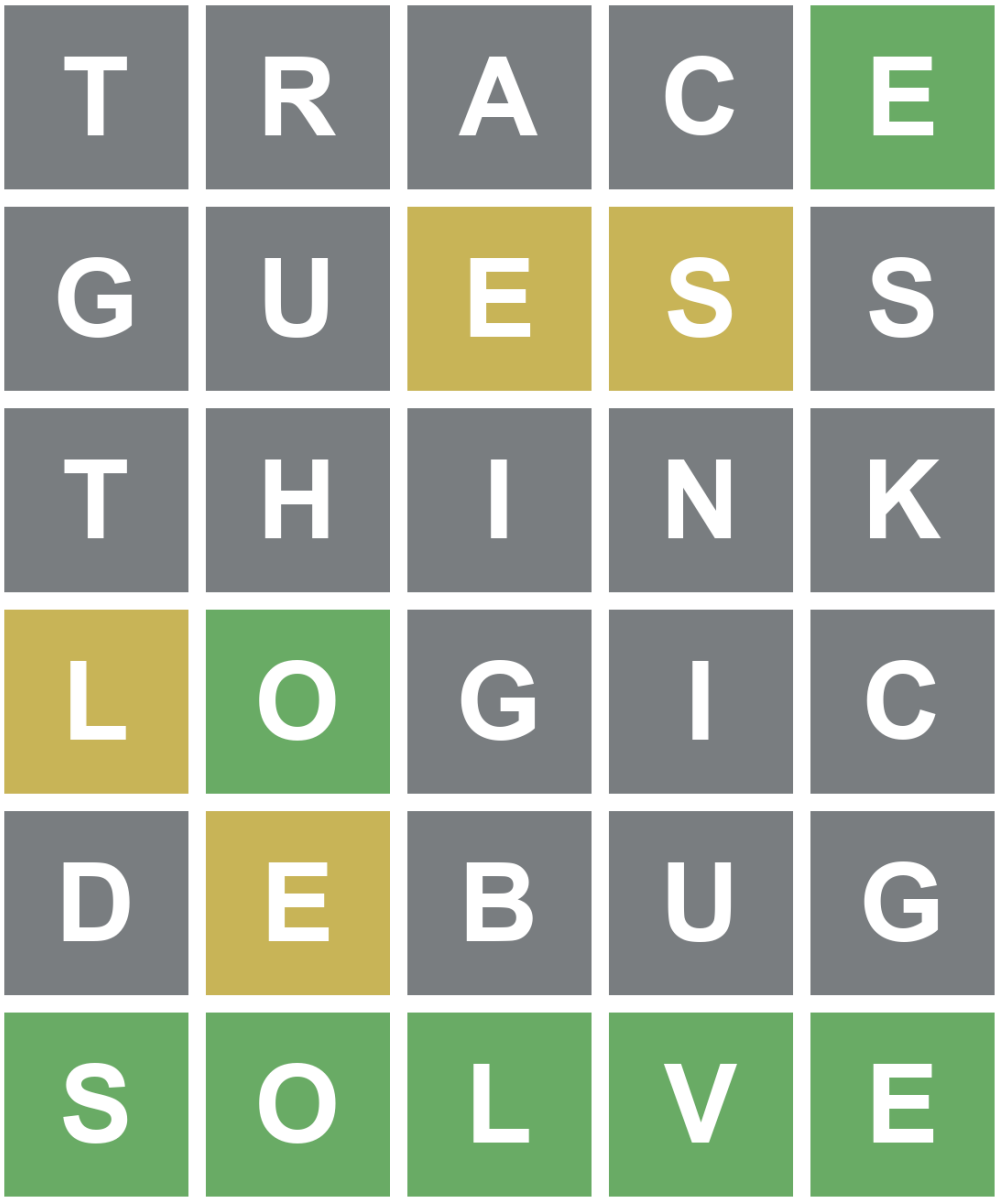
Mastering Wordle
An Algorithmic Approach
Test Format Explanation

ENGR1010J 2024FA Project Manual

Teaching Team, Fall 2024



Introduction

How is your project going? I hope you have already developed a very well algorithm now! Luckily, we will start the algorithm test from this week, November 20th to November 27th.(It can also be done in the next lab session, but you should submit all your code and no change is avaiable)

Document Format

We will have 5 files in total. Their name should be exactly the same with following format:

- `wordle.h`: The header file of the wordle game. No need to submit on this part.
- `wordle.c`: The implementation of the wordle game. No need to submit on this part.
- `player.h`: The header file of the player module. Will be provided by us and no need to submit on this part.
- `player.c`: Implement your player module in this file. You can allowed to use extern variable to get access with wordList and checkWord function. You may also define your function or global variable in this file.
- `main.c`: The main function of the program. Will include both wordle and player module. No need to submit on this part.

So what you should submit is only the `player.c` file. Please make sure that you have implemented the player module correctly and it can be compiled with the provided `player.h` file.
Here is a simple example of the `player.h` file:

```
#ifndef PLAYER_H
#define PLAYER_H

#include "wordle.h"

typedef char* (*Player)(const char lastResult[WORD_LENGTH + 1]);

char* player_AI(const char lastResult[WORD_LENGTH + 1]);

#endif
```

Here is a simple example of the `player.c` file:

```
#include "player.h"
#include "wordle.h" //You are only allowed to use checkWord function
#include <string.h>
#include <stdlib.h>
#include <stdio.h>

extern char wordList[MAX_WORDS][WORD_LENGTH + 1]; //14855 words list, use directly

char* player_AI(const char lastResult[WORD_LENGTH + 1]) {
    char* guess = NULL;
    // Your algorithm here
    return guess;
}
```

Note: return Null will terminate the program and receive a score of 0.

More importantly, you are not allowed to print anything in the player module. If you print anything, you will receive a score of 0 by our autograder program.

Test Format/Time/Space

Besides `player.c`, you should also submit a `README.md` file to explain your algorithm. The `README.md` file should include the following information:

- The algorithm you used in the player module.
- The time complexity of your algorithm.
- The space complexity of your algorithm.
- The test result of your algorithm.

These information will not affect your score, but it is very important for us to better understand your algorithm and give your a reasonable score.

For any additional space usage, i.e. file I/O, you should explain why it is necessary in the README file. Also, there would be a limit on the additional memory usage, which is 100MB

You should test how long did it take to run your algorithm on your computer. Three time should be provided in the README file. Which is the time to run the algorithm for 1000 words once, 10 times, and 100 times. Basically, we will use 100*1000 test scale to lower down the influence of "lucky". In order to prevent some students that are too lucky, we will give 1%deduction if your program cannot run on 100*1000 scale, then we will choose 10*1000 scale. If your program still cannot run on this scale, the deduction will increased to 5%. This time we will only run it once. However, if you program cannot run even once, you will receive 0 in this test. Here are some time scale for you to reference:

- run once for 1000 words: $<$ 12s, $<$ 2 min, $<$ 20 min
- run 10 times for 1000 words: $<$ 2 min, $<$ 20 min, failed
- run 100 times for 1000 words: $<$ 20 min, failed, failed

The score will be round up to integer. We will use a feishu shared document as the score board.

First of all, you should invite one TA to your feishu group(if you don't have one, please create one and invite one TA). Then you should share your `player.c` and `README.md` file in the group. Then TA will test it for you online. If some issue happens, you should make an appointment with TA to test on-site. However, TA will not help to fix the bugs, only tell you whether the format is correct or not.

What is the difference in part 1 and part 2?

Part 1 provide an interactive wordle game to user, which should be able to play with human. And you may define the solution either as random choice or user input. However, in part 2, you should implement an AI algorithm to solve the wordle game. When you test the program, you can keep some output feedback. But for our test purpose, we will prevent it for simplicity. You can also develop some other version that you can play with AI assistant, which give you some good choice.