**CS2040S: Data Structures and Algorithms**

Discussion Group Problems for Week 10

*For: March 24–March 28*

# Problem 1. DFS/BFS

**Problem 1.a.** Recall that when performing DFS or BFS, we may keep track of a parent pointer that indicates the very first time that a node was visited. Explain why these parent edges form a tree (i.e., why there are no cycles).

**Ans:** A total of *n* – 1 new nodes will be discovered starting from the source node. With each discovery, the parent pointer can be thought of as an edge. The final structure will consist of *n* nodes and *n* – 1 edges and will form one connected component. The resultant tree is sometimes known as a BFS/DFS-spanning tree.

**Problem 1.b.** Remember we learned about algorithms called DFS and BFS on trees? Trees are just a type of graph. How does DFS or BFS on a tree relate to DFS or BFS on a graph? Are they the same algorithm? Will the DFS/BFS algorithm we learned about for trees work on a graph? Will the DFS/BFS algorithm we learned about for graphs work on a tree? What happens if you run DFS/BFS on a tree but do not start at the root?

**Ans:** Yes, BFS/DFS algorithm for graphs would work on trees as trees are simple graphs without any cycles, and thus it would be impossible to visit the same node twice. However, the BFS/DFS algorithm for trees would not work on graphs. It is perfectly fine to run BFS/DFS on a non-root node on a tree.

# Problem 2. Graph components

## (Relevant Kattis Problem: [https://open.kattis.com/problems/countingstars/)](https://open.kattis.com/problems/countingstars/)

Given an undirected graph *G* = (*V,E*) as an adjacency list, give an algorithm to: (i) determine if the graph is connected; (ii) return the number of connected components (CC) in the graph.

**Ans:**

1. Pick any node and perform BFS/DFS on it. If the graph is connected, we should be able to traverse to every node. If there are still nodes that have yet to be traversed, it means that the graph is not connected
2. Pick any node and perform BFS/DFS on it. Once the traversal is complete, if there are still nodes that have yet to be traversed, pick that node and perform BFS/DFS on it again. For each time this process is repeated, it means that there is one more connected component in the graph.

**Problem 3. Is it a tree?**

## (Relevant Kattis Problem: [https://open.kattis.com/problems/flyingsafely/)](https://open.kattis.com/problems/flyingsafely/)

Assume you are given a connected graph with *n* nodes and *m* edges as an adjacency list. (You are given *n* but not *m*; assume each adjacency list is given as a linked list, so you do not have access to its size.)

Give an algorithm to determine whether or not this graph is a *tree*. Recall that a tree is a connected graph with no cycles. Your algorithm should run in *O*(*n*); particularly, it should be independent of *m*. Assume *O*(*n* + *m*) is too slow.

**Ans:** Any connected, undirected graph containing *n* nodes and > *n* – 1 edges has a cycle. Thus, simply count edges in the graph, stopping when you find at least *n* edges. Notice though that since the graph is given as an adjacency list, each edge is represented twice: in the list of both the source and the destination. Thus, if you find > 2*n* – 2 edges in the adjacency list, you know there is a cycle, and the graph is not a tree. Otherwise, since we are given that the graph is connected, and have verified that is has *n* – 1 edges, it is a tree.

If we are not given that the graph is connected, the same approach can be used. After verifying that it has *n* – 1 edges, simply modify it to run DFS to check whether all nodes can be reached and that there is no cycle.

# Problem 4. Graph modeling

Here are a bunch of problems. How would you model them as a graph? (Do not worry about solving the actual problem. We have not studied these algorithms. Just think about how you would model it as a graph problem.) Invent some of your own problems that can be modelled as graph problems—the stranger, the better.

**Problem 4.a.** Imagine you have a population in which some people are infected with this weird virus. For any two patients, you want to decide whether the infection might have spread from one to the other.

1. First, you can assume that everyone who passes on the infection is symptomatic and via testing, you can tell that they were infected.
2. Now, what if the virus may be passed by some asymptomatic people (who do not test positive)? You can assume that the virus is not passed too often by such people, so any chain of infections will only include a few asymptomatic cases.

**Ans:** We can model the people as nodes and add edges between people who have been in contact with others. We can then colour the nodes as red if they have the virus and blue if they are healthy/do not have the virus. We then just have to find a path between the two individuals which contains the fewest blue nodes (healthy individual).

**Problem 4.b.** Imagine you have a population in which some few people are infected with this weird virus. You also have a list of locations that each of the sick people were in each of the last 14 days. Determine if any of the sick people ever met.

**Ans:** We can model the people as nodes and every (location, day) pair as a node. This means that we have 14 nodes per location, each representing the same location, but on a different day. We can then add edges between people and the (location, day) node that they visited. This represents visiting that location on a specific day. Now, we can look at any (location, day\_ node that has a degree > 1 – this would show the people that have met if they visited the same location on the same day.

**Problem 4.c.** You are given a set of jobs to schedule. Each job *j* starts at some time *sj* and ends at some time *tj*. Many of these jobs overlap. You want to efficiently find large collections of non-overlapping jobs so that you can assign each collection to a single server.

**Ans:** We can represent each node as a job. Add an edge between two nodes if the respective jobs overlap. Now, we need to find collections of nodes that are not neighbours.

**Problem 4.d.** An English professor complains that students in their class are cheating. The professor suspects that the cheating students are all copying their material from only a few different sources, but does not know where they are copying from. Students that are not cheating, on the other hand, all submit fairly different solutions. How should we catch the cheaters?

**Ans:** We can model every student’s essay as a node. Add an edge between two nodes if their respective essays are similar. A cluster of nodes all connected to each other likely indicates cheating.

**Problem 4.e.** There are *n* children and *n* presents, and each child has told you which presents they want. How do we assign presents to children?

**Ans:** Each child is a node, and each present is a node. We add an edge connecting a child to a present if that present is acceptable to that child. Now, we find a set of edges that do not share any endpoints.

# Problem 5. Word games

## (Relevant Kattis problem: [https://open.kattis.com/problems/sendmoremoney/)](https://open.kattis.com/problems/sendmoremoney/)

Consider the following two puzzles:

* Puzzle 1:

S E N D

+ M O R E

----------

M O N E Y

* Puzzle 2:

F O R T Y

T E N

+ T E N

------------

S I X T Y

In each of these two puzzles, you can assign a digit (i.e., {0*,*1*,*2*,*3*,*4*,*5*,*6*,*7*,*8*,*9}) to each of the letters in the puzzle such that the given equation holds. (Each digit is only assigned to one letter.) The goal is to solve these puzzles. How should you model and solve these puzzles? What is the running time of your solution? Can you optimize your solution to find the answer more quickly, most of the time?

**Problem 5.a.** Explain how to model the problem as a graph search problem. What are the nodes? How many nodes are there? What are the edges? Where do you start? What are you looking for?

**Problem 5.b.** To solve this problem, should you use BFS or DFS? Why? How else can you make it run faster?

**Problem 5.c.** When does your search finish? Can you optimize the algorithm to minimize the amount of searching?

# Problem 6. Good students, bad students

## (Relevant Kattis problem: [https://open.kattis.com/problems/amanda/)](https://open.kattis.com/problems/amanda/)

There are good students and bad students[[1]](#footnote-1). And at the end of every year, we have to divide the students into two piles: *G*, the good students who will get an A, and *B*, the bad students who will get an F. (We only give two grades in this class.)

To help with this process, your friendly tutors have each created a set of notecards. Each card contains the names of two students. One of the two names is a good student, and the other is a bad student. Unfortunately, they do not indicate which is which.

Since the notecards come from thirty eight different tutors, it is not immediately certain that the cards are consistent. Maybe one tutor thinks that Humperdink is a good student, while another tutor thinks that Humperdink is a bad student. (And Humperdink may appear on several different cards.) In addition, the tutors do not provide cards for every student.

Assume you can read the names on a card in *O*(1) time and that there are more good students than bad students.

Devise an algorithm to determine the answers for the following questions:

* Are the notecards consistent, i.e., is there *any* way that we can assign students to *G* and *B* that is consistent with the cards? Note that there must be more good students than bad students.
* Are the notecards sufficient (to make *any* conclusion)? i.e., can we either deduce that the notecards are inconsistent (and so, there’s **no** way to assign the students to the sets *G* and *B*) or can we find **exactly one** possible way to assign students to the sets?
* Assuming that the notecards are consistent and sufficient, determine which set each student belongs to.

**Ans:**

**Problem 7. Gone viral** (*Optional, more challenging*)

There are *n* students in the National University of Singapore. Among them, there are *n* − 1 friendships. Note that friendship is a symmetric relation, but it is not necessarily transitive.

Any two people in the National University of Singapore are either directly or indirectly friends. Formally, between any two different people *x* and *y*, either *x* is friends with *y* or there exists a sequence *q*1*,q*2*,...,qk* such that *x* is friends with *q*1, *qi* is friends with *qi*+1 for all *i < k* and *qk* is friends with *y*.

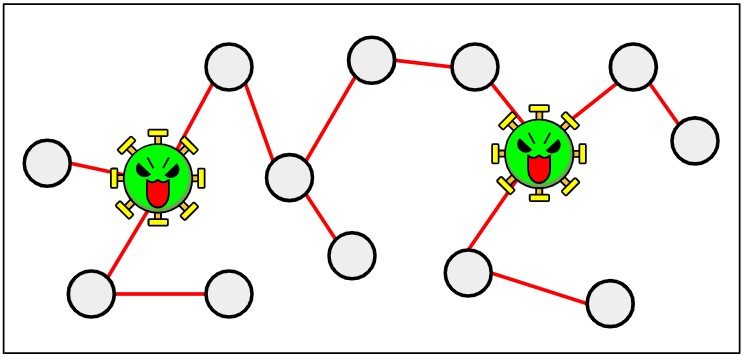
It was discovered today that **two** people were found to have the flu in the National University of Singapore.

Every day, every person can meet with **at most one friend.** When these two people meet, if exactly one of them has the flu, it will be transmitted to the other.

Give an *O*(*n*log2 *n*) algorithm to determine the minimum possible number of days before it is possible that *everyone* has the flu.

Hint: First, solve the case where there is only a single person was infected at the start in

*O*(*n*log*n*)



**Figure 2:** *Gone Viral.* (Matthew Ng Zhen Rui)

1. No, not really. This sort of binary distinction is silly. [↑](#footnote-ref-1)