National University of Singapore

School of Computing

CS4234 - Optimisation Algorithms

(Semester 1: AY2021/22)

Friday, 26 November 2021, PM (2 hours)

INSTRUCTIONS TO CANDIDATES:

1. Do **NOT** open this final assessment paper until you are told to do so.
2. This assessment paper contains FOUR (4) sections.
   It comprises TWELVE (12) printed pages, including this page.
3. This is an **Open Book + Open Laptop Assessment**.
   You are free to use your Laptop as you see fit (to read notes, use Excel, write/run code, etc).
   But you are NOT allowed to use the Internet (web browser, messaging/cloud services, etc).
4. There are 6 pages of answer sheets.
   Answer **ALL** questions within the **given (boxed) space** to make grading easier.
   When you write your answers, you can do so using either pen or pencil, just write **legibly**!
   You can use the last blank page to write a bit more.
   At the end of the paper, you just need to hand in your answer sheets.
5. Important tips: Pace yourself! Do **not** spend too much time on one (hard) question.
   Read all the questions first! Some (subtask) questions might be easier than they appear.
6. You can use **pseudo-code** in your answer but beware of penalty marks for **ambiguous answer**.
   You can use **standard, non-modified** classic algorithm in your answer by just mentioning its name, e.g., run Dijkstra's on graph G, Kuhn-Munkres on graph G', etc.
7. All the best :)

A The Easier Parts (45 marks)

A.1 Create Min-Vertex-Cover test case s.t. $|\text{MVC}| > |M|$ (5 marks)

In Lecture 02, we learn about the deterministic Approx-Vertex-Cover-2 greedy approximation algorithm that takes any edge (u, v) in the graph, put **both** endpoints u and v into the vertex cover, and then remove u, v, plus all edges adjacent to u or v from the graph. We are told that the action of taking an edge and then exclusively pair its endpoints, removing that edge and any other edges adjacent to it, is called 'matching' (before we learn the proper topic of Graph Matching in Lecture 06). This matching is not necessarily the one with maximum cardinality, but we can easily prove that $|\text{MVC}| \geq |M|$ (notice the $\geq$ sign).

Notice that many (almost all) sample small graphs shown in Lecture 01+02 involving MVC, we have $|\text{MVC}| = |M|$. So here is the challenge: Draw any simple and small undirected unweighted graph so that $|\text{MVC}| > |M|$ to be used to improve the explanation slides in the future. Give a short explanation of your drawing by highlighting the edges that are taken in your arbitrary matching M and the vertices in the optimal MVC.

A.2 Greedy Nearest Neighbor for TSP (5 marks)

There is yet another greedy algorithm for TSP (the standard Metric (2D Euclidean) No-Repeat version), called the Greedy Nearest Neighbor (GNN) algorithm. Its pseudo-code is as follows:

```
GreedyNearestNeighborTour
   tour[0] = 0 // assume the n vertices are labeled with 0-based indexing
   used[0] = true
   for i = 1 to n-1 // repeat n-1 times
      best = -1
      for j = 0 to n-1 // dist = Euclidean distance between two points on 2D plane
         if not used[j] and (best = -1 or dist(tour[i-1], j) < dist(tour[i-1], best))
            best = j
      tour[i] = best
      used[best] = true
   return tour
```

Analyze the pseudo-code above and answer the following sub-questions:
What is its time complexity in terms of n? (1 mark)
What is the general behavior of this algorithm? (2 marks)
Is it possible that GNN produces a TSP tour with length longer than $2 \cdot OPT$? (2 marks)

A.3 Finding a Simple Cycle in an undirected unweighted Graph (5 marks)

Show how to use an algorithm that we have learned in class to find any simple cycle involving two special vertices u and v in an undirected unweighted graph $G = (V, E)$. A simple cycle that we want is a cycle that does not repeat any vertex along the cycle, and that cycle passes through the two vertices of interest: u and v. Output "impossible" if there is no such simple cycle or print any valid cycle.

A.4 Kuhn-Munkres Algorithm Tracing Question (10 marks)

Given a small weighted bipartite graph as shown in Figure 1 (the edge weights were not clear on black-and-white print; please zoom-in the PDF to avoid ambiguity), run Kuhn-Munkres (KM) algorithm on it to find perfect matching of size 4 matchings (*involving* 4 *real edges*) with the highest total weight. Please explain the steps taken by the algorithm. It is important that the edges that were not present in Figure 1 should not be chosen by the KM algorithm.

You can assume that KM algorithm always prefer lower vertex number in the event of ties (e.g., the first free vertex in Figure 1 is vertex a (not any other vertices) and the first edge explored by the algorithm is edge (a, e) (not any other edges), and so on.

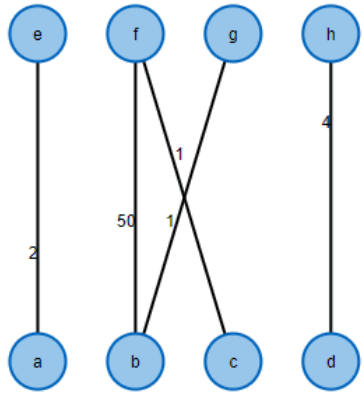

Figure 1: A small weighted MCBM instance

A.5 Edmonds' Matching Algorithm Tracing Question (10 marks)

Given a small unweighted general graph as shown in Figure 2 that already has a *maximal* matching, i.e. edge 0-2 and 1-4 (found by running the randomized greedy preprocessing algorithm), run Edmonds' Matching (Blossom) algorithm on it to find one more matching. Please explain the steps taken by the algorithm. It is important that blossom contraction and subsequent lifting/expansion are shown.

You can assume that Edmonds' Matching algorithm always prefer lower vertex number in the event of ties (e.g., the first free vertex in Figure 2 is vertex 3 (not vertex 5), the first edge explored by the algorithm is edge $(3, 1)$ (not edge $(3, 4)$), the first blossom found is 3-1-4-3, and so on.

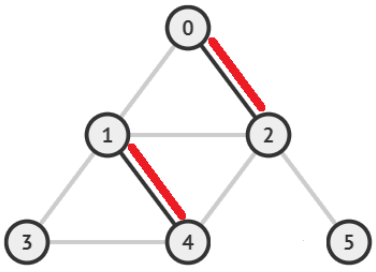

Figure 2: A small unweighted MCM instance

A.6 SLS Statements (10 marks)

For each SLS-related statements below (2 marks each), decide if they are "More towards True", "It depends", or "More towards False" and provide a *short* justification.

1. It is impossible to find an approximation algorithm for TSP that is better than the well-known Christofides's 1.5-approximation algorithm.
2. All instances of M-NR-TSP (the ones in /tsp Mini Project) have similar characteristics.
3. We can simply reuse our best performing SLS algorithm for the /tsp and/or /mwvc Mini Project to solve the same two problems (/tsp and /mwvc) if runtime is not a major constraint (i.e., we are given much more than 2s runtime per test case).
4. In any /mwvc instances (undirected vertex-weighted graphs), all vertices are equally good to be removed/added from/into a vertex cover.
5. It is easy to create an SLS algorithm to attack a new NP-hard problem, given the multitude of 'nature-inspired' SLS algorithms recently.

B $\frac{3}{2}$-Approximation for Maximum Matching (15 marks)

B.1 Prove (5 marks)

Suppose we have a *general* graph $G = (V, E)$ and a *maximal* matching M such that every augmenting path of length 3 forms a triangle. Prove that, for any other matching M', we have $|M'| \leq \frac{3}{2} \cdot |M|$.

B.2 Apply (5 marks)

Using the result from the previous Subsection B.1, construct an $O(|V| + |E|)$ algorithm which gives a $\frac{3}{2}$-approximation for maximum matching.

B.3 Create Test Case (5 marks)

Create a small graph G with $|V| = 7$ vertices (and any $|E|$ edges of your choosing), show its *maximum* matching $M*$ of G, and also show a *maximal* matching M that for every augmenting path of length 3 forms a triangle so that $|M*| = \frac{3}{2} \cdot |M|$.

C Max-Swap-Free (20 marks)

You are given W, a set of N words that are anagrams of each other. There are no duplicate letters in any word. A set of words $S \subseteq W$ is called **swap-free** if there is no way to turn a word $x \in S$ into another word $y \in S$ by swapping only a single pair of (not necessarily adjacent) letters in x. Find the size of the largest swap-free set (called the MAX-SWAP-FREE set) chosen from the given set W.

For example, if $N = 2$ and $W = \{$"sh", "hs"$\}$, then MAX-SWAP-FREE $= \{$"sh"$\}$ (or $\{$"hs"$\}$) with size 1 (we cannot take both words in W as they are not swap-free).

C.1 Sample Test Cases (1+2+2 = 5 marks)

You are given three small sample test cases below, please find their MAX-SWAP-FREE sets!

1. $N = 3$ and $W = \{$"she", "esh", "hes"$\}$ (1 mark).
2. $N = 6$ and $W = \{$"xyz", "xzy", "yxz", "yzx", "zxy", "zyx"$\}$ (2 marks).
3. $N = 5$ and $W = \{$"stevan", "sevant", "stevna", "nevats", "evants"$\}$ (2 marks).

C.2 Solve Optimally and Universally (15 marks)

Now solve this problem with N words in W, each word have the same length – due to them being anagrams of each other (the length is not more than 26 characters). Describe the fastest possible algorithm (you can use pseudo-code) and analyze its time complexity.

If you feel this problem is NP-hard or your solution's time complexity is too high, solve for $1 \leq N \leq 22$. Otherwise, if you feel this problem has a suitable polynomial solution, solve for $1 \leq N \leq 500$. One of the route is the correct one (it performs not more than 10^8 operations) and has up to 15 marks, the other route is capped at partial 7 marks.

D Pair-Programming (20 marks)

Steven is teaching a class of N students (let's just number them from 1 to N). He likes to *approximately* halves his grading workload by creating $\frac{N}{2}$ project groups of twos. As we have learned from other past papers of this module, `Project-Grouping` for group size of 3 or more is just an alternative name of an NP-hard optimization problem `Min-Clique-Cover`.

As usual, arranging N students into $\frac{N}{2}$ groups comes with the usual headache: only friends are willing to work together. Luckily the students in Steven's class are a very friendly bunch. In particular, if p, q, and r are any three distinct students in Steven's class, p and q are friends, q and r are friends, then p and r are interestingly also friends.

Working in a group of two (pre-COVID) entails one traveling to the other's house for project discussion, and that costs traveling time (these days, people can just use an e-Conferencing tool to do the same). Steven have asked all N students in his class to calculate, for each of their friends, how much traveling time is needed to go from one student's house to the other (this value is the same at either direction). There are M ($0 \leq M \leq 231$) friendship information given as an edge list of M tuples (p, q, and c) where p and q are 1-based student number ($p \neq q$) and c is the traveling time ($0 \leq c \leq 10^6$).

Now, can you help Steven figure out what is the minimum total traveling time if he arranges the groups optimally, or determine that it is not possible to arrange all N students into $\frac{N}{2}$ groups of twos (just print "impossible").

Example 1: If $N = 4$ and $M = 6$ friendship information $= \{(1, 2, 10), (1, 3, 10), (1, 4, 10), (2, 3, 20), (2, 4, 30), (3, 4, 10)\}$, then PAIR-PROGRAMMING $= 20$ (pair student $(1, 2)$ and $(3, 4)$ to produce minimum total traveling time of 20; the other two pairings cost 30 and 40).

Example 2: If $N = 4$ and $M = 3$ friendship information $= \{(1, 2, 10), (1, 3, 10), (2, 3, 20)\}$, then PAIR-PROGRAMMING = "impossible" (one reason: student 4 has no friend in class).

D.1 Sample Test Cases (1+2+2 = 5 marks)

You are given three small sample test cases below, please find the solution for PAIR-PROGRAMMING!

1. $N = 2$ and $M = 1$ friendship information $= \{(1, 2, 7)\}$ (1 mark).
2. $N = 5$ and $M = 2$ friendship information $= \{(1, 2, 10^6), (3, 4, 10^6)\}$ (2 marks).
3. $N = 6$ and $M = 7$ friendship information $= \{(1, 2, 10), (1, 3, 12), (1, 4, 11), (2, 3, 8), (2, 4, 6), (3, 4, 10), (5, 6, 59)\}$ (2 marks).

D.2 Solve Optimally and Universally (15 marks)

If you feel this problem is NP-hard or your solution's time complexity is too high, solve for $1 \leq N \leq 22$ (hint: $22 \times (22 - 1)/2 = 231$). Otherwise, if you feel this problem has a suitable polynomial or fast enough solution, solve for $1 \leq N \leq 500$ (the constraint $0 \leq M \leq 231$ still applies). One of the route is the correct one (it performs not more than 10^8 operations) and has up to 15 marks, the other route is capped at partial 7 marks.

– End of this Paper, All the Best –

E Answer Sheets

Write your Student Number in the box below:

A	0							

This portion is for examiner's use only

Section	Maximum Marks	Your Marks	Remarks
A	45		
B	15		
C	20		
D	20		
Total	100		

Box A-1. If you understand what is required, there is a very small graph as the answer.

Box A-2. Answer in three short lines.

Box A-3. There is an easy REDACTED solution.

Box A-4. The tracing should be doable this time (if overflow, use the last page).

Box A-5. The tracing should be doable this time (if overflow, use the last page).

Box A-6. Preferably just in 5 short lines.

Section A Marks = __ + __ + __ + __ + __ + __ = ___

Box B-1

Box B-2

Box B-3

Section B Marks = __ + __ + __ = ___

Box C-1 (write the answers for part 1/2/3 in not more than 3 lines, in sequential order).

Box C-2

Section C Marks = __ + __ = ___

Box D-1 (write the answers for part 1/2/3 in not more than 3 lines, in sequential order).

Box D-2

Section D Marks = __ + __ = ___

– You can use this blank page to write a bit more –

– Be careful that usually the size of the boxes reflect the model answer lengths –