Timur Garifullin
Due: 11:59pm Thursday, Feb 2

Homework 2

Spring 2023, Bhakta
CS 315, Algorithms

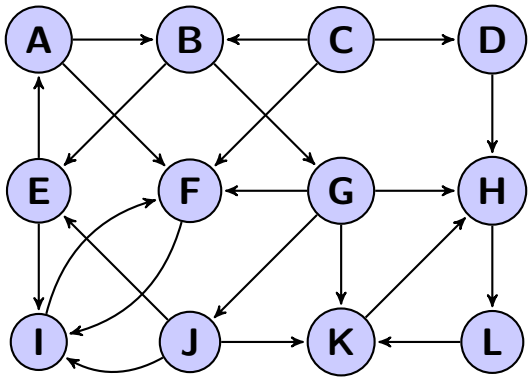
This assignment covers Depth First Search, connected components, and strongly connected components.

0. (0 points) (NOT OPTIONAL) Who did you work with on this homework? What sources did you consult? How long did this assignment take?

Solution: Consulting: Artur, Konstantino
Time Spent: 18 hours

Homework 2

1. (25 points) Run the strongly connected components algorithm on the following directed graph G. When doing DFS, whenever there is a choice of vertices to explore, always pick the one that is alphabetically first.



   (a) (10 points) Label the pre/post numbers from the first DFS run on the reverse Graph.

   Solution:
   (pre/post numbers)
   A: (1/12)
   B: (3/6)
   C: (4/5)
   D: (13/14)
   E: (2/11)
   F: (15/18)
   G: (8/9)
   H: (19/24)
   I: (16/17)
   J: (7/10)
   K: (20/23)
   L: (21/22)

   (b) (10 points) In the second run of DFS, list all the vertices in the order they are visited, and the SCCs in the order they are identified.

   Solution:
   Vertices: A, B, C, D, E, F, G, H, I, J, K, L
   SCCs:
   (A, E, J, G, B)
   (H, K, L)
   (F, I)
   (D)
   (C)

   (c) (5 points) Draw the “metagraph" (each meta-node is a SCC of G).

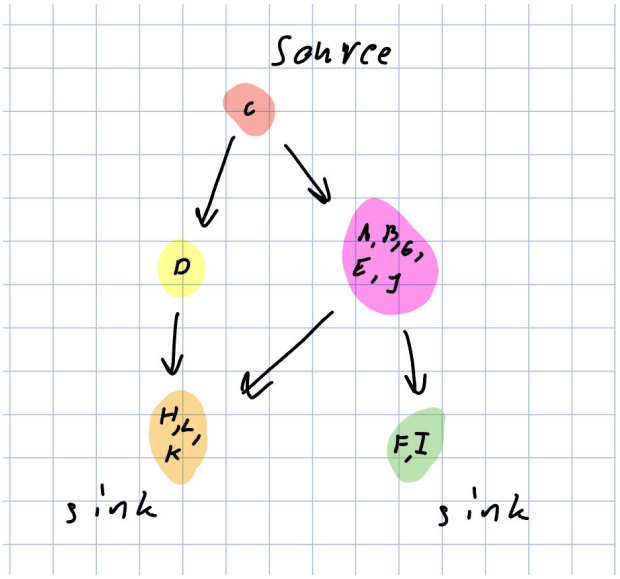


Figure 1: Metagraph for part c)

Solution: Graph above.

2. (15 points) This general fact is enough to prove the essential facts from lecture. Prove that if vertex a has a simple path to vertex b, and if b has a simple path to vertex c, then vertex a also has a simple path to c.

   Note: A simple path is a path with no repeated vertices.

   Hint: Just "attaching" the paths from a to b and the path from b to c together is close, but not quite enough. Think about what can go wrong. Hint: Do a direct proof. Draw a picture to help.

Solution:

Let's consider a simple path from A to B as AB, where a is represented by $t_0 \to t_1 \to ... \to t_k = B$. Additionally, there is a simple path from B to C as BC, where b is represented by $g_0 \to g_1 \to ... \to g_m = C$. b is present in both paths, showing an intersection. Let the first point of intersection be denoted as t_i and g_i.

Let's assume there exists another intersection point before t_i, g_i (E.g. if the vertices are connected in a loop). This new intersection point (hypothetical) should occur earlier in the paths since, after the intersection, only the continuation of one path is possible. However, this contradicts the assumption that t_i and g_i are the first intersection points.

Therefore, through the proof by contradiction, we see that there can't possibly be another intersection point. Consequently, we combine the two paths, forming $t_0 \to t_1 \to ... \to t_i \cup g_i + 1 \to ... \to g_m$. Starting from the intersection, we follow the simple path to point C, demonstrating that vertex A has a simple path to vertex C.

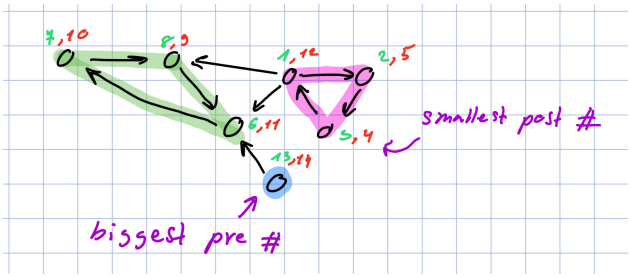


Figure 2: Counterexample for exercise 3, parts a) and b)

3. (20 points) Dr. Bhakta wants to find strongly connected components in graphs, and knows that the usual method is to start with the vertices with the highest post number in a DFS run on the reverse graph. This vertex is in a source SCC in the reverse graph, and therefore in a sink SCC in the original graph.

   However, Dr. Bhakta is annoyed about having to construct a reverse graph to run DFS in to find sinks. Instead, Dr. Bhakta is brainstorming possible shortcuts to try to find a sink in the original graph from running DFS in the original graph. Explain to Dr. Bhakta why his proposed shortcuts don't work.

   (a) (10 points) Show that the vertex with the smallest post number in a DFS run (first to return from the recursive call) may not be in a sink SCC.

   > **Solution:** Metagraph: pink, blue - sources, green - sink
   > As can be seen above, the smallest post number is located in a source (pink), however, the actual

   (b) (10 points) Show that the vertex with the largest pre number (last to start the recursive call) may not be in a sink SCC.

   > **Solution:** Metagraph: pink, blue - sources, green - sink
   > As can be seen above, the largest pre number is located in a source (blue), however, the actual sink is green.

4. (40 points) (Logic Solving) You have n very picky world leaders, and you need to choose some of them to join your new committee. For historical and political reasons, there are well known conventions that you have to follow when choosing a set of nations for this committee.

   For example, it might be that if Applevania joins the committee, then Bananaland and Cucumberstan would also demand to join. Also, if Bananaland joins, then Cucumberstan might refuse to join. If Great Durian is not chosen, then its ally Elderberria will demand to join, etc. Some countries will just demand to be included or excluded, for example, the Fig Islands demand to join the committee, and the United Grapes demand to not be included.

   You need to choose a (possibly empty) set of nations to include in the committee that meet all conventions. Because of the consequences of some of these choices, there may be some countries you have have to pick or exclude. For example, if the above was your input, you can't possibly include Applevania, otherwise then you'd have to include both Bananaland and Cucumberstan, but because Bananaland is there, you would have to exclude Cucumberstan. This is inconsistent, so there is no possible way to take Applevania.

   The question is: given a set of n nations and m requirements, can you find a set of countries to join the committee that will meet everyone's requirements?

   (a) (10 points) Come up with a few nations and some requirements where it is provably impossible for all the requirements to be met. Prove why it is impossible for all requirements to be met.

   Solution:

   $\overline{P} \implies Z$
   $Z \implies P$
   $P \implies E$
   $E \implies \overline{P}$

We want to design an algorithm that can take all this information and decide if it's even possible to find a suitable committee, and if possible, tell us who to choose. We will create an algorithm that can answer this question using the Strongly Connected Components algorithm.

First, we will represent all these requirements with a graph.

Start by creating a graph with $2n$ vertices, where each country i has one vertex that represents choosing that country, x_i and one vertex that represents not choosing that country, $\bar{x_i}$.

Each requirement is a boolean implication of the form $a \implies b$ between some two of these vertices. For example, if Applevania causes Bananaland to join, this would be expressed with the implication $a \implies b$. If Great Durian not joining makes Elderberria join, this would be represented as $\bar{d} \implies e$, etc.

Each implication $a \implies b$ appears in our graph as an edge (usually two). For each implication $i \implies j$, we add a corresponding directed edge $x_i \implies x_j$ to the graph, as well as the edge $\bar{x_j} \implies \bar{x_i}$. Remember that $\bar{\bar{x}} = x$.

(b) (5 points) What does a *path* in this graph *mean*? By this, I do NOT want you to explain what a path in any graph means, I want you to explain what a path in *this* specific graph means. In other words, I want you to answer the question: if there is a path from x_i to x_j (not necessarily an edge), what is the relationship between x_i and x_j in terms of the underlying boolean variables?

Solution: x_i causes x_j to join.

(c) (5 points) Similarly, what does a strongly connected component in this graph mean? What does it mean if there is a path from x_i to $\bar{x_j}$ and also from $\bar{x_j}$ back to x_i, in terms of the logic of the underlying boolean variables?

Solution: x_i causes x_j to NOT join and x_j by NOT joining causes x_i to join

(d) (10 points) What simple to detect property in this graph might a strongly connected component have so that you know immediately that the requirements are impossible to meet? Hint: Try to construct the graph for your impossible scenario in part a and see what happens with the variables in the strongly connected components.

Solution: If the same country wants to join or not join is existent in the same loop.

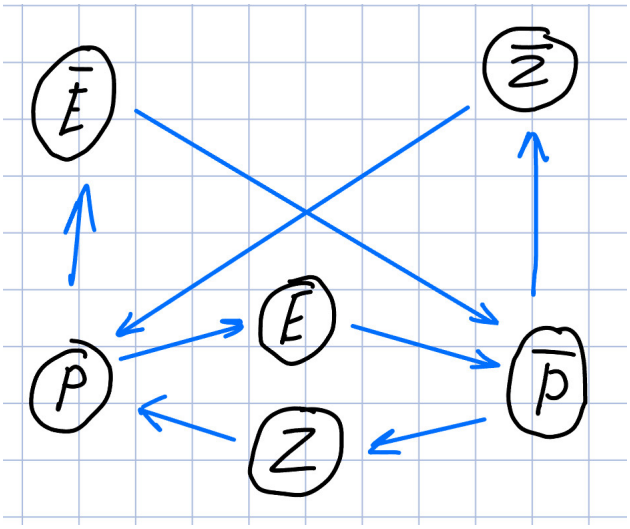


Figure 3: Graph for part d)

(e) (10 points) Argue that if the above property does NOT hold in your graph then will always be a valid assignment of True or False to all the variables so that everything is consistent. (Hint: Try an induction argument on the number of strongly connected components in the graph. Think about sources/sinks).

Solution: Best case will only have 2 SCCs.

Figure 4: My solution's result

5. (10 points) (Bonus) Kattis is a programming puzzle website, similar to popular websites like leetcode and topcoder, but with a focus on programming contest level problems.

   Solve this problem. https://open.kattis.com/problems/equivalences

   Feel free to use and modify any code for any *base* algorithms (like dfs, scc, etc.) that you find on the internet instead of writing these from scratch. Cite anything that you use. You may not use code that you lift completely from someone who is intentionally trying to solve this problem on kattis or similar websites.

   Explain your solution in words, and provide both your code and a screenshot that you passed all the tests.

   Solution: My solution was only able to solve the first test case.