**DÉPARTEMENT D'INFORMATIQUE ET DE RECHERCHE OPÉRATIONNELLE**

**SIGLE DU COURS:** IFT 2015 (E21)

**NOM DU PROFESSEUR:** Neil Stewart

**TITRE DU COURS:** Structures de Données

**FINAL EXAM**

Date : Tuesday-Wednesday July 20-21, 2021
Heure : Exam available: July 20, 9:00. Exam due: Wednesday July 21, 12:00.
Lieu : Take-home exam

**DIRECTIVES PÉDAGOGIQUES:**

- You have 27 hours to complete this exam.
- All documentation is permitted. Give your references. No LaTeX bonus.
- You must submit *exactly* five pages in pdf format.
  The first page is to identify the exam, but the most important thing is that it contain a rectangle with your family name in upper-case, your given name, and your U de M matricule number, thus: STEWART, Neil. Matricule 235711.
- Your answer to question $i$ must be put on the page $i + 1$ (one side only), $i = 1, 2, 3, 4$. Follow this format exactly.

**PLAGIARISM.** The following constitute plagiarism:

- to have your work done by someone else
- to use, without mentioning it, the work of other persons
- TO EXCHANGE INFORMATION WITH OTHERS DURING AN EXAM
- to falsify documents.

Plagiarism is subject to penalties, up to and including exclusion from the program.

Figure 1: Style for the required rectangular box.

1. Question 1 *(10 points)*

   In Figure 2 there is an example of a 1-2-3 deterministic SkipList, as seen in the course. There are four levels, 1, 2, 3 and 4. The header (Entête $E$) is linked to the terminal node (noeud terminal $T$) at level 3. The node with key 28 has just been inserted, but the person who implemented the insertion algorithm forgot to increase the level of certain nodes when doing the insertion:

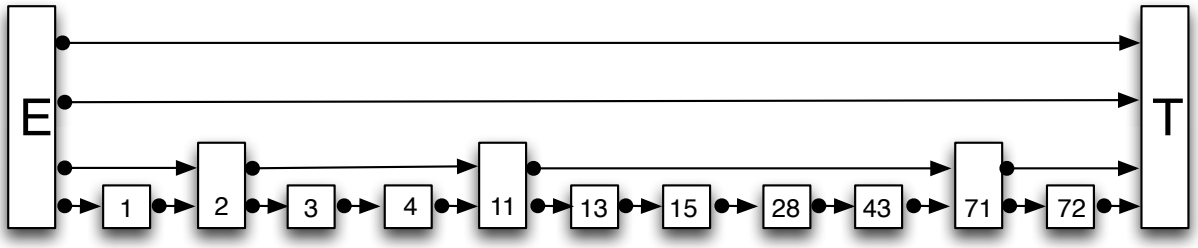


Figure 2: "1-2-3 Deterministic Skiplist"

   (a) Descending from level 3, in order to insert the node containing 28, there was a gap of 3. The height of which node exactly should have been raised at this step?

   Next, the node whose height was just incremented is linked to the node containing 71 at level 2, and descending from level 2, to insert the node containing 28, there was yet another node whose height should have been adjusted. Which one? (It is not necessary to make a drawing: tell me in one or two short sentences which nodes are now included in which lists.)

   (b) Provide a drawing of the 2-4 tree corresponding to the the SkipList after insertion, with the appropriate height adjustments, of the node containing 28. (It's not necessary to prepare a beautiful drawing using the computer: if you prefer, you can for example do a careful drawing by hand, and include a photo of the drawing on the answer page.)

   (c) We mentioned the idea of transforming the top-down SkipList deletion algorithm into a deletion algorithm for 2-4 trees.

   Suppose we want to delete the key 72 from the 2-4 tree that you have just drawn. Give me the rules that must be followed at each step in order to do that (following the topdown SkipList algorithm), but *express yourself only by reference to the 2-4 tree.* I don't want to hear anything about the SkipList from now on! I'm looking for a statement of each *rule,* and not just a description of what happened in the case we're looking at.

   To do this, you must state a rule that might denoted by "(a)", a rule "L-to-R" or "R-to–L" (choose the case that's appropriate in the example we're studying), and a rule "(b-1)" or "(b-2)" (choose the case that's appropriate in the example we're studying).

2. Question 2 *(10 points)*

(a) In the method of external chaining, we could envisage keeping the lists sorted. What are the costs (average cost) as a function of the list length $\lambda = N/M$ for the operations:

- Insertion when we know *a priori* that the key is absent.
- Search when we do not know whether or not the key is present. The answer here may perhaps depend on which case we consider: *a posteriori* the key was present, or *a posteriori* the key was not present.

When I say "as a function of $\lambda$" I am referring to functions like $2\lambda$, $\lambda/4$, 17, $\lambda^2$, $e^{17\lambda}$, ...

(b) When using Threaded trees, we replaced the null references by threads leading to predecessor and successor nodes. If there are $N$ nodes with key in the Threaded tree, there are $N + 1$ threads, including the two null pointers at the beginning and the end.

Give a formal proof of the fact that the complexity of inorder (GRD) traversal of a Threaded tree is $O(N)$.

3. Question 3 *(10 points)*

(a) In the pseudo-code for the Prim algorithm there was a **for** loop which had to be executed for each $w$ adjacent to $v$ that has not been visited: **do** ...

What are the necessary modifications (replacing the three small dots ...) to transform the algorithm into Dijkstra's algorithm?

(b) The field $p$ in the Prim algorithm could be interpreted as the "attachment point". How can we interpret the field $p$ in the case of the Dijkstra algorithm? (It may be worth looking at part (c) before answering.)

(c) Suppose we have applied Dijkstra's algorithm, using a table $T$ with the form given in the course for Prim's algorithm. Write the pseudo-code (with the notation used in the course, in the style $T[v].d$, ..., **for**, **do, if**, ...) for a recursive procedure that prints the path to get from an origin node to the prescribed node. For example, if the origin node was $v_4$ in the graph used as an example in the course to illustrate the Prim algorithm, then *Imprimer*$(v_6, T)$ will print $v_4$ to $v_5$ to $v_7$ to $v_6$. You can suppose available a function *Write*(...) that is capable of writing out character strings and particular nodes.

4. Question 4 *(10 points)*

   (a) To do top-down insertion in a red-black tree, we said that if we have inserted a key as a red node, and the parent already has a red child (we therefore now have two red siblings), then what we should do is colour the two siblings black, and colour the parent red. Afterward, we must look at the grandparent and possibly do other things.

   But what should we do in the special case when the parent is the root (there is no grandparent)? We mostly discussed 2 of the rules that must be satisfied by a red-black tree, but in fact there are 4 rules in total: explain your answer in terms of these four rules.

   (b) A friend says to you that the Splay Tree is a terrible idea because, for example:

      i. to insert $N$ keys in order it takes $O(N^2)$ operations to create what turns out to be a simple linear list ($1 + 2 + 3 + \ldots + N = N(N+1)/2 = O(N^2)$), and then ...

      ii. ... then, to access the key 1, including doing the subsequent rotations, you have to do $O(N)$ operations, whereas it would only involve $O(\log N)$ for red-black trees.

   You give a short and precise answer to the first of these two remarks. What is your answer?

   (c) For the remark ii), your answer is a little more nuanced. What is it?

   To begin with, is your friend right? Can we point out a result that might at least be reassuring? Is this result stronger or weaker than what we could say about red-black trees in the context of an amortized analysis? Can we in fact give a result for red-black trees, of the amortized sort? Can we give a result like the ordinary analysis for red-black trees, but this time for Splay Trees?

   I am purposely being a little vague in the paragraph I just wrote, it's there only to guide you in giving your answer. You can use the $O(\log N)$ notation throughout your discussion.

**End of examination.**

Neil Stewart