CPSC 221 Written Homework #2

Due: Friday, April 1st, 2016 at 9pm

Submission Instructions

Handin your solutions using `handin`. You can write your solutions by hand and scan the pages or take pictures of them with your phone. If you choose to do this, convert the image files to `.pdf` files.[1] Or use a word processing package to typeset your solutions and produce a `.pdf` file. Do *not* submit `.doc`, `.docx` or `.pages` files!

Please split up the answers and make a separate file for each question. Your files should be named like `assignXqY.pdf` for assignment X and question Y. So the file that contains the answer to the first question of this assignment, your file should be named `assign2q1.pdf`.

If you do not follow these instructions, we may take off marks!

To handin: Copy the files that contain your solutions to the directory `~/cs221/assign2` in your home directory on a UBC CS undergraduate machine. (You may have to create this directory using `mkdir ~/cs221/assign2`.) Then run `handin cs221 assign2` from your home directory.

We encourage you to work in pairs. Please: If you are working with a partner, only one partner should handin the assignment. **Be sure to include names and ugrad login IDs of both partners on all solution pages!**

Late submissions are accepted subject to the following penalty: You lose $100 \times (2^{\lfloor m/5 \rfloor})/64$ percent of your mark, where m is the number of minutes late your assignment is. For example, if you hand in 10 minutes late, you'll lose $100 \times (2^2)/64 = 6.25\%$ of the mark, but if you hand in 25 minutes late, you'll lose 50% of the mark. (Handing in more than 30 minutes late is very dangerous to your entire course mark!)

Acknowledgments: Acknowledge all collaborators or sources of assistance (besides the course staff, handouts, and required textbooks) on the first page of your assignment by name. (If you quote from or derive work from any source, you *must* acknowledge that source at that point as is usual for citations. We don't need a formal citations list, although that's not too hard to produce with LaTeX. See the course website for details on the collaboration policy.

[1] Note that changing the file extension from `.jpg` to `.pdf` does *not* do this conversion.

Questions

1. Theory vs. Practice.

   (a) List at least 3 reasons why asymptotic analysis may be misleading with respect to actual performance in practice. (6 points)

   (b) Suppose finding a particular element in a binary search tree with 1000 elements takes 5 seconds. Given what you know about the asymptotic complexity of search in a binary search tree, how long would you guess finding the same element in a search tree with 10000 elements takes? Explain your reasoning. (3 points)

   (c) You measure the time with 10000 elements and it takes 100 seconds! List at least 3 reasons why this could be the case, given that reasoning with the asymptotic complexity suggests a different time. (6 points)

2. Proofs.

   Selection sort can be implemented as follows.

```
void selectionSort(int A[], int n) {
    for(int j = 0; j < n-1; j++) {
        int min = j;
        for(int i = j + 1; i < n; i++) {
            if(A[i] < A[min]) {
                min = i;
            }
        }
        swap(A[j], A[min]);
    }
}
```

   (a) Propose a loop invariant for the inner loop that captures what the loop is actually computing, and is sufficient (once you've proven it) to prove that the outer loop performs sorting. (10 points)

   (b) Prove that the inner loop terminates. (3 points) Hint: This should be short.

   (c) State and prove an invariant for the outer loop that is really a loop invariant, and also is strong enough so that when the function terminates, you can use the loop invariant to prove that the entire array is sorted. (10 points)

   (d) Prove that the outer loop terminates. (3 points) Hint: This should be short.

   (e) Prove that this implementation of selection sort sorts correctly. (4 points)
   Hint: This should be really short and easy given all that you've proved before.

(f) Optional: Is this implementation stable? Is this implementation in-place? (+2 Health Points and Magic Resistance for 5 turns)

3. Fourteen people are in a room. Each person is friends with 0 or more of the other people in the room. Show that there are at least two people in the room that have the same number of friends. (5 points)

   Hint: You might first try the problem assuming each person has at least one friend.

4. Draw the dictionary data structure obtained after inserting:

   $$1, 12, 111, 34, 56, 122, 342, 254, 925, 89, 210$$

   one after the other into the following initially empty structures.

   (a) A hash table of size 11 that uses chaining (with unsorted linked-list chains, insert items at the front of the chain). Use hash function $\text{hash}(k) = k \mod 11$. (5 points)

   (b) A hash table of size 11 that uses open addressing with quadratic probing. The hash function is the same as above. For each key, indicate the table slots probed. (5 points)

   (c) A hash table of size 15 that uses open addressing with double hashing. The first hash function is $\text{hash1}(k) = k \mod 15$. The second is $\text{hash2}(k) = 13 - (k \mod 13)$. For each key, indicate the table slots probed. (5 points)

5. Parallellism.

   (a) Devise a Map-Reduce implementation for searching in a sorted list. Describe the Map and Reduce steps in detail. (5 points)

   (b) How many comparisons does this version perform compared to serial binary search? How does this change with the number of processors? (5 points)

   Hint: Consider the extreme cases where we have as many processors as elements in the list and where we have only one processor.