Week 5: Lab 4

Lab 4

A warning about GenAI: Many of the problems given in the first few labs are classical problems with well known solutions. They are well known because they are useful learning problems. This means tools like ChatGPT and Copilot are almost certain to have been trained on similar problems. This means they are easily able to solve many of these problems. While this is useful sometimes, using these solutions will prevent you from learning what the labs are trying to teach you. Even worse, later labs will build on prior labs and increase in novelty and difficulty. Generative AI tools will not be able to solve them, and neither will you if you didn't understand the prior labs. We strongly suggest solving these problems yourself and writing your own code. You are free to use Gen AI, but you should consider it to be the similar to Googling the answer or asking your lab facilitator--only use it if you are stuck and make sure you understand the solution it produces.

This lab covers several problems that can be solved with DFS and BFS. However, these problems don't have an obvious graph. Instead, there is an "implicit graph". This means that it is up to you to decide how to model the problem as a graph problem and then apply the right graph algorithm.

Mountain Waterfall

Go to the DOMjudge server (http://domjudge.gozz.au/) select lab4 and look at "mountainwaterfall". This problem can be viewed as a graph problem. For each location in the grid, make a node. Connect a node to it's immediate left and right neighbours if they are empty. Also, connect a node to it's neighbour directly below if it is empty. Now, a DFS from each initial water node will find all the nodes that are covered by water. Notice how the problem contains no explicit graph but instead we needed to model the solution by building a graph that represents our problem? This allowed us to use DFS to solve the problem.

Hint: Don't forget about using a visited set, otherwise your DFS may revisit the same node many times and run slowly!

This problem was taken from the South Pacficic ICPC 2020 Divisionals.

Go Stone Puzzle

Next, look at "gostonepuzzle". At first, this problem may seem like it has nothing to do with graphs. However, observe that it is asking for the minimum number of operations. Recall that BFS can find the minimum number of steps to get to a location in a graph. Can you try to find an implicit graph structure that models that problem and allows you to find the answer using BFS?

Hint: Treat every arrangement of stones as a string of length N+2. You can use some character like "E" to represent an empty location. Make a graph where every possible string is a node and there is an edge between nodes if an operation can be done to get from one string to another.

Compleixty analysis: What is the complexity of your algorithm? How many nodes and edges does your graph have?

This problem was adapted from AtCoder.

Chameleon Maze (optional challenge question)

This is a challenge question and most students aren't expected to solve it.

Next, look at "chameloenmaze". Your target complexity for this problem is O(r*c+k). This problem will require a non-trivial implict graph then a BFS on that graph. We can turn the grid into a graph by connecting neighbouring cells together. However, this is not enough. We need a way of keeping track of which candy we are currently up to, since they must be collected in order.

This problem was taken from the South Pacficic ICPC 2021 Divisionals.