CPSC 320 2023W1: Assignment 2

This assignment is due on **Wednesday October 11 at 10pm Vancouver time** on Gradescope. Assignments submitted before noon on Thursday will be accepted, but a penalty of 15% will be applied. Please follow the guidelines provided in Assignment 1. All the submission and formatting rules for Assignment 1 apply to this assignment as well.

1 List of names of group members (as listed on Canvas)

Provide the list here. This is worth 1 mark. Include student numbers as a secondary failsafe if you wish.

2 Statement on collaboration and use of resources

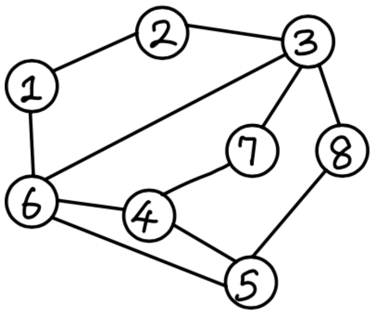
To develop good practices in doing homeworks, citing resources and acknowledging input from others, please complete the following. This question is worth 2 marks.

1. All group members have read and followed the guidelines for groupwork on assignments given on the website (see `https://www.students.cs.ubc.ca/~cs-320/2021S2/coursework.html`, under Assignments). ◯ Yes ◯ No
2. We used the following resources (list books, online sources, etc. that you consulted):
3. One or more of us consulted with course staff during office hours.
   ◯ Yes ◯ No
4. One or more of us collaborated with other CPSC 320 students; none of us took written notes during our consultations and we took at least a half-hour break afterwards.
   ◯ Yes ◯ No
   If yes, please list their name(s) here:
5. One or more of us collaborated with or consulted others outside of CPSC 320; none of us took written notes during our consultations and we took at least a half-hour break afterwards.
   ◯ Yes ◯ No
   If yes, please list their name(s) here:

3 Counting Shortest Paths

Let $G = (V, E)$ be an undirected, unweighted, connected graph with $n \geq 1$ nodes and m edges. For any pair of nodes x and v, let $c(x, v)$ be the total number of shortest paths from x to v.

Example: The following graph has one shortest path from 1 to 6. Also there are three shortest paths from 1 to 8. Two of these, namely path 1,2,3,8, and path 1,6,3,8, go through node 3, while one, namely path 1,6,5,8 goes through node 5.



1. (2 points) Draw a breadth first search tree rooted at 1 for the graph above. Include all dashed edges as well as tree edges.
2. (2 points) How many shortest paths are there from node 1 to node 7? That is, what is the value of $c(1, 7)$? Give a list of the paths.
3. (4 points) Here is an inductive definition for $c(x, v)$: If $x = v$ then there is exactly one shortest path from x to x, so $c(x, x)$ is 1. Otherwise,

$$c(x,v) = \sum_{\substack{u \mid (u,v) \in E, \text{ and} \\ d_x[u] = d_x[v]-1}} c(x,u).$$

   Intuitively, in the bfs tree rooted at x, we count the number of shortest paths to all nodes u at level $d_x[v] - 1$, such that there is an edge (either a tree edge or a dashed edge) from u to v.

   Provide code in the spaces indicated below, to obtain an algorithm that computes $c(x, v)$ for all pairs of nodes. For each node x, this code first initializes $c(x, v)$ for all v, and then calls a version of breadth first search that your selection will modify. Your algorithm should run in $\Theta(nm)$ time.

```
procedure COUNT-SHORTEST-PATHS($G$)
    ▷ $G$ is an undirected, connected graph with at least one node
    ▷ compute $c(x, v)$, the number of shortest paths, from $x$ to $v$, for all pairs of nodes $(x, v)$
    for all $x \in V$ do
        ▷ add code here to initialize $c(x, v)$ for all $v \in V$:



        call MODIFIED-BFS($x$)

procedure MODIFIED-BFS($s$)
    for each $v \in [1..n]$ do
        $d_s[v] \leftarrow \infty$
    $L_0 \leftarrow \{s\}$; $d_s[s] \leftarrow 0$; $d \leftarrow 1$
    while $L_{<d} \neq V$ do
```

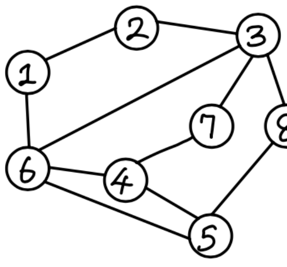
```
$L_d \leftarrow \emptyset$
**for** each $u \in L_{d-1}$ **do**
    **for** each $v$ adjacent to $u$ **do**
        **if** $d_s[v] == \infty$ **then**
            add $v$ to $L_d$; $d_s[v] \leftarrow d$;
            $p[v] \leftarrow u$
        $\triangleright$ **add your code here**:


$d \leftarrow d + 1$
```

4. (2 points) Explain briefly why the runtime of Algorithm Count-Shortest-Paths from part 3 is $O(nm)$.

4 Edge Load

Let $G = (V, E)$ be an undirected, unweighted, connected, graph, with n nodes and m edges. For two nodes x and y let the (x, y)-*edge load* of an edge e in the graph, which we'll denote by $\text{load}(x, y, e)$, be the number of shortest paths between x and y that pass through e. Let $\text{load}(e)$ be the sum of $\text{load}(x, y, e)$, taken over all pairs of nodes (x, y).

Example: Consider the following graph:



- If $e = (2, 3)$, then $\text{load}(1, 8, e) = 1$, since the only shortest path from 1 to 8 that goes via edge (2,3), is the path $1, 2, 3, 8$.
- If $e = (1, 6)$ then $\text{load}(1, 7, e) = 2$, with the two paths being $1, 6, 3, 7$ and $1, 6, 4, 7$.
- If $e = (2, 3)$ then $\text{load}(1, 4, e) = 0$, since no shortest path from 1 to 4 goes through edge $(2, 3)$.

1. (2 points) Answer the following two questions for the graph shown above.

   (a) Let $e = (6, 4)$. What is $\text{load}(1, 8, e)$? Choose one.

   ○ 0 ○ 1 ○ 2 ○ 3 ○ 4

   (b) Let $e = (1, 6)$. What is $\text{load}(1, 8, e)$?

   ○ 0 ○ 1 ○ 2 ○ 3 ○ 4

2. (3 points) We want to compute $\text{load}(e)$, for all edges e. For each e, our approach will be to first calculate $\text{load}(x, y, e)$ for each x and y, and then sum up these quantities to get $\text{load}(e)$. What lower bounds can we get on the runtime of this approach if we assume that it takes $\Omega(1)$ time to compute each quantity $\text{load}(x, y, e)$? Choose all that apply.

   ○ $\Omega(n + m)$ ○ $\Omega(n^2 m)$
   ○ $\Omega(n(n + m))$ ○ $\Omega((n + m)^2)$
   ○ $\Omega(n^2(n + m))$ ○ $\Omega((n + m)^3)$

3. (3 points) Let $d(x, y)$ denote the shortest path from x to y in graph G. Suppose that $e = (u, v)$ and furthermore that $d(x, u) < d(x, v)$, that is, u is closer to x than v is to x. Show that under these conditions,

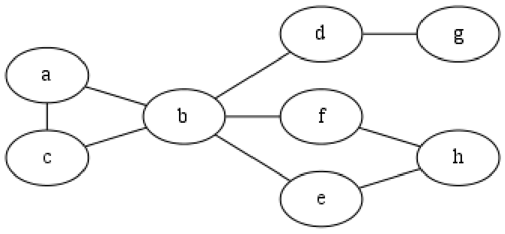
$$\text{load}(x, y, e) = \begin{cases} 0, & \text{if } d(x, u) + d(v, y) + 1 > d(x, y) \\ c(x, u) \times c(v, y), & \text{otherwise.} \end{cases}$$

4. (2 points) Explain how to compute all quantities $\text{load}(x, y, e)$, for $x, y \in V$ and $e \in E$, in $O(n^2 m)$ time.

5 Connector nodes

Let $G = (V, E)$ be a connected, unweighted, undirected graph with $n \geq 2$ nodes and m edges. We call a node v of G a *connector* if its removal (along with its incident edges) would make the remaining graph disconnected.

1. (2 points) Which, if any, nodes are connectors in the following graph? Select all that apply.



○ a ○ b ○ c ○ d ○ e ○ f ○ g ○ h

2. (2 points) Suppose that a depth first search tree T of G happens to be rooted at a node s, where s is *not* a connector. Explain why s can only have one child in T.

3. (2 points) Suppose that a depth first search tree T of G happens to be rooted at a node s, where the degree of s is tree T is 1. Might s be a connector of the graph? Justify your answer.

4. (3 points) Let v be an internal node of DFS tree T of G. Show that v is a connector if and only if for some child v' of v, there is no dashed edge from a node in the subtree rooted at v' to an ancestor of v. Here, an *ancestor* of v is any node w that is on the path from the root of the tree to v, other than v itself.

5. (4 points) For each node v of G other than s, let $l(v)$ be the level of node v in the DFS tree T. Let
$$l'_G(v) = \min\{l(v') \mid (v, v') \in E\}.$$
That is, $l'_G(v)$ is the minimum level $l(v')$ of a neighbour v' of v in G. Finally, let
$$l'_T(v) = \min\{l'_G(v') \mid v' \text{ is a descendant of } v \text{ in tree } T\}.$$
That is, $l'_T(v)$ is the minimum level of a neighbour of some node in the subtree of T rooted at v. (The *descendants* of a node $v \in V$ are all of the nodes, including v itself, in the subtree rooted at v.)

   Let v be an internal node of DFS tree T of G and let v' be a child of v in T. Show that $l'_T(v') \geq l(v)$ if and only if there is no dashed edge from a node in the subtree of T rooted at v' to an ancestor of v.

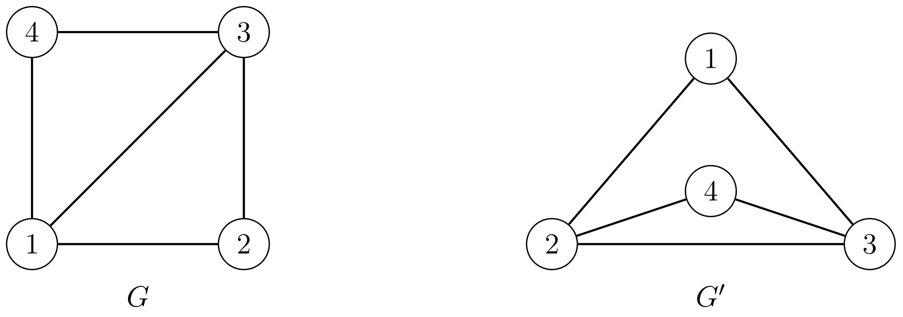
6. (4 points) Given the DFS tree T rooted at s, explain how to compute the quantities $l(v)$, $l'_G(v)$, and $l'_T(v)$ for all $v \in V$ in linear time.

7. (3 points) Describe an algorithm that finds all connector nodes of G in $O(n + m)$ time.

6 Graph isomorphism

1. (1 point)

   Let $G = (V, E)$ and $G' = (V, E')$ be two undirected graphs with the same set of nodes. G and G' are *isomorphic* if there is a permutation π of the nodes of G such that (i, j) is an edge of E if and only if $(\pi(i), \pi(j))$ is an edge of E'. For the following example, what permutation π in the list below shows that G and G' are isomorphic? Choose one.



   ○ $\pi[1..4] = [1, 4, 2, 3]$

   ○ $\pi[1..4] = [2, 4, 3, 1]$

   ○ $\pi[1..4] = [3, 2, 4, 1]$

   ○ $\pi[1..4] = [4, 1, 2, 3]$

2. (2 points) An instance of the Graph Isomorphism (GI) problem is a pair of undirected, connected graphs $G = (V, E)$ and $G' = (V, E')$. The output is "yes" if the graphs are isomorphic, and "no" if not.

   For many decades, researchers have sought efficient algorithms for Graph Isomorphism. One breakthrough, back in 1983, was Eugene Luks' algorithm with runtime $L(n) = \exp(\Theta(\sqrt{n \log n}))$, where n is the number of nodes of the graph. Here, $\exp(x)$ is e^x. That is, for some constants $c_L > 0$ and $c'_L > 0$, and sufficiently large n,

$$\exp(c'_L \sqrt{n \log n}) \leq L(n) \leq \exp(c_L \sqrt{n \log n}).$$

   More recently, Lazlo Babai has found an algorithm with runtime $B(n) = \exp((\log n)^{\Theta(1)})$. Explain this notation by writing down upper and lower bounds for Babai's runtime $B(n)$, similar to the bounds for Luk's algorithm above.

3. (4 points) Show that Babai's algorithm beats Luk's algorithm, by showing that $B(n) = o(L(n))$.