National University of Singapore

School of Computing

CS4234 - Optimisation Algorithms

(Semester 1: AY2019/20)

Date and Time: Wednesday, 09 October 2019, 12.10-13.40 (90m)

INSTRUCTIONS TO CANDIDATES:

1. Do **NOT** open this midterm test assessment paper until you are told to do so.
2. This assessment paper contains THREE (3) sections.
   It comprises TEN (10) printed pages, including this page.
3. This is an **Open Book Assessment**.
4. Answer **ALL** questions within the **boxed space** in this booklet.
   Only if you need more space, then you can use the empty last page 10.
   You can use either pen or pencil. Just make sure that you write **legibly**!
5. Important tips: Pace yourself! Do **not** spend too much time on one (hard) question.
   Read all the questions first! Some 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, run Kruskal's on graph G', etc.
7. Write your Student Number in the box below:

My Student Number:

A	0							

This portion is for examiner's use only

Section	Maximum Marks	Your Marks	Remarks
A	30		
B	40		
C	30		
Total	100		

A The Easier Ones (30 marks)

Q1. A Real Life Problem (2+1+4+1 = 8 marks)

Earlier this semester, Steven has to decide which of the $m = 4$ CS4234 tutorial groups this semester (initial setup for $4 \times 15 = 60$ students) has to be closed as the class size as of Week 01 was just $n = 39$ students and thus the expected tutorial group size of $\frac{n}{m} < 10$ – too costly to run (from School's economical point of view). Eventually, Steven closed CS4234 Monday tutorial group 02. Steven wants to generalize this problem with the following constraints and assumptions:

1. Each of the n students must be inside a tutorial group,
2. It is ok if the size of some tutorial group(s) < 10 in order to satisfy the first constraint,
3. Each student is honest and declares as many of the m tutorial groups that they can attend,
4. $4 \leq m \leq 20$ and $20 \leq n \leq 400$ (constraints assumption for Steven's possible largest module),
5. Steven wants to minimize the number of tutorial groups that he has to setup.

What is the name of the optimization problem that Steven faced?

Is it an NP-hard problem or is there a polynomial-time solution for it?

What will you suggest to Steven so that he can solve this early-semester routine problem?

Is the optimal answer unique?

Q2. `Steiner-Tree` on Tree (1+1+6 = 8 marks)

In PS2, we had (General-)`Steiner-Tree` problem where the input is a general (weighted) graph $G = (V, E)$ with a set of required terminal vertices (the remaining vertices can be candidates of Steiner vertices). But we can also give a Tree as the input graph (a Tree is also a valid graph) along with the usual set of required terminal vertices.

Is this special case remains an NP-hard problem (also see Section B) or is there a polynomial-time solution for it?

Is the answer for this special case always a connected subtree of the input Tree?

Please elaborate the solution and state its time complexity!

Q3. `Max-Flow` and `Min-Cut` (4 marks)

Draw a flow graph with $n = 7$ vertices labeled from vertex 0 (source s), 1, 2, ..., vertex 6 (sink t) (The number of edges m and their respective capacities are up to you) so that the max flow value and the min cut value **differ by exactly one unit**. If it is not possible to do so, explain the reason.

Q4. `Edmonds-Karp` **(6 marks)**

Draw a flow graph with $n = 7$ vertices labeled from vertex 0 (source s), 1, 2, ..., vertex 6 (sink t) (The number of edges m and their respective capacities are up to you) so that Edmonds Karp's algorithm removes at least an edge from the residual graph because that edge is a bottleneck edge along the earlier (e.g. first) augmenting path but that deleted edge reappears after processing latter (e.g. second) augmenting path. Highlight the important steps in your answer. If it is not possible to do so, explain the reason.

Q5. Push-Relabel (4 marks)

Draw a flow graph with $n = 7$ vertices labeled from vertex 0 (source s), 1, 2, ..., vertex 6 (sink t) (The number of edges m and their respective capacities are up to you) so that Push-Relabel algorithm will perform **ZERO** relabel operation (the number of saturating and non-saturating push(es) are not constrained) en route to find the max flow value. If it is not possible to do so, explain the reason.

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

B CS4234 Review (40 Marks)

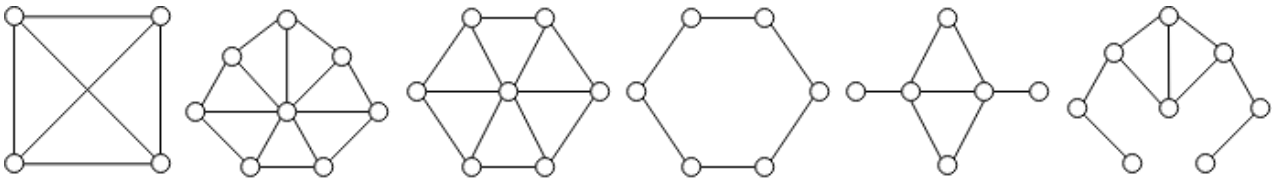
In lecture, we have discussed that we sometimes deal with special cases of NP-hard or NP-complete problems which can be solved in polynomial time, such as `Min-Vertex-Cover` on a tree. However, some special cases may still be NP-hard or NP-complete and are thus not (that) useful to consider. For example, it has been proven that `Min-Vertex-Cover` on a planar graph is still NP-hard. In this section, we will discuss a new problem, `3-Clique-Cover`, and look at some of its special cases.

B.1 `3-Clique-Cover` (40 Marks)

Given an **undirected, unweighted, and connected** graph $G = (V, E)$ with at least 2 vertices, let S be the set of (all possible) cliques in G. The `3-Clique-Cover` is a decision problem to decide whether it is possible to choose **at most 3 cliques** from S such that every vertex in V appears in **exactly one of the cliques**. We also define the **distance** between 2 distinct vertices in G as the number of edges in the (unweighted) shortest path connecting the 2 vertices. The diameter of a graph, $diam(G)$, is then the **maximum distance** between any 2 vertices in G.

Part A (6 marks)

Demonstrate your understanding of the above on the following graphs. For each graph, state the diameter of the graph and whether it can be covered by **at most 3 cliques**. If it can be covered, please circle the cliques used to cover the graph.



Part B (4 marks)

Prove that `3-Clique-Cover` is NP-hard by reduction from `3-Coloring`, which asks whether the vertices of a graph G can be colored using 3 colors such that no 2 adjacent vertices have the same color. You do not need to prove that `3-Coloring` is NP-hard.

Part C (4 marks)

Formulate an ILP and explain how to use it to solve `3-Clique-Cover`.
You can use variables V, E, S as mentioned above and you can write in pseudo-code.

Part D (5 marks)

Prove that if $diam(G) \geq 6$ then G cannot be covered with 3 cliques.
Hint: Consider the two vertices u, v with the furthest distance, which is at least 6, and the shortest path connecting these 2 vertices. Also, one of the graphs in Part A may give you a clue.

Part E-F-G (21 marks)

We have already seen one special case, $diam(G) \geq 6$, that can be solved in $O(1)$ time since the answer is always a 'NO'. You will now be given some other special cases for `3-Clique-Cover`. For each of them, decide whether the special case is solvable in polynomial time, or if it is still NP-hard. If it is still NP-hard, prove it. Otherwise, provide a working polynomial time algorithm to solve it and state its time complexity. You can use the results from previous parts in your answer.

E). $diam(G) = 1$ (2 marks)

F). $diam(G) = 2$ (7 marks)

G). $diam(G) = 3$ (12 marks)

Hint: Consider the two vertices u and v with the furthest distance, which is 3, and suppose G can be covered by 3 cliques C_1, C_2, C_3, which can be empty. What can we say about each vertex in G?

Section B Marks = (__) + (__) + (__) = ___

C Optimization Problem (30 Marks)

Let's name this problem as `Project-Grouping`. A certain NUS module (not necessarily CS4234) with n students has an end-of-semester team project. The lecturer is aware which pair of students don't work well together. These pair of students should **NOT** be in the same project group. Each student has to be given a project group and no student should be in two different project groups. Obviously creating n project groups with one student each is a waste of resources, so the lecturer wants to minimize the number of project groups that will be formed. The lecturer has one more additional constraint, each project group can not have more than c students in it (obviously $1 \leq c \leq n$).

Design an algorithm to group these students optimally. Given the number of students n (students are numbered from [0..n-1]) and list of m distinct pairs of students X who don't work well together (obviously $0 \leq m \leq n \times (n-1)/2$), find the minimum number of project groups required.

Part A (1+1+2+2+2 = 8 Marks)

To show your understanding of this problem, what is the optimal answers for the following test cases:
Example: $n = 2$, $m = 0$, $X = \{\}$, $c = 2$.
The answer is obviously 1 project group, put both $\{0, 1\}$ in the only group (please follow this format).

1. $n = 2$, $m = 0$, $X = \{\}$, $c = 1$.

2. $n = 2$, $m = 1$, $X = \{(0, 1)\}$, $c = 2$.

3. $n = 3$, $m = 2$, $X = \{(0, 2), (1, 2)\}$, $c = 3$.

4. $n = 9$, $m = 9$, $X = \{(0, 1), (0, 2), (1, 2), (3, 4), (3, 5), (3, 6), (4, 5), (4, 6), (5, 6)\}$, $c = 3$.

5. $n = 17$, $m = 1$, $X = \{(0, 1)\}$, $c = 5$.

Part B (2 Marks)

Do you happen to know the real name of this problem?

Part C (20 Marks)

Is `Project-Grouping` an NP-hard optimization problem?

- If yes, prove that it is NP-hard and then propose an exponential algorithm to solve it.
  If you choose this route, you can assume that n is a very small integer.
- If no, please solve `Project-Grouping` optimally using a polynomial time algorithm.
  If you choose this route, propose the largest n so that plugging n to the worst time complexity of your algorithm is still far smaller than 2×10^8 operations (the rule of thumb of 1 second time limit in most programming contest problems).

– End of this Paper, All the Best, You can use this Page 10 for extra writing space –

Section C Marks = (__) + __ = ___