HW4 Grading Rubric

Written(30)

W1(6) Weiss 5.1:

- Start at full points, -0.2pts for each incorrect hashing (max -1.4pts per part)
- -0.2 for incorrect uninsertable element or not identifying uninsertable element
- -0.4 overall for having wrong table sizes
- 0 pts if no answer given

W2(6) Weiss 5.2:

- Start at full points, -0.2pts for each incorrect hashing (max -1.4pts per part)
- -0.4 overall for having wrong table sizes
- 0 pts if no answer given

W3(6) Weiss 6.2:

For each part, (a) and (b):

- +3 if all steps shown and end heap is correct (i.e. if this is the case for both parts, 6pts are awarded)
- +2 if end heap is correct but no steps shown
- 0 if wrong end heap and no steps shown
- IF WORK IS SHOWN AND HEAP RESULT IS WRONG
    - start at +3, -0.5 for each node swap needed to correct heap
    - Min of +0.5 per part if completely wrong, but work shown, total +1 for the problem

W4(6) Weiss 6.3:

NOTE: If heap from 6.2(b) is used grade as normal and subtract 2 points form what would have been awarded.

With no or insufficient working shown:

- +4 if all three end heaps are correct, but no work shown
- +2 if final end heap is correct, one or more incorrect intermediate end heaps AND no work shown
- +2 if one more correct intermediate end heaps, final end heap wrong AND no work shown
- +1 if only final heap is shown and is correct
- 0 if all end heaps are wrong and no steps shown

With all necessary working shown - i.e., three end heaps, movement of numbers (either as arrows, or as intermediate heaps between the end heaps):

- +6 if all three end heaps (after each deleteMin()) are correct and work shown
    - -1 per error
    - Floor of 1 point with work shown

W5(6) Weiss 6.8:

Parts (a) and (c):

- +2 for correct explanation (i.e. total of +4 if both (a) and (c) are correct)
- +1 for reasonable attempt at correct explanation
- 0 for completely wrong explanation or no answer given

Part (b)

- +2 for correct explanation or proof
- +1 for small mistake in proof or explanation, but otherwise correct
- 0 for completely wrong explanation or no answer given

Programming (70)

- Compiling Issues:
- -2 Simple Fix to Compile (Removing package statements, missing semicolons)
- -4 Slightly more complicated issue (Multi line compilation errors)
- Major Issue: Max ⅔ of total grade based on attempt (compilation and/or logic errors)

P1(35):

Style (4):

- +1 File and class correctly named SpellChecker.java
- +1 For providing sample test file
- +2 Filenames taken as command line arguments

Implementation (12):

- +8 correct use of HashMap or HashTable, i.e. O(1) lookup
    - Here, using map.containsValue(word) or using a different data structure would be incorrect and would result in 6/8 pts.
- +2 correctly identifies line numbers for misspelled words
- +2 each word is correctly transformed, i.e. case insensitive and stripped of non-alphanumeric characters at the end and beginning of word (Only one punctuation in each side should be tested)
- -1 if dictionary and input text are written in wrong order (args[0] should be dictionary, args[1] should be input text)

Test Cases (19):

- +2 for each correctly identified misspelled word (+10 total)
- +2 for each correct set of spelling suggestions (+8 total)
- +1 for correctly identifying no suggestions can be found for (7) and (kasakhsahse) (+1 total)
- -1 for each word incorrectly identified as misspelled (max deduction of 10pts total)
    - In this case, no points are lost for the suggestions
- -1 for each erroneous spelling suggestion, including extra suggestions beyond the correct ones (max deduction of 10pts total)

Test Cases: ALHENA, America's, !Berlin?, Dennny, adins, fakri, 7, kasakhsahse
Suggestions: (Dennny: denny ; adins: dins, adkins; fakri: fakir ; 7: no suggestions ; kasakhsahse: no suggestions)

P2(35)

Style(3)

- +3 Filename taken as command line arguments

Implementation (14 Points)

- Uses HashMap, HashSet to calculate the frequency of characters. (2)
- Uses Priority Queue to build the Huffman Tree. (3)
- Creates an encoding table (only once). (3)
- Prompts the user for encoding/decoding. (2)
- Traverses the tree to decode. (4)

Test Cases (18 Points Total)

Test 1 - test1.txt (10 Points)

- +2 Prints a correct character table.
- Encode 1 (2): 1 character
- Encode 2 (2): 3 characters
- Decode 1 (2): 1 number
- Decode 2 (2): 2 numbers

Test 2 - test2.txt (8 Points)

- Encode 1 (2): 1 character
- Encode 2 (2): 2 characters
- Decode 1 (2): 1 number
- Decode 2 (2): 2 numbers