CPSC 320 2019S2: Assignment 2

Please submit this assignment via GradeScope at `https://gradescope.com`. Be sure to identify everyone in your group if you're making a group submission (which we encourage!).

Submit by the deadline **Tuesday July 16 at 11PM**. For credit, your group must make a **single** submission via one group member's account, marking all other group members in that submission **using GradeScope's interface**. Your group's submission must:

- Be on time.
- Consist of a single, clearly legible file uploadable to GradeScope with clearly indicated solutions to the problems. (PDFs produced via LaTeX, Word, Google Docs, or other editing software work well. Scanned documents will likely work well. **High-quality** photographs are OK if we agree they're legible.)
- Include prominent numbering that corresponds to the numbering used in this assignment handout. Put these **in order** starting each problem on a new page, ideally. If not, **very clearly** and prominently indicate which problem is answered where!
- Include at the start of the document the **ugrad.cs.ubc.ca e-mail addresses** of each member of your team. Please **do not** include names on the assignment. If you don't mind private information being stored outside of Canada and want an extra double-check on your identity, include your student number rather than your name.
- Include at the start of the document the statement: "All group members have read and followed the guidelines for academic conduct in CPSC 320. As part of those rules, when collaborating with anyone outside my group, (1) I and my collaborators took no record but names (and GradeScope information) away, and (2) after a suitable break, my group created the assignment I am submitting without help from anyone other than the course staff." (Go read those guidelines!)
- Include at the start of the document your outside-group collaborators' ugrad.cs.ubc.ca IDs, but **not** their names. (Be sure to get those IDs when you collaborate!)

1 O'd to a Pair of Runtimes

1. The pairs of functions below represent algorithm runtimes on an **undirected, connected** graph with n vertices and m edges. Assume that $n \geq 2$. For each pair, fill in the circle next to the best choice of:

   LEFT: the left function is big-O of the right, i.e., left $\in$ O(right)

   RIGHT: the right function is big-O of the left, i.e., right $\in$ O(left)

   SAME: the two functions are Θ of each other, i.e., left $\in \Theta$(right)

   INCOMPARABLE: none of the previous relationships holds for all allowed values of n and m.

   Do not choose **LEFT** or **RIGHT** if **SAME** is true. The first one is filled in for you.

This work is licensed under a Creative Commons Attribution 4.0 International License.
For license purposes, the author is the University of British Columbia.

Left Function	Right Function	Answer
n	n^2	**LEFT**
$n^2 + 3n$	m^2	◯ LEFT ◯ RIGHT ◯ SAME ◯ INCOMPARABLE
$186n$	$\sqrt{m}$	◯ LEFT ◯ RIGHT ◯ SAME ◯ INCOMPARABLE
$n \log n$	m	◯ LEFT ◯ RIGHT ◯ SAME ◯ INCOMPARABLE
$n \log n + m \log m$	$n \log(\sqrt{m}) + m \log(\sqrt{n})$	◯ LEFT ◯ RIGHT ◯ SAME ◯ INCOMPARABLE
$\frac{n}{\log m}$	1	◯ LEFT ◯ RIGHT ◯ SAME ◯ INCOMPARABLE

2. Exactly **one** of the problems above has the answer **INCOMPARABLE**. Identify that problem and briefly justify why neither function is big-O of the other.

3. Prove that $\log_a n \in \Theta(\log_b n)$ for all constants $a, b > 1$.

4. Let $h = \max\{f, g\}$, i.e., $h(x) = \max f(x), g(x)$ for all x. Prove that $h \in \Theta(f + g)$.

2 Dancing the Two-Step

Consider the SMP variant where, instead of a woman having an ordered preference list of all n men, she instead has a list of $n-1$ *pairwise* preferences between two men. For example, if a woman had the preference list $[m_1, m_2, m_3]$, this could be expressed as a *pairwise list:* $\{m_1 > m_2, m_2 > m_3\}$ (which we read to mean, "prefers m_1 over m_2, prefers m_2 over m_3").

Assume that every man appears in at least one of the pairwise preferences for each woman (so you will never have a man who is not ranked against any of the other men). Moreover, assume that all preferences are transitive: that is, if a woman prefers m_i to m_j and prefers m_j to m_k, it cannot also be the case that she prefers m_k to m_i.

This work is licensed under a Creative Commons Attribution 4.0 International License.
For license purposes, the author is the University of British Columbia.

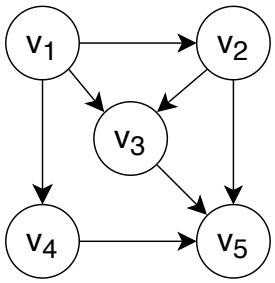
We say that an SMP preference list is **compatible** with a pairwise list if the pairwise list is consistent with the SMP preference list. For example, the pairwise list we saw above – $\{m_1 > m_2, m_2 > m_3\}$ – has a single compatible SMP preference list $[m_1, m_2, m_3]$. The SMP preference list $[m_1, m_3, m_2]$ is **not** compatible with the pairwise list because it's inconsistent with the pairwise preference $m_2 > m_3$.

1. Give and briefly explain an example of a pairwise list with $n = 3$ which has more than one compatible SMP preference list.

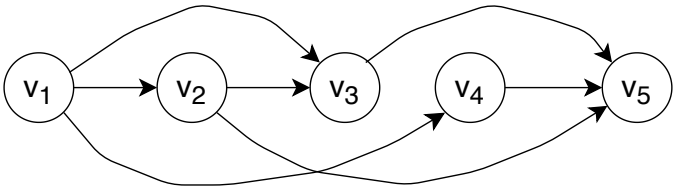
2. In terms of n, what is the largest possible number of compatible SMP preference lists for a given pairwise list? For what kind of pairwise list does this occur?

Partner Ordering

A **topological ordering** of a directed acyclic graph (DAG) $G = \{V, E\}$ is an ordering of the vertices such for every edge $(v_i, v_j) \in E$, v_i comes before v_j in the ordering. For example, the graph below has $\{v_1, v_2, v_3, v_4, v_5\}$ as a topological ordering.



Another way to think of topological orderings: if we place the vertices in their topological order, the edges all point left to right, as below.



For the following questions, assume you have a helper function called `TopoSort` which takes as input a connected, directed acyclic graph $G = \{V, E\}$ as input and returns a topological ordering of G in $O(|V|+|E|)$ time. You should also assume that `TopoSort` will return an error if G is disconnected or contains a cycle.

3. Give an algorithm that takes a pairwise list of men and generates a compatible SMP preference list. (When there is more than one SMP preference list, your algorithm should return a single one.) Your algorithm must call the `TopoSort` function.

4. Give and briefly justify a good asymptotic bound on the runtime of your algorithm.

5. **[BONUS, WORTH 1 COURSE BONUS POINT]** We assumed in the problem description that every man appears in at least one pairwise preference and that preferences were transitive. Explain why your algorithm would **not** work if either of these assumptions was violated.

This work is licensed under a Creative Commons Attribution 4.0 International License.
For license purposes, the author is the University of British Columbia.

3 Smooth Travels

Based on accessibility guidelines, an institution has classified all of its campus's pathways connecting points of interest into three groups: those that are at recommended specifications (R), at minimum specifications, and below minimum specifications (X). For each pathways that is rated M or X, you also have a cost to upgrade to the higher specification(s).

Note that a "pathway" may take various forms (e.g., a flight of steps), but that it will always connect exactly two points of interest. Furthermore, although two points of interest may be physically connected "in the real world" by multiple pathways (e.g., a flight of stairs and an elevator), the "logical" pathway will appear only once in the data rated according to the most accessible of the physical pathways connecting the two points.

A Staircase Too Many

Ideally, every point on the campus should be reachable from every other point using only pathways rated at R. However, the institution is nervous about the cost of upgrading the paths and wants to determine whether all areas are reachable with a "single point of assistance." That is, can every point on campus be reached from every other point using at most one pathway that is rated below the recommended specifications (i.e., at M or X)?

1. Give a small example of a graph that **does** pass this test even though it has four points of interest with the property that no one of the four points can be reached from any other of the points without using at least one pathway rated M or X. Clearly label each edge with a R, M, or X.

Consider the following algorithm, which assumes each vertex has extra fields `visited` (a Boolean that is initially `false`) and `steps` (a non-negative integer that is initially 0):

```
BFSWithSteps(G):
    if G has no vertices, return

    let Q be an empty queue (where each entry is a pair of a vertex and an integer)
    let v be an arbitrary vertex in G
    enqueue (v, 0) into Q
    while Q is not empty:
        dequeue the first pair (v, s) off of Q
        if v.visited is false:
            set v.visited to true
            set v.steps to s

            // Remember that (v,u) and (u,v) are the same edge
            for each edge (v,u) in the graph:
                if (v,u) has the label R:
                    enqueue (u, s) into Q
                else:
                    enqueue (u, s+1) into Q
```

This is like BFS except that it keeps track of how many assisted steps it took along the path to a particular point. The intent of the algorithm is that we would run it and report that all areas are reachable with a single point of assistance if every vertex has a `steps` value of 0 or 1.

2. Give an example of a graph (with each edge clearly labeled as R, M, or X) for which the algorithm labels every vertex with `steps` of 0 or 1, yet it is **not** the case that all areas are reachable using only a single point of assistance. **Label each vertex** with the `steps` value the algorithm assigns to it.

This work is licensed under a Creative Commons Attribution 4.0 International License.
For license purposes, the author is the University of British Columbia.

NOTES: you should choose (and clearly indicate!) which vertex the algorithm enqueues first. Your graph **must** be quite small for credit.

3. Give an example of a graph (with each edge clearly labeled as R, M, or X) for which the algorithm labels at least one vertex with `steps` of 2 or larger, yet all areas **are** reachable using only a single point of assistance. **Label each vertex** with the `steps` value the algorithm assigns to it.

   NOTES: you should choose (and clearly indicate!) which vertex the algorithm enqueues first. Your graph **must** be quite small for credit.

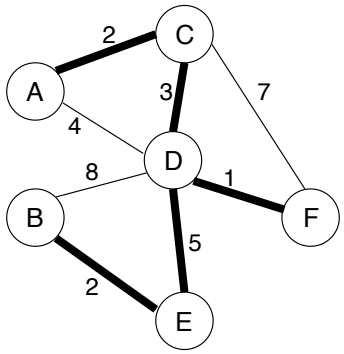
Reduction in Cost

The institution has decided to improve its standards, and now wants every point connected to every other point using only pathways at the recommended specifications.

4. Give pseudocode for an algorithm that determines whether every point is reachable from every other point using only vertices labeled R, and give (and justify) a good asymptotic bound for the runtime of your algorithm.

Based on your analysis, the institution has decided that it will need to upgrade its pathways, but it wants to do so for the lowest possible cost. The R Path Problem (RPP) is the problem of selecting the least expensive plan to upgrade pathways so that all points are connected to all other points using only pathways rated at R. Specifically, a solution to RPP is the set of M and X edges that should be upgraded.

Now, consider the **Minimum Spanning Tree (MST)** problem, we are given a weighted, undirected, connected graph and want to find the spanning tree – that is, a subset of edges that connects all the vertices together and contains no cycles – with the minimum possible total edge weight. For example, the graph below has as its minimum spanning tree the edges $(A, C), (C, D), (D, E), (E, B), (D, F)$, with total edge weight 13:



5. Give a reduction from RPP to MST and briefly explain why your reduction works.

4 All-Stars

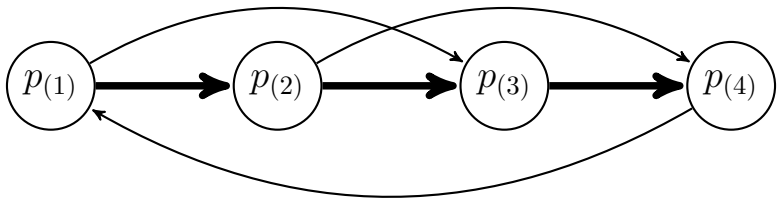
Suppose you're organizing a tennis tournament between n players $p_{(1)}, \ldots, p_{(n)}$. Each player competes against every other player, and there are no repeat matches and no tied scores. We model the results of the competition as a directed graph with n vertices and exactly one directed edge between each pair of vertices: $G = (V, E)$, with $V = p_{(1)}, \ldots, p_{(n)}$, where either $(p_{(i)}, p_{(j)}) \in E$ (if $p_{(i)}$ defeated $p_{(j)}$), or $(p_{(j)}, p_{(i)}) \in E$ (if $p_{(j)}$ defeated $p_{(i)}$). Such a graph is called a *tournament*.

You want to determine a way to rank these players, but a challenge here is that cycles **may** exist in your tournament – e.g., $p_{(i)}$ may defeat $p_{(j)}$ in a match, and $p_{(j)}$ could defeat $p_{(k)}$, but then we could still have $p_{(k)}$ win a match against $p_{(i)}$, which would lead to a cycle of length three. We define a *ranking* to be

This work is licensed under a Creative Commons Attribution 4.0 International License.
For license purposes, the author is the University of British Columbia.

a permutation of players $p_{(1)} \dots p_{(n)}$ so that edge $(p_{(i)}, p_{(i+1)}) \in E$ for all $1 \leq i < n$. In this problem we will show that a ranking always exists, *even if there are cycles in the original graph.*

Here's an example of a tournament of size $n = 4$ with a ranking in bold:



1. *Suppose* there exists a ranking for a tournament of size $n-1$. Draw a sketch[1] representing the ranking for this tournament and consider an n^{th} vertex. Under what condition can it be inserted into position 1? Under what condition can it be inserted somewhere in the middle of the existing chain? Under what condition can it be inserted into position n? Must one of those conditions exist?

2. Using your insights from the previous question, prove by induction that for any n, a tournament of size n always has a ranking.

3. Describe a brute force algorithm to find a ranking, given a tournament, and determine a good asymptotic bound on its runtime.

4. **[BONUS, WORTH 1 COURSE BONUS POINT]** Give pseudocode for a correct and efficient recursive algorithm to find a ranking, given a tournament.

[1] The sketch is to help you generate ideas, and does not need to be submitted with your assignment – though you certainly *can* submit it if it helps you explain your answers to the rest of this question.

This work is licensed under a Creative Commons Attribution 4.0 International License.
For license purposes, the author is the University of British Columbia.