National University of Singapore

School of Computing

CS4234 - Optimisation Algorithms

(Semester 1: AY2020/21)

Date and Time: Thursday, 08 October 2020, 12.04-13.44 (100m)

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 FOURTEEN (14) printed pages, including this page.
   However, only page 3 to 8 (6 pages) contain questions.
   Page 9-14 (6 pages) are the empty boxes (the answer sheet).
3. This is an **Open Book/Open Laptop/Open PC Assessment**.
   But you are NOT allowed to use the Internet.
   Students who take this paper from their room will only use the Internet for Zoom e-proctoring.
4. Answer **ALL** questions within the **boxed space at the back** of this booklet.
   Thus, students who take this paper from their room only need to scan/photo the answer sheet.
   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 (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, run Kruskal's on graph G', etc.
7. All the best :)

Write your Student Number in the box below:

A	0							

– This page is intentionally left blank so that the answer sheet starts from even numbered page –

A Draw Test Cases (30 marks)

For all parts below, please ensure that your test cases comply with the requirements, i.e., use the correct number of vertices, with the stipulated vertex labels, use the correct number of edges, etc. The grading for this part will be quite fast and strict, i.e., not much partial marks.

A-1. `Min-Weight-Vertex-Cover (MWVC)=Max-Weight-Independent-Set (MWIS)` (6 marks)

Draw a vertex (integer) weighted connected undirected graph with exactly 7 vertices (labeled as [0..6]) and exactly 6 edges such that its MWVC value is equal to its MWIS value.

Put your answer on the answer sheet, box A-1.
To facilitate quick grading, circle the vertices **that are part of the MWVC**.
If it is not possible to draw this test case given the constraints, explain the reason.

A-2. `Dominating-Set` (6 marks)

In class, we learn about `Dominating-Set`. In case you forget, here is a copy: DOMINATING-SET of a graph $G = (V, E)$ is a subset D of V such that every vertex not in D is adjacent to at least one member of D. We usually want to find the domination number $\gamma(G)$, the smallest size of a valid D. This problem is similar but not the same as the MIN-VERTEX-COVER problem. This problem can also be written as a MIN-SET-COVER problem.

Draw an unweighted undirected graph with exactly 7 vertices (labeled as [0..6]) and exactly 12 edges such that its domination number is 2. Moreover, this graph cannot have odd length cycle.

Put your answer on the answer sheet, box A-2.
To facilitate quick grading, circle the vertices **that are part of the Dominating-Set**.
If it is not possible to draw this test case given the constraints, explain the reason.

A-3. `Steiner-Tree` (6 marks)

Draw a weighted undirected graph with exactly 6 vertices (labeled as [0..5]) (the number of edges m is up to you) with exactly 3 required vertices (must be vertex 0, 1, and 2) and up to $6 - 3 = 3$ other Steiner vertices (i.e., vertex 3, 4, and/or 5 can be candidate Steiner vert(ices)) such that *any combination* of Steiner vertices (including not using any Steiner vertex at all) results in equally optimal Steiner Tree. Moreover, the weight of the optimal Steiner Tree of these 3 required vertices also equals to the weight of the MST of all 6 vertices.

Put your answer on the answer sheet, box A-3.
To facilitate quick grading, write a short sentence on why your test case works.
If it is not possible to draw this test case given the constraints, explain the reason.

A-4. `Traveling-Salesman-Problem (TSP)` (6 marks)

Create a complete weighted (you can assign any integer weight) undirected graph with exactly 5 vertices (labeled as [0..4]), i.e., K_5 and $5*4/2 = 10$ edges such that its TSP tour/cycle length is 77.

Put your answer on the answer sheet, box A-4.
To facilitate quick grading, highlight the **TSP tour/cycle**.
If it is not possible to draw this test case given the constraints, explain the reason.

A-5. `Dinic`'s Algorithm (6 marks)

Draw a flow graph with $n = 8$ vertices labeled from vertex 0 (source s), 1, 2, ..., vertex 7 (sink t) (the number of edges m and their respective capacities are up to you) such that Dinic's algorithm finds only two level graphs: level graph $L = 3$ first (+ finds **just 1 unit of blocking flow**) and then level graph $L = 7$ (+ finds **another 6 units of blocking flow(s)**), before it stops with a total of $1+6 = 7$ units of maximum flow. You can assume that this Dinic's algorithm break ties by favoring vertices with lower vertex number.

Put your answer on the answer sheet, box A-5.
To facilitate quick grading, **highlight the selected augmenting paths**.
If it is not possible to draw this test case given the constraints, explain the reason.

B The Longest Ranking Chart (40 Marks)

You are a reporter of a "Competitive Sport for 2-3-4" event. In this event, there are $N \geq 2$ contestants. Each pair of two contestants will battle head-to-head, and they either result in one's winning or a draw. Thus, there are exactly $\frac{N \times (N-1)}{2}$ matches in total.

You want to write a spicy news, because of course that's how you make profit. Therefore, you want to create the longest possible ranking chart. A **ranking chart** is a sequence of **distinct** contestants $x_1, x_2, \cdots, x_k$; such that x_i wins against x_{i+1} for all $1 \leq i < k$. The longest ranking chart is basically maximizing the value of k.

But of course, you still have a brilliant idea to make this even spicier. Contestant 1 is apparently a fan-favorite. A **spicy ranking chart** is basically a ranking chart where $x_1 = 1$. The longest spicy ranking chart is maximizing the value of k given that $x_1 = 1$.

Part 1: Original Scenario (12 marks)

You are given these 3 complete result tables. Let's call them Left, Middle, and Right table, respectively.

vs	1	2	3	4
1		DRAW	3 WINS	1 WINS
2	DRAW		DRAW	2 WINS
3	3 WINS	DRAW		DRAW
4	1 WINS	2 WINS	DRAW	

vs	1	2	3	4
1		1 WINS	3 WINS	4 WINS
2	1 WINS		3 WINS	4 WINS
3	3 WINS	3 WINS		3 WINS
4	4 WINS	4 WINS	3 WINS	

vs	1	2	3	4
1		DRAW	DRAW	DRAW
2	DRAW		2 WINS	4 WINS
3	DRAW	2 WINS		3 WINS
4	DRAW	4 WINS	3 WINS	

Figure 1: Left, Middle, and Right table, respectively

B-1-1 (6 marks)

For each table, please write any longest ranking chart and also any longest spicy ranking chart!
Put your answer on the answer sheet, box B-1-1.

B-1-2 (3 marks)

Given an $N \times N$ complete result table, how hard is it to construct any longest ranking chart?
Please prove its hardness or give the fastest algorithm to solve this!
Put your answer on the answer sheet, box B-1-2.

B-1-3 (3 marks)

How about the longest spicy ranking chart?
Please prove its hardness or give the fastest algorithm to solve this!
Put your answer on the answer sheet, box B-1-3.

Part 2: Strength Competition Scenario (6 marks)

Let's define the following assumption: the sport event is apparently a **strength competition**. Each contestant has their own strength level denoted by an integer. Contestant i has strength level of s_i. Contestant i win against contestant j if and only if $s_i > s_j$. The draw occurs when $s_i = s_j$. Note that we do not know the actual strength level of each contestant.

B-2-1 (4 marks)

Given an $N \times N$ complete result table and it is a **strength competition**,
how hard is it now to construct any longest ranking chart?
Please prove its hardness or give the fastest algorithm to solve this!
Put your answer on the answer sheet, box B-2-1.

B-2-2 (2 marks)

How about the longest spicy ranking chart?
Please prove its hardness or give the fastest algorithm to solve this!
Put your answer on the answer sheet, box B-2-2.

Part 3: Incomplete Scenario (10 marks)

Sometimes, the result table is **incomplete**. That means the results of some of the matches are unknown. You, as a reporter, have a right to ~~manipulate~~ fill in the loophole of the table. For each unknown entry, you can assume its result whatever you want.

You are given these 2 incomplete result tables.

vs	1	2	3	4
1		DRAW	3 WINS	?
2	DRAW		?	2 WINS
3	3 WINS	?		DRAW
4	?	2 WINS	DRAW	

vs	1	2	3	4
1		?	?	?
2	?		2 WINS	4 WINS
3	?	2 WINS		3 WINS
4	?	4 WINS	3 WINS	

Figure 2: Table 3A (left) and 3B (right)

B-3-1 (4 marks)

For each table, please write any longest ranking chart and also any longest spicy ranking chart!
Put your answer on the answer sheet, box B-3-1.

B-3-2 (4 marks)

Given an $N \times N$ **incomplete** result table,
how hard is it now to construct any longest ranking chart?
Please prove its hardness or give the fastest algorithm to solve this!
Put your answer on the answer sheet, box B-3-2.

B-3-3 (2 marks)

How about the longest spicy ranking chart?
Please prove its hardness or give the fastest algorithm to solve this!
Put your answer on the answer sheet, box B-3-3.

Part 4: Strength Competition + Incomplete Scenario (6 marks)

B-4-1 (4 marks)

Given an $N \times N$ **incomplete** result table and it is a **strength competition** (B-2 + B-3),
how hard is it now to construct any longest ranking chart?
Note that for any unknown result, the reporter can ~~manipulate~~ fill in the loophole of the table in a way that strength competition constraint is still satisfied.
Please prove its hardness or give the fastest algorithm to solve this!
Put your answer on the answer sheet, box B-4-1.

B-4-2 (2 marks)

How about the longest spicy ranking chart?
Please prove its hardness or give the fastest algorithm to solve this!
Put your answer on the answer sheet, box B-4-2.

Part 5: No-Draw Scenario (6 marks)

For this part, let's have the following assumption: each match **never** ends in a draw!

B-5-1 (4 marks)

Given an $N \times N$ complete result table and there is **no draw** in the table,
how hard is it to construct any longest ranking chart?
Please prove its hardness or give the fastest algorithm to solve this!
Put your answer on the answer sheet, box B-5-1.

B-5-2 (2 marks)

How about the longest spicy ranking chart?
Please prove its hardness or give the fastest algorithm to solve this!
Put your answer on the answer sheet, box B-5-2.

C Min-Clique-Cover (Again) (30 Marks)

Given an undirected graph $G = (V, E)$, let X be the set of cliques in G. The Min-Clique-Cover is the minimum possible size of a subset X' of X such that every vertex in V appears in exactly one clique in X'. We will call this minimum size $C(G)$.

Given an algorithm A, let $A(G)$ be the answer returned by algorithm A to the Min-Clique-Cover on a graph G and let $R(G) = A(G)/C(G)$. Define $R(n) = \max\{R(G) | G \textit{ has at most } n \textit{ vertices }\}$. We say that algorithm A gives an $O(f(n))$ approximation if $R(n) \in O(f(n))$. Given that definition, it has been proven on a general graph G, for all $\epsilon > 0$, it is NP-hard to approximate Min-Clique-Cover within a factor of $O(n^{1-\epsilon})$ for a graph with n vertices. However, we can still design approximation algorithms such as the one below.

1. Let S be an empty set of cliques.
2. Check if G can be covered using 1 or 2 cliques. If so, add the 1 or 2 cliques to S, return S and terminate the algorithm.
3. Otherwise, create a duplicate of G called C and set all the vertices as unmarked.
4. Mark the vertex v in C that has the largest degree among all the unmarked vertices C. If there are multiple such vertices, choose any of them.
5. For all vertices in C that are not adjacent to v, remove them and the edges adjacent to them from C. If all vertices left in C are marked, move to step 6. Otherwise, repeat from step 4.
6. For each vertex in C, remove them and the edges adjacent to them from G. Add C to S. If G is empty, return S and terminate the algorithm. Otherwise, repeat from step 3.

Suppose $G = (V, E)$ is a simple graph (no repeated edges and no self loops) that has n vertices, m edges and let $C(G) = c$. Answer the following questions on the answer sheet, box C-A to C-G.

A). Explain why this algorithm returns a valid solution to the MIN-CLIQUE-COVER problem. (4 marks)

B). Find a graph G with the smallest number of vertices such that this algorithm will **sometimes** give a non-optimal answer to the MIN-CLIQUE-COVER on G. (2 marks)

C). Find a graph G with the smallest number of vertices such that this algorithm will **always** give a non-optimal answer to the MIN-CLIQUE-COVER on G. (2 marks)

D). Give an efficient polynomial time implementation for step 2 and state its time complexity. The algorithm must check whether a graph G can be covered in 1 or 2 cliques, and if so, find the cliques to cover G. (8 marks)

E). Show that the first time step 5 finishes executing, there are at least n/c vertices left in C. (4 marks)

F). Hence, show that the first clique added to S has at least $\log_c n$ vertices. (6 marks)

Hint: Let n_i be the number of **unmarked** vertices after the i-th iteration of step 5, with $n_0 = n$. Derive an inequality for n_{i+1} in terms of n_i based on the previous part.

G). Based on the previous part, we can show that $A(G) \in O(n/\log_c n)$ for this algorithm (you do not need to prove this). Show that this algorithm gives an approximation ratio of $O(n/\log n)$. (4 marks)

– End of this Paper, All the Best –

D 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	30		
B	40		
C	30		
Total	100		

Box A-1.

Box A-2.

Box A-3.

Box A-4.

Box A-5.

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

Box B-1-1.

Box B-1-2.

Box B-1-3.

Section B-1 Marks = __ + __ + __ = ___

Box B-2-1.

Box B-2-2.

Section B-2 Marks = __ + __ = ___

Box B-3-1.

Box B-3-2.

Box B-3-3.

Section B-3 Marks = __ + __ + __ = ___

Box B-4-1.

Box B-4-2.

Section B-4 Marks = __ + __ = ___

Box B-5-1.

Box B-5-2.

Section B-5 Marks = __ + __ = ___

Section B Marks = __ + __ + __ + __ + __ = ___

Box C-A.

Box C-B.

Box C-C.

Box C-D.

Box C-E.

Box C-F.

Box C-G.

Section C Marks = __ + __ + __ + __ + __ + __ + __ + __ = ___