+3+4=14 marks]

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

1001$, 001$, 1111$, 10110$, 10$, 11$, 10100$, 1$, 000$, 101$, 00$

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

10100$, 11$, 1001$

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

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