CPSC 320 2024S: Assignment 5

This assignment is due **Friday, August 8 at 7 PM**. Late submissions will not be accepted. Please follow these guidelines:

- Prepare your solution using LaTeXand submit a pdf file. Easiest will be to submit using the .tex file provided. For questions where you need to select a circle, you can simply change `\fillinMCmath` to `\fillinMCmathsoln` .
- Enclose each paragraph of your solution with `\begin{soln}Your solution here...\end{soln}`. Your solution will then appear in dark blue, making it a lot easier for TAs to find what you wrote.
- Clearly label your answer to each question.
- Submit the assignment via GradeScope. 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**.
- After uploading to Gradescope, link each question with the page of your pdf containing your solution.

Before we begin, a few notes on pseudocode throughout CPSC 320: Your pseudocode should communicate your algorithm clearly, concisely, correctly, and without irrelevant detail. Reasonable use of plain English is fine in such pseudocode. You should envision your audience as a capable CPSC 320 student unfamiliar with the problem you are solving. You may **neither** include what we consider to be irrelevant coding details **nor** assume that we understand the particular language you chose. (So, for example, do not write `#include <iostream>` at the start of your pseudocode, and avoid language-specific notation like C/C++/Java's ternary (question-mark-colon) operator.)

Remember also to **justify/explain your answers**. We understand that gauging how much justification to provide can be tricky. Inevitably, judgment is applied by both student and grader as to how much is enough, or too much, and it's frustrating for all concerned when judgments don't align. Justifications/explanations need not be long or formal, but should be clear and specific (referring back to lines of pseudocode, for example). Proofs should be a bit more formal.

On the plus side, if you choose an incorrect answer when selecting an option but your reasoning shows partial understanding, you might get more marks than if no justification is provided. And the effort you expend in writing down the justification will hopefully help you gain deeper understanding and may well help you converge to the right selection :).

Ok, time to get started...

Group Members

Please list the CWLs of all group members here (even if you are submitting by yourself). We will deduct a mark if this is incorrect or missing.

1 Clustering 2.0

Recall the photo categorization problem described in Worksheets 5 and 6. An instance of the problem is given by

- n, the number of photos (numbered from 1 to n);
- E, a set of weighted edges, one for each pair of photos, where the weight is a similarity in the range between 0 and 1 (the higher the weight, the more similar the photos); and
- c, the desired the number of categories, where $1 \leq c \leq n$.

A solution to this problem is a *categorization*, which we define as a partition of photos into c (non-empty) sets, or categories. In the version of this problem we dealt with in class, our goal was to find the categorization that minimized the maximum inter-category edge weight. Suppose now that we change the goal of the photo categorization to **maximize the minimum intra-category edge similarity**. We call this the *Max-Min Clustering Problem* (MMCP).

1. [2 points] In class, we saw that an optimal greedy algorithm for minimizing the maximum inter-category edge weight was to initialize all photos in their own category and merge the photos adjacent to the highest-weight intercategory edge until we achieved the desired number of categories. Below is a greedy algorithm for MMCP, which starts with all photos in a single category and then separates two photos that are joined by a lowest-weight intracategory edge. This is repeated until we achieve the desired number of categories.

```
function MMCP-Greedy(n, E, c)
▷ n ≥ 1 is the number of photos
▷ E is a set of edges of the form (p, p', s), where s is the similarity of p and p'
▷ c is the number of categories, 1 ≤ c ≤ n
    create a list of the edges of E, in increasing order by similarity
    let C be the categorization with all photos in one category
    Num-C ← 1                ▷ Initial number of categories
    while Num-C < c do
        remove the lowest-similarity edge (p, p', s) from the list
        if p and p' are in the same category S of C then
            ▷ split S into two new categories T and T' as follows:
            put p in T
            put p' in T'
            for each remaining p'' in S do
                put p'' in T if p'' is more similar to p than p'
                put p'' in T' otherwise
            end for
            ▷ now S is replaced by T and T' in C
            Num-C ← Num-C + 1
        end if
    end while
    return C
end function
```

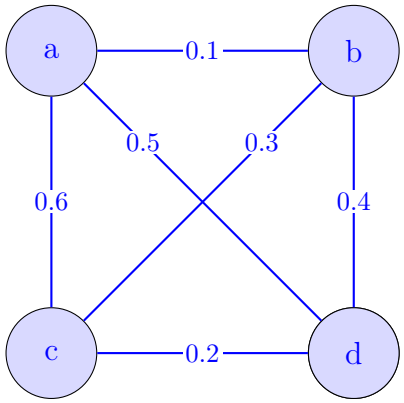
Give and explain a 4-node counterexample to show that this greedy algorithm is not optimal (i.e., does not always produce a solution that maximizes the minimum intra-category edge weight).

Let $C = C_1, C_2, \ldots, C_c$ be a partition of a nodes of a set V into c categories.

An edge, $(u, v) \in E$, is $intra - category$ if $\exists C_i \in C$ s.t. $u, v \in C_i$.

An edge, $(u, v) \in E$, is $inter - category$ if $\exists C_i, C_j \in C, i \neq j$ s.t. $u \in C_i$ and $v \in C_j$.

Consider the node structure, and we want to split this into two categories.



Following the algorithm we get the following.

$S = \{a, b, c, d\}$	one total category initially
$T = \{a\}, T' = \{b\}$	since (a, b) have lowest edge weight
$T = \{a, c, d\}, T' = \{b\}$	since $(c, a), (d, a)$ have heigher edge weight than $(c, b), (d, b)$
$C = \{T, T'\}$	final collection of categories

Observe the min-weight intra-category edge of this collection comes from $(c, d) \in T$ with weight 0.2.

But the better solution is $T = \{a, c\}, T' = \{b, d\}$ since the min-weight edge for each, respectively is 0.6 and 0.4.

We see that $0.4 > 0.2$, thus the greedy algorithm fails on this four node instance when consdiering two category partion.

2. [2 points] We've stated MMCP as an optimization problem. Give the decision version of MMCP, and prove that MMCP is in NP.

Defintition: We say a problem is in NP if there exists a verifier that can check in polynomial time.

Decision Version: Does there exist a partition of V into c categories such that all intra-category edges have similarity at least k, where $0 \leq k \leq 1$?

Proof that MMCP is in NP: To show a problem is in NP, by definition, we must show that a proposed solution can be verified in polynomial time.

A valid certificate is a partition of V into c categories $C = \{C_1, C_2, \ldots, C_c\}$. Note that $|C| \leq n$.

The verifier checks whether the similarity of all intra-category edges is at least k.

Let $|V| = n$. Since the graph is complete, $|E| = \frac{n(n-1)}{2} \in O(n^2)$.

We provide the following Verification Algorithm:

```
1: procedure VERIFY(E, C, k)
2:     for (u, v, s) ∈ E do
3:         for i = 1 to c do
```

```
            if u ∈ C_i and v ∈ C_i then
                if s < k then
                    return No
                end if
            end if
        end for
    end for
    return Yes
end procedure
```

Analyzing Running Time:

- Outer loop over all edges: $O(n^2)$
- For each edge, checking all $c \leq n$ categories for membership of u, v: $O(n)$

Total running time: $O(n^2 \cdot n) = O(n^3)$

Thus, the verifier runs in polynomial time, so MMCP is in NP.

3. [4 points] We will now show that MMCP is NP-hard (which, together with the proof from 3.2 that MMCP is in NP, means that it is NP-complete). We do this by reducing a known NP-hard problem to it. For this problem, you must do a reduction from the graph colouring problem, which (as a decision problem) is: given a graph G and a bound k, can we colour the vertices of G using no more than k colours such that no two adjacent vertices share a colour? You may assume that $k \leq n$, where n is the number of vertices in G.

   For this part of the question, you do not need to prove the correctness of your reduction (that comes next), but you should explain the reasoning behind your reduction and why it runs in polynomial time.

   Definition: A problem is NP-complete if it's NP and all NP-complete problems can be reduced to it.

   To satisfy this definition, it sufficies to reduce graph colouring to MMCP.

   Let $G = (V, E)$ be given. Let $k \in \mathbb{N}$ such that $k \leq n$. We constuct the following partition and edge set.

   For each pair $u, v \in V$, if $(u, v) \in E$ add $(u, v, 0)$ to E', otherwise add $(u, v, 1)$ to E'.

   Then we say that each $v \in V$ is a photo with the weighted edge set E'.

   We then ask the MMCP solver if there exists a parition of V into k categories so that each edge intra-category edge is at least 1.

4. [3 points] To prove your reduction correct, you must prove that the answer to the MMCP instance is YES if and only if the answer the answer to the graph colouring instance is YES. For this part of the question, prove that if the answer to the graph colouring instance is YES, then the answer to your reduced MMCP instance is YES.

   Claim: Given the reduction, Graph Colouring instance is YES $\iff$ MMCP instance is YES.

   Proof $\implies$: Let an instance of Graph colouring be given, with $G = (V, E)$. Reduce E to E' as before.

   Suppose G can be coloured with no more than k colours, such that no adjacent vertices share a colour.

   We show there exists partition of V to k categories so that the intra-category edges have weight 1.

   Place each node in a category of it's respective colour, C_j. If G is $i \leq k$ colourable, then it's k colourable.

   WLOG, we will assume that G is exactly k colourable. Thus, we have a partition of k categories.

   Observe that by definition, if $u, v \in C_j$ then they share a colour, which implies they are not adjacent.

   Which means that the edge weight for (u, v) in E' is 1 by construction. Hence, we get that the intra-category edges have weight at least 1.

   This completes the proof.

5. [3 points] Now for the other direction of the if-and-only-if: prove that if the answer to the reduced MMCP instance is YES, then the answer to the original graph colouring instance is also YES.

   Proof $\Longleftarrow$: Let the reduction from graph colouring to MMCP be given.

   We assume that MMCP returned to use the answer YES.

   We have a node set V, edge set E', partition into k categories with intra-category edge weight 1.

   For each category, assign a distinct colour, and colour all nodes in that category with the same colour.

   The claim is this is a valid colouring of $G = (V, E)$. It suffices to show no adjacent nodes share a colour.

   Consider any edge $(u, v) \in E$. By construction, this edge appears in E' with weight 0.

   Since the intra-category edges in the MMCP partition all have weight at least 1, it must be that u and v belong to different categories, otherwise it would have returned NO as an answer.

   Thus, u and v received distinct colours, which means that adjacent vertices do not share a colour.

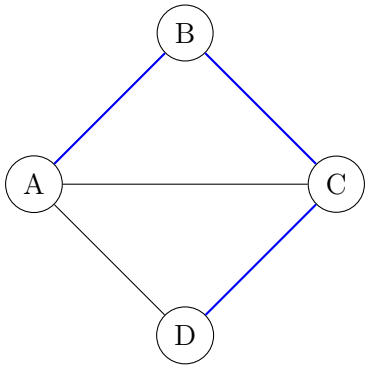
   Therefore, the constructed colouring is valid and uses at most k colours.

   This completes the proof.

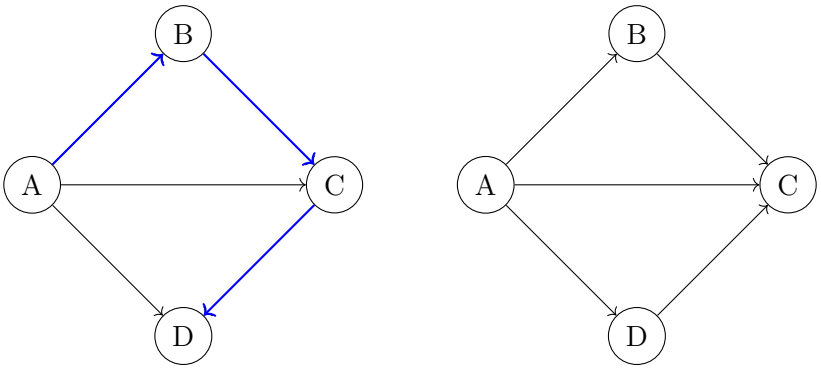
2 The Path To Victory 2.0

A Hamiltonian Path is a path in a graph that visits each vertex exactly once. In this question, we consider the variant of the Hamiltonian Path problem with the start and end node specified: that is, given a graph $G = (V, E)$ and two nodes s and t, does there exist a path from s to t that visits every node in G exactly once? You can may assume that G contains at least three vertices.

The Hamiltonian Path problem can be applied to graphs that are either undirected or directed. For example, the undirected graph below has a Hamiltonian Path from A to D given by (A, B, C, D), shown in blue:



For the directed graphs below, the graph on the left has a Hamiltonian Path from A to D, while the graph on the right does not have a Hamiltonian Path from A to D.



We refer to the undirected version of this problem as **UHP** and the directed version as **DHP**.

1. [3 points] Give a reduction from UHP to DHP.

   Let $G = (V, E)$ be an undirected graph. We construcut a graph $G' = (V, E')$ by the following.

   For each $\{u, v\} \in E$ add $(u, v), (v, u)$ to E'.

   Then for $s, t \in V$ we ask DHP if there exists a Hamiltonian path from s to t in G'.

2. [4 points] Prove the correctness of your reduction from UHP to DHP. That is, prove that the answer to your reduced DHP instance is YES if and only if the answer to UHP is YES.

   We'll say that $V = \{s, v_1, v_2, \ldots, v_{n-2}, t\}$. And we ask for Hamiltonian path from s to t.

   ($\implies$): Suppose that the answer to the reduced DHP instance is YES.

   Then there is a Hamiltonian path from $s - t$ in $G' = (V, E')$.

   Then there is a simple path on V, $(s, v_1, v_2, \ldots, v_{n-2}, t)$ so that $(s, v_1) \in E', (v_{n-2}, t) \in E'$ and $(v_i, v_{i+1}) \in E'$ for $i = 1, 2, \ldots n - 3$.

   By construction, we have that $\{s, v_1\}, \{v_{n-2}, t\} \in E$ and $\{v_i, v_{i+1}\} \in E$ for $i = 1, 2, \ldots, n - 3$.

   By definition, $(s, v_1, v_2, \ldots, v_{n-2}, t)$ is also a simple path in $G = (V, E)$.

   But this contains all nodes in V, thus it is a Hamiltonian path.

   ($\impliedby$): Suppose that the answer to UHP is YES for the graph $G = (V, E)$.

   This means that there is a Hamiltonian path, $(s, v_1, v_2, \ldots, v_{n-2}, t)$ so that $\{s, v_1\}, \{v_{n-2}, t\} \in E$ and $\{v_i, v_{i+1}\} \in E$ for $i = 1, 2, \ldots, n - 3$.

   We aim to show there exists a Hamiltonian path in the reduction instance.

   For $G' = (V, E')$, $(s, v_1) \in E', (v_{n-2}, t) \in E'$ and $(v_i, v_{i+1}) \in E'$ for each $i = 1, 2, \ldots n-3$ by assumption.

   Then the same path, $(s, v_1, v_2, \ldots, v_{n-2}, t)$ is by definition Hamiltonian, since each node in V appears in it exactly once and the associated directed edges exist.

   This completes the proof.

3. [7 points] Give a reduction from DHP to UHP. You **do not** need to prove the correctness of this reduction, but you should clearly explain the key components of your reduction and why they are there.

   Let $G = (V, E)$ be a directed graph. With $V = \{s, v_1, v_2, \ldots v_{n-2}, t\}$.

   Let V_o, V_i be vertex sets such that $|V| = |V_o| = |V_i|$.

   Let $f_o : V \to V_o$ and $f_i : V \to V_i$ be bijective functions.

   Here, we make two copies of the vertex set and will encode the directed edge using copies of the nodes.

   We then construct a new edge set E' based on the original directed edge set E.

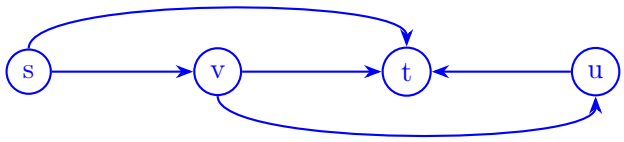
   Then for each edge $(v, u) \in E$ we add $\{f_o(v), f_i(u)\}$ to E'. We also add for each $v \in V, \{f_o(v), f_i(v)\}$ to E'.

   Then our new graph is $G' = (V_o \cup V_i, E')$.

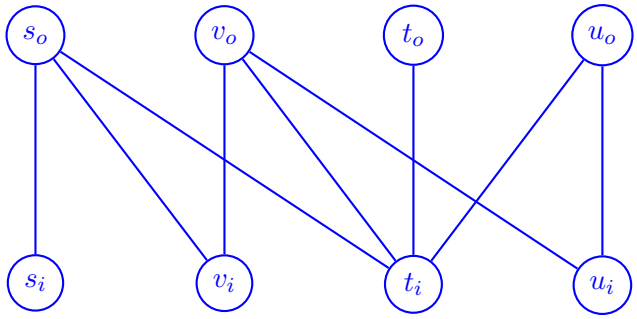
   In other words, V_o corresponds to the original vertices acting as outgoing nodes, and V_i corresponds to the original vertices acting as incoming nodes.

   As an illustration,

   Original Directed Graph



Reduced Undirected Graph using V_o and V_i



We then ask UHP does there exists a hamiltonian path from $f_i(s)$ to $f_o(t)$ in G'?

If yes, our original graph has a hamiltonian path, if not, then it does not.

Proof of Correctness: Say that $|V| = n$, where $n \geq 3$.

Assume DHP returns yes when asked if $G = (V, E)$ if it contains hamiltonian path from s to t in V.

We show there exists a hamiltonian path in $G' = (V_i \cup V_o, E')$ so that UHP returns yes on $f_i(s), f_o(t)$.

By construction, each of $\{f_i(v), f_o(v)\} \in E'$. Let $P = (s, v_1, v_2, \ldots, v_{n-2}, t)$ be the hamiltonian path.

By definition, $(s, v_1), (v_{n-2}, t) \in E$ and $(v_i, v_{i+1}) \in E$ for $i = 1, 2, \ldots, n-3$.

Then consider the permutation of nodes on $V_o \cup V_i$, by the following:

$P' = (f_i(s), f_o(s), f_i(v_1), f_o(v_1), f_i(v_2), f_o(v_2), \ldots f_i(v_{n-2}), f_o(v_{n-2}), f_i(t), f_o(t))$.

By construction, $\{f_o(s), f_i(v_1)\}, \{f_o(v_{n-2}), f_i(t))\} \in E'$. As those directed edges exist in that order.

Similarly, $\{f_o v_i, f_i(v_{i+2})\} \in E'$ for $i = 1, 2, \ldots, n-3$. But by definition, P' is a hamiltonian path in G'.

Thus, UHP will return yes when asked if G' contains a hamiltonian path from $f_i(s)$ to $f_o(t)$.

This proves one direction, now assume that UHP returns yes upon begin given $G' = (V_i \cup V_o, E')$.

We will show that this means there is a hamiltonian path in G from s to t.

Observe that for any $(u, v) \in E$ that $\{f_o(u), f_o(v)\}, \{f_i(u), f_i(v)\} \notin E'$ by construction.

In other words, the outcoming nodes do not connect to each other and the the incoming nodes do not connect to each other in G'.

Thus, for any path in G' of the form, $\rho = (v_1, v_2, \ldots, v_k)$ if $v_j \in V_o$ then $v_{j+1} \in V_i$ for $j = 1, 2, \ldots, k-1$.

Let $P = (f_i(s), o_1, i_1, o_2, i_2, \ldots, o_{n-1}, i_{n-1}, f_o(t))$ be the hamiltonian path from $f_o(s)$ to $f_i(t)$ in G' such that $i_j \in V_i, o_j \in V_j$.

Now construct, $P' =$

3 Very Special Problems

In this question, we consider special cases of NP-complete problems. Formally, we say that Problem B is a **special case** of Problem A if every instance of Problem B can be viewed as an instance of Problem A. We've seen examples of this in class: for example, 3-SAT is a special case of SAT (where every clause has length 3). Minimum spanning tree is a special case of Steiner Tree, in which the set of vertices we need to connect includes all the vertices in V.

1. [4 points] Consider the **Bounded-Leaf Spanning Tree Problem (BLST)**: given a graph $G = (V, E)$ and an integer k, does G have a spanning tree with no more than k leaves?

   Give an NP-complete problem that is a special case of BLST, and justify why this problem is a special case.

   Let $G = (V, E)$ be a graph. Suppose that this graph had a hamiltonian path.

   This means we can form a spanning tree out of this graph using exactly $n - 1$ edges.

   Let $P = (v_1, v_2, \ldots, v_n)$ be the hamiltonian path in G, and consider it the spanning tree structure.

   By definition, a leaf in a tree is a node with degree 1. We see that v_1, v_n both have degree 1.

   Hence, this tree has exactly two leaves.

   Thus, if we take G and ask the question to HAMPATH, does this graph contain a hamiltonian path?

   If the reply is yes, then G has a spanning tree with 2 leaves, or we can also say no more than 2 leaves.

   In other words, HAMPATH is a special case of BLST where we specify $k = 2$.

2. [3 points] You showed in the previous question that there is an NP-complete problem that is a special case of BLST, and it is not difficult to show that BLST is in NP (though we are not asking you to do this). Does this imply that BLST is NP-complete? Justify your answer.

   This does not necessarily imply that BLST is NP-complete. Let Y be some NP-complete problem.

   By definition, to prove BLST is NP-complete, it remains to show that $Y \leq_p BLST$.

   While we showed in the previous part that an NP-complete problem (HAMPATH) is a special case of BLST (for $k = 2$), this is not sufficient to conclude NP-completeness of BLST in general.

   In particular, it remains to show that $BLST(k = 2) \leq_p BLST$ or equivalently $HAMPATH \leq_p BLST$.

   That is, we must show a polynomial time reduction from HAMPATH (or any another problem Y) to BLST, not just observe that one is a special case of the other.

   Hence, unless such a reduction is explicitly given with proof of correctness, we cannot conclude that BLST is NP-complete.

3. [4 points] Give an example of a polytime-solvable problem you have seen in this class that is a special case of an NP-complete problem. Justify your answer.

   Recall the problem of finding a topological ordering on a directed graph $G = (V, E)$.

   The dynamic programming solution can produce a solution in polynomial time.

   Now recall the