CSE 2010, Term Project, Fall 2019

Due Tue Nov 26 at the start of your lab section; Submit Server: class = cse2010, assignment = termProjectSxInitial

Due Thu Dec 5 at the start of your lab section; Submit Server: class = cse2010, assignment = termProjectSxFinal

x is 2 or 3—"section number" (or j for java).

The game of Boggle asks a player to identify words from a 4x4 board of letters (printed on dice). A valid word:

- has at least 3 letters,
- has successive letters that are horizontally, vertically or diagonally adjacent on the board,
- can use each letter on the board at most once, and
- is in the dictionary.

Since "Q" is usually followed by "U", "Q" on the board means "QU", which is counted as two letters. How would you design a system that can find words correctly and quickly?

The challenge of the term project is to design and build BogglePlayer, a system that can identify words from a 4x4 board. In addition to a time limit of 3 minutes, the number of returned words is limited to at most 20 and the number of points for each unique valid word is $(length(word) - 2)^2$ [corresponding negative points for invalid/duplicate words with at least 3 letters; -1 point for each word with fewer than 3 letters]. The total number of points is the sum of the word points. We desire a system that achieves more points, is fast, and has low memory usage, which can be measured by:

1. total number of points for each 4x4 board
2. time used to identify words from each board
3. number of bytes used in the memory
4. $performance = points^2/\sqrt{time * memory}$

Since this course focuses on data structures and algorithms (not concurrency/parallelism), we would like you to focus on designing/selecting data structures and algorithms to improve performance. That is, using threads is NOT desirable for this term project.

Program Files: Among your program files, you will proivde bogglePlayer.c (BogglePlayer.java) that supports at least:

- initBogglePlayer(...) [java:BogglePlayer(...)] // initialization
- getWords(...) // return found words with board positions of letters

The initialization allows you to process the data and construct data structures in preparation for identifying words. The time limit for the initialization is 10 minutes. We will provide evalBogglePlayer.c [EvalBogglePlayer.java] (not to be modified) and word.c [Word.java] (additional fields and functions/methods can be added, but existing functions/methods shouldn't be modified) to help you evaluate your system. evalBogglePlayer.c, word.c [EvalBogglePlayer.java, Word.java], and a template of bogglePlayer.c [BogglePlayer.java] are on the course website.

You may use any program segment/file from the textbook. However, you may *NOT* use program segments/files from other sources. You may borrow ideas on data structures and algorithms from other sources, but you need to clearly cite them at the top of your program files and implement them on your own.

Input: Input is from the command-line arguments for evalBogglePlayer.c [EvalBogglePlayer.java]:

- filename for a list of known words, one word per line
- optional seed for random number generator

Sample input files are on the course website. You may have additional "hardcoded" (not from command line or prompt) input data files.

Output: Measured performance by evalBogglePlayer.c [EvalBogglePlayer.java] goes to the standand output (screen).

Presentation: Project presentations (about 10 minutes each) will be during the lab on Dec 5. Create a presentation file following this outline:

1. Title, Group Name, and Members
2. Goal and Motivation
3. Initial Approach/Submission
   (a) Algorithms and supporting data structures
   (b) Additional input data and reasons
   (c) Ideas devised within the group
   (d) Ideas discussed in the course/book, cite them
   (e) Ideas borrowed from other sources, cite them
4. Final Approach/Submission
   (a) Changes in algorithms and supporting data structures
   (b) Changes in additional input data and reasons
   (c) Ideas devised within the group
   (d) Ideas discussed in the course/book, cite them
   (e) Ideas borrowed from other sources, cite them
5. Evaluation: Points, time, memory, and performance: initial vs. final approach/submission.
6. Analysis
   (a) Why more/fewer points?
   (b) Why faster/slower (including time complexity—n words, each word has m characters)?
   (c) Why more/less memory?
   (d) Possible further improvements

Evaluation:

- Initial submission (20%) – including performance and points
- Final submission (40%) – including performance and points
- Presentation (20%, oral and written)
- Teammate evaluation (20%)
- Extra Credit: Final Submission has an average performance above 100, using less than 0.5 seconds and 1.5×10^8 bytes (~150 MB), as measured by evalBogglePlayer.c [EvalBogglePlayer.java]. Average performance of at least 101 earns one extra point, at least 102 earns two extra points, ... 10 is the maximum number of extra points. Late submission is not applicable.

Submission: Initial submision has bogglePlayer.c [BogglePlayer.java], other program files, and additional data files (if any).

Final submission has bogglePlayer.c [BogglePlayer.java], other program files, additional data files (if any), and presentation (preferably in PDF).

To reduce issues on presentation day, we suggest you to send the presentation file in PDF to the TA/GSA via email (in addition to Submit Server).

Note the late penalty on the syllabus if you submit after the due date and time as specified at the top of the assignment.