Assignment 1

Artificial Intelligence

August 12, 2020

Written Component: Please submit a PDF containing your solutions called LastName-FirstName-A1.pdf, where you replace LastName and FirstName with your last and first names. You may write out your answers on paper, take pictures of your work, compile them into a PDF and submit in that manner if you prefer not to type out the answers.

Programming Component: Please submit a .py or .java file containing your solutions called LastName_FirstName_Driver, where you replace LastName and FirstName with your last and first names.

The POST CORRESPONDENCE PROBLEM is defined as follows: You are given a collection of dominoes. Each domino has a string of characters on top and another string on the bottom. (Both strings are non-empty.) You can make as many copies of the dominoes as you need. The problem is to place the dominoes side by side so that the combination of the strings on the top is the same as the combination of the strings on the bottom.

Example:
Consider the following instance of the problem with four dominoes that will be used for various parts of the assignment.

1. D1 = $\frac{c}{cca}$
2. D2 = $\frac{ac}{ba}$
3. D3 = $\frac{bb}{b}$
4. D4 = $\frac{ac}{cb}$

Then the sequence D3, D2, D1, D4, D3 spells out "bbaccacbb" on both the top and bottom of the dominoes.

The Post Correspondence problem is not Decidable, but it is Recognizable. This means that there is no algorithm that is guaranteed to terminate when trying to find a solution whether a solution exists or not. However, if a solution exists, it is possible to find one using exhaustive search.

In the following problems, our goal is to explore this problem further using what we have learned about search and to gain insight into subtleties that have to be taken into account when designing solutions to these problems.

1. **Question 1**
   For this question, let us characterize the state space as follows:

   - A state is sequence of dominoes where either the top string is a prefix of the bottom string or the bottom string is a prefix of top string. For example, {D3, D2} = $\frac{bbac}{bba}$ is a valid state because "bba" is a prefix of "bbac", while {D1, D4} = $\frac{cac}{ccacb}$ is not a valid state.
   - The only we can take is to add a domino from the current
   - The start state is an empty sequence of dominoes.
   - A goal state is any sequence of dominoes where the top and bottom strings are equal.

   (a) Solve the example instance of the Post Correspondence given in the introduction using BFS search. Show the state space that is generated. When taking an action, test adding the dominoes in numerical order (i.e. try D1 first, then D2 and so on).

   (b) Solve the example instance of the Post Correspondence given in the introduction using DFS search. Show the state space that is generated.

(c) Explain that if the order we tested adding dominoes was different (not numerical order), DFS will not find a solution. Show the state space until at least depth 4 for the DFS search if the order the dominoes was tested in was inverse numerical order.

2. Clearly, for any set of dominos, the state space has no cycles, since adding another domino always makes the top and bottom strings longer. However, for some sets of dominos, the state space may not be a tree. Give an example of a set of dominos in which the state space is not a tree.
   Hint: Your answer should be a graph where two nodes have an outgoing edge to a single node.

3. There are many ways to approach problems using search. Sometimes a small tweak in the way states are encoded or search is performed can make the difference in the algorithm finding a solution. For this question, we will revisit the example instance of the Post Correspondence problem using an alternative state space. Instead of building strings by adding dominoes to the end of our sequence. We can add dominoes to the start of our sequence. This way a valid state is when the bottom string is a suffix of the top string or the top string is a suffix of the bottom string. As in Problem 1, show the state space generated using breadth-first search to solve the example problem with this alternative approach.

4. A cleverer state space for this problem defines a "state" to be either the part of bottom that goes past the top, or vice versa. For example, if you have the dominoes D1=$\frac{abb}{ab}$, D2=$\frac{a}{baab}$, D3=$\frac{ac}{acab}$, then the two strings {D1,D2}=$\frac{abba}{abbaab}$ and {D3}=$\frac{ac}{acab}$ can be considered the same state, because in either case you have to "make up" a trailing "ab" on the bottom, and any sequence that can be added after {D1,D2} to solve the problem will work just as well when added after {D3}. Construct an example of a sequence of dominoes whose state space, defined this way, has a cycle.

Programming Component

Programming Component: Please submit either a Python or Java file that contains your program that fulfills the requirements detailed below. You should also include in your submission a README file with quick overview of how to run your program and anything the grader should be aware of.

- **Assignment**
  Write a program that tries to solve the Post Correspondence problem according to the following algorithm.
  - In the first stage of search, use a breadth-first search through the state space detailed in problem 4 from the written component until the queue of frontier states has reached a specified maximum. You should use a HashMap or Dictionary over strings to eliminate repeated states.
  - In the second stage of search, use iterative deepening starting from the frontier created in the first stage, until a specified limit has been reached.
- **Input**
  Your code should accept the following inputs:
  - The maximum size of the queue (the frontier) used in the breadth-first search.
  - The value of some kind of parameter (e.g. maximum depth; maximum number of states; maximum run time) that will to prevent the program from going into an infinite loop. Note that the program can go into an infinite loop either in stage 2 or in stage 1, if the search can proceed without exhausting the queue, so the limit must apply to both stages.
  - The set of dominos.
  - (*) A flag indicating the type of output, as described in the next section (Output).

  Use a simple format for inputting, but do not hard code any aspect of the input.
- **Output**
  The program should always output one of the following three:
  - A sequence of dominos that solve the problem.
  - A flag indicating that no solution exists.
  - A flag indicating that no solution was found within the limits of search.
    In addition if the flag in (*) above is set, the program should output the sequence of states generated in searching for the solution.
- **Grading:**
  8 points for correctly running code and 2 points for well-written code for a total of 10 points on the programming component. The written component and programming component are weighed equally for this assignment.

- **Sample Inputs** Format:

  First line: max size of queue
  Second line: Max total number of states
  Third line: either 0 or 1 (refers to (*) input in Input section).
  Fourth line: the number of Dominoes
  Remaining lines: Dominoes

  See Canvas for actual sample input files.