BLG336E - Analysis of Algorithms II, Spring 2021
Homework 1

Solving Cryptarithmetic Puzzles with Breadth-First Search & Depth-First Search

Handed out: 23 March 2021, Tuesday
Due: 06 April 2021, Tuesday, 23:59

Notes:

- Please submit your homework only through Ninova. Late submissions will not be accepted.
- Please do not forget to write your name and student ID on the top of each document you upload.
- You should write your code in C++ language and try to follow an object-oriented methodology with well-chosen variables, methods, and class names and comments where necessary.
- Your code should compile on Linux using g++. You can test your code through ITU's Linux Server (you can access it through SSH).
- You may discuss the problem addressed in the homework at an abstract level with your classmates, but you should not share or copy code from your classmates or any web sources. You should submit your individual homework. In case of detection of an inappropriate exchange of information, disciplinary actions will be taken.
- If you have any questions, please contact T.A. Beyza Eken via beyzaeken@itu.edu.tr.

Overview

Breadth-First Search (BFS) and Depth-First Search (DFS) are well-known graph traversal algorithms. In this homework, you will need to implement BFS and DFS algorithms in order to solve cryptarithmetic puzzles.

Cryptarithmetic Puzzles

Cryptarithmetic puzzles are mathematical puzzles that consist of mathematical equations such as:

$$\begin{array}{cccc} & T & W & O \\ + & T & W & O \\ \hline F & O & U & R \end{array}$$

In the given puzzle, each letter is assigned to a number in the base 10 number system (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) and no two letters are assigned to the same number. The operation in the given puzzle is a summation. A solution to the given puzzle is a list of assigned numbers to letters T, W, O, F, U, R that satisfies the given summation. The solution for the given puzzle is as follows:

T: 7, W: 3, O: 4, F: 1, U:6, R: 8

$$\begin{array}{cccc} & 7 & 3 & 4 \\ + & 7 & 3 & 4 \\ \hline 1 & 4 & 6 & 8 \end{array}$$

Implementation

You are expected to implement a program that takes a puzzle as input and outputs a solution for the given puzzle. In your implementation, you should formalize the puzzle problem in a well-defined graph form and your program should use the BFS and DFS algorithms to search for a solution to the puzzle. Please consider further explanations below for your implementation.

- First, your program should find distinct letters (this part is done for you in input files). Second, it should generate a search tree with the possible mappings of distinct letters to numbers (between 0 and 9). The first layer of the tree should contain the possible assignments for the first letter, while the second layer contains the possible assignments for the second layer given the assignment of the first letter and so on. Each state of mappings will be represented as a node, and a new mapping from a letter to a number may lead to a new node.
- You can think of the state of the assignments in the form of a matrix. The start state is an empty matrix as in the left below, whereas the matrix at the right represents the solution to the puzzle. The rows in a matrix show the distinct letters in the puzzle, whereas the columns show numbers used in the 10 base number system. An empty matrix shows an unsolved puzzle. A filled matrix shows a solution to the puzzle (each row contains only one assignment of numbers).

	0	1	2	3	4	5	6	7	8	9
T										
W										
O										
F										
U										
R										

	0	1	2	3	4	5	6	7	8	9
T								**1**		
W				**1**						
O					**1**					
F		**1**								
U							**1**			
R									**1**	

- Assignments of numbers to the letters should be made in ascending order (0 to 9) during the creation of the search graph.

- While you are searching the tree by visiting new nodes, check the constraints on a node. If the node does not meet the constraints, the node will not be expanded. The constraints for the given problem:
  - Different letters should be mapped to different numbers.
  - The mapping should satisfy the summation rules given below. C_1, C_2 *and* C_3 represents the carry digits and can take the value of 0 or 1.

$$O + O = R + 10 * C_1$$

$$C_1 + W + W = U + 10 * C_2$$

$$C_2 + T + T = O + 10 * C_3$$

$$F = C_3$$

  - $F \neq 0$, $T \neq 0$, $W \neq 0$ since the TWO is a three-digit number, whereas the FOUR is a four-digit number.
- You should only focus on the summation operation on two operands in this homework. You do not need to consider other operations, e.g., multiplication (A * A = B) or conducting operations on more than two operands, e.g., A + A + A = BC.
- Your implementation could solve a puzzle with four-digit operands also such as below:

$$\begin{array}{ccccc} & S & E & N & D \\ + & M & O & R & E \\ \hline M & O & N & E & Y \end{array} \qquad \begin{array}{ccccc} & 9 & 5 & 6 & 7 \\ + & 1 & 0 & 8 & 5 \\ \hline 1 & 0 & 6 & 5 & 2 \end{array}$$

- You are not required to optimize the search procedure. Do not implement any forward checking mechanisms.
- When you find a solution, your program should stop and print the following information:
  - The number of visited nodes
  - The maximum number of nodes kept in the memory
  - The running time
  - The solution found for the given puzzle
- You are given three input files for the homework, each in the form of an empty matrix. You are expected to fill each matrix (input file) with a correct solution (your program's output). Columns of a matrix is separated by tab (\t) character and the rows are separated by line feed (\n) character.
- Sample commands to compile and run your program should follow the format below. The algorithm (BFS or DFS) to be used for the search should be able to be specified in the command.

```
g++ sourceCode.cpp -o hw1
./hw1 DFS TWO TWO FOUR outputFileName
./hw1 BFS SEND MORE MONEY outputFileName
```

- A sample output of your program should follow the below format (the numbers may not reflect the real results). The final line in the output (solution line) is optional but producing an output file that contains the solution (filled matrix) is necessary.

  ```
  Algorithm: DFS
  Number of the visited nodes: 15
  Maximum number of nodes kept in the memory: 10
  Running time: 0.34 seconds
  Solution: T: 7, W: 3, O: 4, F: 1, U:6, R: 8
  ```

Report

Please prepare a file that reports and discuss the following items:

1. Report your problem formulation:
   a. Describe node and assignment representations in detail.
   b. Write your pseudo-code.
   c. Show the complexity of your algorithm on the pseudo-code.
2. Analyze and compare the algorithm results in terms of:
   a. the number of visited nodes
   b. the maximum number of nodes kept in the memory
   c. the running time.
3. Discuss why we should maintain a list of discovered nodes? How this affects the outcome of the algorithms?