CS2040S: Data Structures and Algorithms

Discussion Group Problems for Week 10

For: March 23–March 27

Below is a proposed plan for tutorial for Week 10. In Week 10, we will continue talking about graph modelling, and continuing to look at shortest path problems. (This week we will mostly focus on problems that can be solved via Bellman-Ford, while next week we will talk about Dijkstra's algorithm.)

Problem 1. (Exploring all Paths)
Relevant Kattis Problem: https://open.kattis.com/problems/beepers

Sometimes, it seems like a good solution to a graph problem is to explore all possible paths in the graph. Starting from some vertex s, we explore all possible paths to some destination t, and examine the resulting cost. For example, imagine you have a highway road map, with tolls specified along various roads. You want to find the route with the smallest cumulative toll.

Perhaps we can explore all possible routes using a cleverly modified DFS or BFS? Why is this a bad idea? Draw a graph in which exploring all possible paths from s to t will not work well.

Problem 2. Overcooked
Relevant Kattis Problems:

- https://open.kattis.com/problems/pickupsticks
- https://open.kattis.com/problems/reactivity
- https://open.kattis.com/problems/easyascab

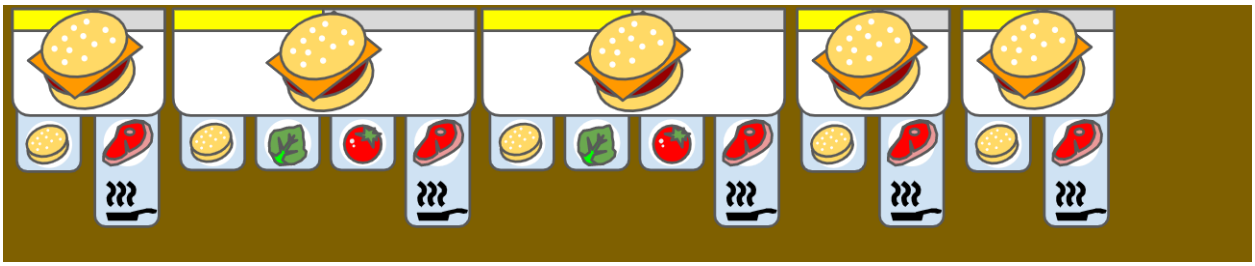

Figure 1: *Overcooked.* (Matthew Ng Zhen Rui)

After many years of playing Overcooked, you finally have fulfilled your dream of opening your own restaurant. On the opening day you are given a list of reviewers that will be coming. You want to ensure that they are served in a timely manner so that they leave good reviews for your restaurant.

Problem 2.a. Firstly, you want to determine in what order you want to serve your dishes. There are a lot of dishes to serve out but you can't serve them in any order. Given a set of n dishes you know certain foods must be served after another (for example, you will serve appetizers like mushroom soup before the main course like filet mignon). Given a list of n dishes as well as k constraints on relative orderings of the dishes, output a valid sequence in which the dishes can be served. You may assume that such an ordering always exists.

In addition, you want to ensure that if there are multiple possible ways to order the dishes, you want output the one which is *lexicographically* the smallest (So that it looks presentable on a menu).

As an example, let's say there were 4 dishes: Garlic Bread, Mushroom Soup, Filet Mignon Burger and Banana Split. Now, if we are given the constraints as follows:

- Garlic Bread must be served before Filet Mignon Burger
- Mushroom Soup must be served before Filet Mignon Burger
- Banana Split must be served after everything else

These give the following valid orders:

- Garlic Bread, Mushroom Soup, Filet Mignon Burger, Banana Split
- Mushroom Soup, Garlic Bread, Filet Mignon Burger, Banana Split

However, we will want to output the former, since it is lexicographically smaller than the latter (since Garlic Bread is alphabetically before Mushroom Soup).

Optional : In order to protect against potential modifications of the order of dishes being served, you also want to check whether any valid ordering of the dishes is unique. Note that if this is the case then the valid order will instantly be the smallest lexicographically as well. How would you do this?

Problem 2.b. You quickly realise that based on your chef's schedules, that you actually need to precisely time when each dish will be served to the guests! The situation is the same as before, but you now also need to assign each of n the dishes to the n time slots there exists where each dish can be served. Based on how long a dish remains fresh after being prepared, you have determined intervals where each dish can be served, where each interval is a contiguous series of slots within 1 to n. (Example: Mushroom Soup must be served between slots 3 to 10, otherwise it will be too cold). All the intervals are inclusive of their endpoints. Note that you are also NOT allowed to serve two dishes at the same time. Hence, each timeslot will only allow to serve 1 dish. Determine whether you can give a schedule where each dish will be served, and if so, output that schedule.

Problem 3. (Weighted Diameter of a Graph)
Relevant Kattis Problems: https://open.kattis.com/problems/allpairspath

The diameter of an unweighted graph is the maximum length "shortest path" between any two nodes. In a weighted graph, the weighted diameter is also the maximum length "shortest path" between any two nodes (but now the weights of the edges matter). Give an algorithm for finding the weighted diameter of a weighted graph.

Problem 4. (Internet Routing)
https://open.kattis.com/problems/shortestpath3

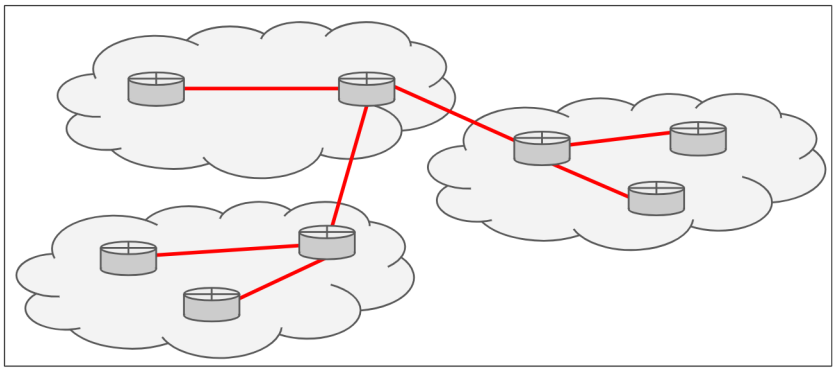

Figure 2: A Network Diagram

Imagine a hypothetical world where there are a set of computers connected by wires called *the internet*. Each of these computers has a unique address that is used for routing. (We might call this an IP address.) Each computer needs to maintain a routing table, i.e., a table that maps destination addresses to the next hop of the route. For example, if computer X has links to computers A, B, and C, and if it receives a message for destination D, the routing table might specify that that message should be forwarded to B. The goal is to forward the message to the destination using the minimum number of hops possible.

Problem 4.a. Collectively, the routing tables specify shortest path trees. How might you use **Bellman-Ford** to construct the routing tables?

Problem 4.b. Now imagine we want to construct the routing tables in parallel. That is, imagine that all the computers can send information to their neighbors at the same time. How might you implement a relax step so that all the computers can run it at once? Will that work? (Why or why not?) How long will the entire Bellman-Ford execution take?

Problem 4.c. Imagine that the computers run Bellman-Ford relax rounds forever. E.g., every one minute they send their vector of estimates to their neighbors and update their routing tables

as necessary. Assume that a new edge is added to the network, e.g., A and B were not neighbors and now they are. Will the ongoing algorithm fix the estimates? If so, how long will it take? If not, what needs to be done to make it work?

Problem 4.d. Imagine that the computers run Bellman-Ford relax rounds forever. E.g., every one minute they send their vector of estimates to their neighbors and update their routing tables as necessary. Assume that one edge fails in the network, e.g., A and B were neighbors and now they are not anymore. Will the ongoing algorithm fix the estimates? If so, how long will it take? If not, what needs to be done to make it work?

Problem 5. (Tourism)
Relevant Kattis Problem: https://open.kattis.com/problems/maximizingwinnings

You are off to travel the world in your trusty old beat-up Chevrolet. You have a map, a full tank of gas, and you are off. Just as soon as you figure out where you want to go. Looking at the map, you notice that some roads look very appealing: they will make you happy with their winding curves and beautiful scenery. Other roads look boring and depressing: they will make you unhappy with their long straight unwavering vistas.

Being methodical, you assign each road a value (some positive, some negative) as to how much happiness driving that road will bring. You may assume that the happiness is spread out uniformly over the entire road, for every road on your map. Pondering a moment, you realize that you can only travel a fixed number k kilometers. Starting from here on the map, find the destination that is exactly k kilometers away that will bring you the most net happiness. (Write out your algorithm carefully. There is a subtle issue here!)

Problem 6. (Spaceship Troubles)

The wonderful, fantastic product made by the Whoozit Company is a newfangled spaceship with a fancy warp drive. Now, warp drives are not quite as useful as you might think: they only let you travel between a set of fixed locations, along fixed (directed) paths. One of the possible jobs at Whoozit Company is *Guinea Pig*, and you have just been promoted. They hand you a map of the universe (complete with locations and directed paths), put you in the new spaceship, and off you go.

At first, everything works fine. You start off at the little dot marked "Earth", chose one of the neighboring dots ("Alpha Centauri", at only 4.24 light years away), and hit the big green *GO* button. Whoosh. You made it in one piece! Wow, each hop takes exactly 1 minute, regardless of how far you go!

But then a little red light starts blinking, and the screen reads out the following error mes-

sage: `Error: your warp drive is now broken. Abort, Retry, Fail?` After some fussing with the computer, you discover that the warp drive still works, but it has the following limitation: it can only jump five hops at a time. You cannot just jump to a neighbor of Alpha Centauri; you have to jump directly to some location that is exactly five hops away.

Describe an algorithm that will get you from Alpha Centauri back to Earth in minimum number of hops, explain why it works, and give its efficiency.

Problem 7. (Course Correction)

Remember when you first entered university and you were overwhelmed with all the different modules that were offered to you, hence leaving you confused as to how to do your module planning? Well now with the knowledge you have gained, you are better equipped to tackle this problem. After researching all the courses offered closely, you have extracted a set of courses that you wish to take, and have noted their required prerequisites. You are not allowed to take a course with one of its prerequisites in the same semester, and courses may have multiple prerequisites such that you are required to clear all the previous prerequisites before you can take that particular course. You may assume that there are no requirements for modules to be taken together, and there are no workload limits (i.e. you can take as many modules as you want in a single semester). Given this information, what's the minimum time you will take to clear all these courses?