Homework 2

CS 4104 (Fall 2021)

Assigned on September 9, 2021.
Submit a PDF file containing your solutions on Canvas by 11:59pm on September 16, 2021.

Instructions:

- The Honor Code applies to this homework with the following exception:
  - You can pair up with another student to solve the homework. Please form teams yourselves. Of course, you can ask the instructor for help if you cannot find a team-mate. You may choose to work alone.
  - You are allowed to discuss possible algorithms and bounce ideas with your team-mate. **Do not discuss proofs of correctness or running time in detail with your team-mate. You must write down your solution individually and indepedently. Do not send a written solution to your team-mate for any reason whatsoever.**
  - *In your solution, write down the name of the other member in your team. If you do not have a team-mate, please say so.*
  - Apart from your team-mate, you are not allowed to consult sources other than your textbook, the slides on the course web page, your own class notes, the TAs, and the instructor. In particular, do not use a search engine.
- Do not forget to typeset your solutions. *Every mathematical expression must be typeset as a mathematical expression, e.g., the square of n must appear as* n^2 *and not as "n^2".* You can use the LaTeX version of the homework problems to start entering your solutions.
- Do not make any assumptions not stated in the problem. If you do make any assumptions, state them clearly, and explain why the assumption does not decrease the generality of your solution.
- You must also provide a clear proof that your solution is correct (or a counter-example, where applicable). Type out all the statements you need to complete your proof. *You must convince us that you can write out the complete proof. You will lose points if you work out some details of the proof in your head but do not type them out in your solution.*
- If you are proposing an algorithm as the solution to a problem, keep the following in mind (the strategies are based on mistakes made by students over the years):
  - Describe your algorithms as clearly as possible. The style used in the book is fine, as long as your description is not ambiguous. Explain your algorithm in words. A step-wise description is fine. *However, if you submit detailed code or pseudo-code without an explanation, we will not grade your solutions.*
  - Do not describe your algorithms only for a specific example you may have worked out.
  - Make sure to state and prove the running time of your algorithm. You will only get partial credit if your analysis is not tight, i.e., if the bound you prove for your algorithm is not the best upper bound possible.
  - You will get partial credit if your algorithm is not the most efficient one that is possible to develop for the problem.
- In general for a graph problem, you may assume that the graph is stored in an adjacency list and that the input size is $m + n$, where n is the number of nodes and m is the number of edges in the graph. Therefore, a linear time graph algorithm will run in $O(m + n)$ time.

Problem 1 (5 points) Consider the version of BFS where we store both the node index and its distance from the starting node s in the queue L. In other words, when we want to push a node v into L, we actually push the pair $(v, d(s, v))$. Now suppose that u and w are two consecutive nodes that we pop from L. State and prove the relationship between $d(s, u)$ and $d(s, w)$.

Problem 2 (25 points) This problem is in three parts.

(a) (10 points) If G is an undirected graph and v is a node in G, let us use $G-\{v\}$ to denote the graph formed by deleting v and all the incident edges on v from G. In general, even if G is connected, it is possible that $G - \{v\}$ may contain two or more connected components, e.g., if G is a tree and v is a node with degree greater than one. However, this fragmentation cannot happen for every node v. Prove this fact, i.e., prove that in every connected, undirected graph G, there is at least one node v such that the graph $G - \{v\}$ is connected. *Hint:* In the special case that G is a tree, any leaf can serve as v. How will you find a "leaf" in a general undirected graph?

(b) (10 points) Now prove that if G is an undirected, connected graph on n nodes that G must contain at least $n - 1$ edges. *Hint:* Use the statement in part (a) in combination with mathematical induction. Your proof must clearly state the quantity over which you are inducting (e.g., the number of nodes n or the number of edges m or something else that may be appropriate to the problem) and prove the base case, state the inductive hypothesis, and then prove the inductive step.

(c) (5 points) Finally, prove that if G is an undirected, connected graph on n nodes that contains exactly $n - 1$ edges, then G does not contain a cycle. *Hint:* The statement you proved in part (b) may be helpful.

Problem 3 (20 points) Given an undirected graph G and an edge e in this graph, develop a linear time algorithm to determine if there is a cycle in G that contains e.

Problem 4 (20 points) Solve exercise 7 in Chapter 3 (page 108–109) of your textbook. I provide the essential part of the problem here.

> *Claim: Let G be an undirected graph on n nodes, where n is an even number. If every node of G has degree at least $n/2$, then G is connected.*

Decide whether you think the claim is true or false, and give a proof of either the claim or its negation.

Problem 5 (30 points) Solve exercise 9 in Chapter 3 (page 110) of your textbook. Suppose that an n-node undirected graph G contains two nodes s and t such that the distance between s and t is strictly greater than $n/2$. Show that G must contain some node v, not equal to s or t, such that deleting v from the graph destroys all s-t paths. (In other words, the graph obtained from G by deleting v contains no path from s to t.) Give an algorithm with running time $O(m + n)$ to find such a node v. *Hint:* At least one of the layers in the BFS tree rooted at s has a special property.