Version: 11-14-2015 7:00 AM

CS 2223 B15 Term. Homework 3

I am aware that the midterm exam is Thursday November 19th. This homework assignment is a bit smaller in length and also has different types of questions. I believe that **completing this homework** before the exam will be a good way to **prepare** for the exam.

Homework Instructions

- This homework is to be completed individually. If you have any questions as to what constitutes improper behavior, review the examples as I have posted online http://web.cs.wpi.edu/~heineman/html/teaching_/cs2223/b15/#policies .
- Due Date for this assignment is 2PM November 20th. Homeworks received after 2PM receive a 25% late penalty. Homeworks received after 6PM will receive zero credit.
- Submit your assignments electronically using the blackboard site for CS2223. Login to **my.wpi.edu** and go to CS2223 under "My Courses" then go to "Assignments" and submit your homework under "HW3". You must submit a single ZIP file that contains all of your code as well as the written answers to the assignment.
- All of your Java classes must be defined in a packager USERID where USERID is your CCC user id (i.e., your email address without the @wpi.edu).

Q1. Stack Exercise (24 pts)

Suppose that a client performs an intermixed sequence of stack *push* and *pop* operations. The push operations put the integers 0 through 9 **in order onto the stack**; the pop operations print out the value popped from the stack. Which of the following sequences **can** occur, and which ones **cannot** occur.

(a) 4 3 2 1 0 9 8 7 6 5
(b) 4 6 8 7 5 3 2 9 0 1
(c) 2 5 6 7 4 8 9 3 1 0
(d) 4 3 2 1 0 5 6 7 8 9
(e) 1 2 3 4 5 6 9 8 7 0
(f) 0 4 6 5 3 8 1 7 2 9
(g) 1 4 7 9 8 6 5 3 0 2
(h) 2 1 4 3 6 5 8 7 9 0

For example, with the input (note that the digits 0 to 9 always must appear in order):

0 1 2 - - 3 4 - 5 - - 6 - 7 8 - - - 9 -

the resulting output is:

2 1 4 5 3 6 8 7 0 9

For the sequences that you believe cannot occur, provide a brief explanation.

Version: 11-14-2015 7:00 AM

Q2. Data Type Exercise (25 pts)

You are asked to count the duplicate words in a file that English words all in lowercase separated by spaces and new lines. Once you have processed all of the strings in the file, you must output the duplicate strings in reverse order by their frequency of occurrence, together with the number of times they appeared in the file.

For example, given the "10sonnets.txt" file (which is found in the Git Repository, and which you should copy into your local Project area), the output will start with:

```
thou (37)
thy (35)
the (31)
to (27)
and (23)
in (20)
that (19)
be (17)
of (16)
thee (16)
```

There is a **`ReportDuplicates`** class file that you are to modify to produce the above output. Your implementation must take advantage of two data types that are provided to you:

```
SequentialSearchST<String,Integer> words = new SequentialSearchST<String,Integer>();
MaxPQ<WordPair> pq = new MaxPQ<WordPair>(100);
```

The **`SequentialSearchST`** and **`MaxPQ`** implementations are only slight modifications to what you have seen in class. You should copy these classes and use them "as is". You only need to modify **`ReportDuplicates`**. The helper class **`WordPair`** is described in the sample code provided with this assignment.

In completing this problem, you need to only modify the **`ReportDuplicates`** class.

Version: 11-14-2015 7:00 AM

Q3. Algorithm Design (25 pts)

Let a[0 .. N-1] be an array of unique positive and negative integers **sorted in ascending order**. That is a[0] < a[1] < a[2] < … < a[N-2] < a[N-1].

For example, `[-5, -3, -1, 1, 4, 6, 8]` is an example of such an array of N=7 elements. Design an algorithm that computes and returns an index `idx` such that `a[idx] = idx`. If no such index exists, the algorithm should return -1. In the above example, you should return 4 because a[4]=4.

Modify the `ValueSameIndex` class to implement this algorithm.

(a) Demonstrate that it works on a number of arrays of size 8 that you choose. Make sure you validate both situations when the algorithm returns -1 and when it returns an actual index.

(b) Compute exactly the number of array inspections A(N) that you would use in the worst case to answer the problem for any N > 0. An array inspection is when you retrieve the value of a[`idx`] for a valid index `idx`.

You should be able to implement an algorithm whose performance is ~ log N.

Version: 11-14-2015 7:00 AM

Q4. Analyzing Runtime Performance (26 pts)

Continuing from the discussions in class and the reading, you are to analyze the running time of the count method as shown below (and found in **`ThreeSumFast`** in repository).

```
public static int count(Integer[] a) {
	int N = a.length;
	Quick.sort(a);

	int cnt = 0;
	for (int i = 0; i < N; i++) {
		for (int j = i+1; j < N; j++) {
			int k = rank (a, -(a[i] + a[j]));
			if (k > j) cnt++;
		}
	}
	return cnt;
}
```

Develop a table such as you see on P. 181 which includes (a) identification of statement blocks; (b) variables for time in seconds; (c) frequency analysis; (d) total time. Then have a final grand total followed by your Tilde Approximation.