National University of Singapore

School of Computing

CS4234 - Optimisation Algorithms

(Semester 1: AY2018/19)

Tuesday, 27 November 2018, EV (2 hours)

INSTRUCTIONS TO CANDIDATES:

1. Do **NOT** open this assessment paper until you are told to do so.
2. This assessment paper contains THREE (3) sections.
   It comprises TWELVE (12) printed pages, including this page.
3. This is an **Open Book Assessment**.
4. Answer **ALL** questions within the **boxed space** or **on free page 10 and 12**.
   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 *and its time complexity*, e.g. run $O(n^2m)$ Dinic's algorithm on flow graph G, run $O(n^3)$ Push-Relabel on flow graph G', run $O(n^3)$ Edmonds Matching on unweighted graph G'', run $O(n^3)$ Hungarian algorithm on weighted bipartite graph G''', etc.
7. Please write your Student Number in the box below (do **NOT** write your name):

A	0							

This portion is for examiner's use only

Section	Maximum Marks	Your Marks	Remarks
A	25		
B	45		
C	30		
Total	100		

A (P?-)easy Questions (25 marks)

Q1. Push-Relabel (5 marks)

Draw a small flow graph so that Push-Relabel algorithm will perform at least 1 relabel operation, at least 1 saturating push operation, and at least 1 non-saturating push operation in order to get the correct max flow value **(3 marks)**:

To facilitate quicker grading, please roughly explain the process taken by Push-Relabel algorithm on your own small flow graph and on why it satisfies the requirements for this question **(2 marks)**:

Q2. Graph Matching (5 marks)

Draw a small unweighted bipartite graph so that it is **very unlikely (probability** $< 5\%$**)** that the **randomized** greedy pre-processing step that matches random trivial augmenting paths that consist of 'free vertex on left set, a free edge, a free vertex on right set' will be super lucky and no (zero) actual Augmenting Path Algorithm step being used at all **(3 marks)**:

To facilitate quicker grading, please roughly explain the process taken by Augmenting Path Algorithm++ on your own small unweighted bipartite graph and on why it satisfies the requirements for this question **(2 marks)**:

Q3. Which Algorithm is the Best? (6 Marks)

For this question, you will be given several optimization problem scenarios (2 marks each). For each one of them, please mention the best algorithm and its time complexity so that the optimization problem is solved *correctly* with the worst case time complexity that is *as low as possible*.

1. Solve `Max-Flow`. The input flow graph has $n = 10\,000$ vertices but only $m = 20\,000$ edges. The edge capacities are positive integers not more than 1 000. The maximum possible max flow value is $f^* \leq 1\,000$.

2. Solve `Min-Cut`. The input graph is grid-based flow graph $r = 1\,000$ rows and $c = 1\,000$ columns. The source vertex is connected to one (or more) cells inside this grid. All cells at the top-most row/bottom-most row/left-most column/right-most column are all connected to the sink vertex. The edge capacities are all ones.

3. Solve `Min-Weight Max-Cardinality-Bipartite-Matching`. The input bipartite graph is a $K_{n,m}$ where $1 \leq n, m \leq 100$. The edge weights are positive integers not more than 1 000.

Q4. Statements About SLS (9 marks)

For each statement below about Stochastic Local Search (SLS) algorithm, determine if it is More towards True/It depends/More towards False and give a short explanation (3 marks per question).

1. All NP-hard Metric No-Repeat TSP instances have similar structure and thus we should just design one good SLS algorithm for all instances in order to achieve the universality requirement.

2. Using delta evaluations between the current search position s and its neighbors s' instead of re-computing the objective value of s' from scratch is necessary in order to get *better SLS algorithm (produce results closer to optimal values or best known values)*. You can assume that you can run your SLS algorithms until it terminates without any limited run time constraint.

3. Suppose you are given a problem X (that is new for you) that you have to solve. After performing some 'research', you found that this X is actually an NP-hard optimization problem. Your best course of action is to design a new SLS algorithm variant (e.g. Tabu Search) in order to get reasonable good results in reasonable run time.

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

B `Min-Set-Cover` Variants (45 marks)

Some problems such as `Min-Set-Cover` are **NP-hard to approximate to a constant factor**. It was shown in lecture that there is a greedy algorithm that gives an $O(\log n)$ approximation to the `Min-Set-Cover` problem (which is not a constant factor) but we did not really elaborate further. In this question, we will explore a few problems that can be reduced to `Min-Set-Cover` and see if they are also NP-hard to approximate to a constant factor too.

B.1 `Min-Edge-Cover` (20 marks)

Given an undirected, connected graph $G = (V, E)$ with more than 1 vertex, the `Min-Edge-Cover` of the graph is the minimum number of edges that need to be selected from E such that every vertex in V is adjacent to at least one of those selected edges. A **small** example: $G = (V = \{0, 1, 2, 3\}, E = \{\{0, 1\}, \{1, 2\}, \{2, 3\}\})$. The optimal answer is to pick 2 edges $\{\{0, 1\}, \{2, 3\}\}$.

To ensure your understanding of this problem, what are the answers for G_1, G_2, and G_3 (**4 marks**)?
$G_1 = (V = \{0, 1\}, E = \{\{0, 1\}\})$,
$G_2 = (V = \{0, 1, 2, 3\}, E = \{\{0, 1\}, \{1, 2\}, \{2, 3\}, \{3, 0\}\})$
$G_3 = (V = \{0, 1, 2, 3, 4, 5\}, E = \{\{0, 1\}, \{1, 2\}, \{2, 0\}, \{1, 3\}, \{1, 4\}, \{1, 5\}\})$

Show how to reduce this problem to `Min-Set-Cover` (**4 marks**).

Express `Min-Edge-Cover` as an ILP and explain how to construct the solution from the ILP solution and why it is correct. (**4 marks**).

Is this problem NP-hard? If so, prove it and provide a working exponential algorithm to solve it. Otherwise, provide a polynomial time algorithm to solve it (both routes worth **8 marks**).

B.1 Marks = ___ + ___ + ___ + __ = ___

B.2 `Min-Clique-Cover` (25 marks)

Given an undirected graph $G = (V, E)$, let S be the set of cliques in G. The `Min-Clique-Cover` is the minimum possible size of a subset S' of S such that every vertex in V appears in at least one clique in S'. We will call this minimum size $C(G)$. A **small** example: $G = (V = \{0, 1, 2, 3\}, E = \{\{0, 1\}, \{1, 2\}, \{2, 0\}\})$. The optimal answer is to pick 2 cliques $\{\{0, 1, 2\}, \{3\}\}$.

To ensure your understanding of this problem, what are the answers for G_1, G_2, and G_3 (**4 marks**)?
$G_1 = (V = \{0, 1\}, E = \{\{0, 1\}\})$,
$G_2 = (V = \{0, 1, 2, 3\}, E = \{\{0, 1\}, \{1, 2\}, \{2, 3\}, \{3, 0\}\})$
$G_3 = (V = \{0, 1, 2, 3, 4, 5\}, E = \{\{0, 1\}, \{1, 2\}, \{2, 0\}, \{1, 3\}, \{1, 4\}, \{1, 5\}\})$

Show how to reduce this problem to `Min-Set-Cover` (**4 marks**).

Hence, derive a simple $O(\log |V|)$-approximation to the `Min-Clique-Cover` problem. Does this show that finding an $O(\log |V|)$-approximation to the `Min-Clique-Cover` problem is not NP-hard? Why or why not? (**4 marks**).

Suppose the graph G is planar. Design a linear time 4-approximation algorithm and prove that it returns a valid answer that is a 4-approximation. (**4 marks**). Hint: In graph theory, Kuratowski's theorem is a mathematical forbidden graph characterization of planar graphs: "A finite graph is planar if and only if it does not contain a subgraph that is a subdivision of K_5 or of $K_{3,3}$".

Use the ideas from the `Min-Edge-Cover` problem to design a 2-approximation algorithm to the `Min-Clique-Cover` problem on a planar graph. Prove its correctness and its approximation ratio. (**9 marks**).

B.2 Marks = ___ + ___ + ___ + ___ + __ = ___

Section B Marks = ___ + __ = ___

C (NP?-)hard Questions (30 Marks)

C.1 Duty Roster (15 marks)

Disclaimer: The following question is modified from a Kattis problem.

Steven lives in Sheares Hall as one of the n Resident Fellows (RFs) where he has to be on (standby) duty on specific weeks of the month. On each week, there should be exactly 2 RFs[1] on duty (to minimize the occurrences of unwanted cases). However, RFs are also (academic) staffs who frequently travel overseas for conferences (or for Steven's case: Programming Competitions). So, just before the start of each semester the hall office ask each RFs to submit their availability for the next w weeks.

Now, write a program that will take an $n \times w$ Boolean matrix of the n RFs and their availabilities for the w weeks of this semester and assign exactly two RAs for duty on each week. While there is no limit for how many weeks each RF can be on duty, hall office staff in charge wants to reduce unfairness by limiting the imbalance between the number of weeks each RF is assigned[2]. Thus, your program needs to minimize the maximum number of weeks to which any RFs may be assigned.

An example: Suppose there are $n = 5$ RFs who are actually not traveling at all this semester ($w = 18$, 1 hall orientation week + 13 teaching weeks + 1 recess week + 1 study week + 2 exam weeks), so the $n \times w = 5 \times 18$ Boolean matrix contains all true. Then the answer is 8, i.e. RF1 and RF2 paired for first 8 weeks; RF3 and RF4 paired for next 6 weeks; RF3 and RF5 paired for the next 2 weeks; and lastly RF4 and RF5 paired for the last 2 weeks. This way, no RF works more than 8 weeks this semester.

To ensure your understanding of this problem, what is the answer if the $n \times w$ contains all true, $n = 2$, $w = 52$ (**2 marks**)?

Is this problem NP-hard? If so, prove it and provide a working exponential algorithm to solve it. Otherwise, provide a polynomial time algorithm to solve it (both routes worth **13 marks**).

[1] In real life, actually there is only one Duty RF per week, but the Hall Master is usually also available, so this statement is actually almost true.

[2] In real life, we are actually assigned using a black-box greedy (or maybe randomized) algorithm :O...

– You can use this Page 10 for extra writing space –

C.2 `Max-Non-Divisible-Subset` (15 marks)

Given a set S of positive (not necessarily distinct) integers, the problem of `Max-Non-Divisible-Subset` is to find the *cardinality* of the largest subset S' of S such that no two numbers in the subset are divisible by each other. A **small** example: $S = \{7, 15, 7, 14, 30, 5\}$. The optimal answer is subset $S' = \{7, 5\}$ (or $\{7, 15\}$, $\{7, 30\}$, etc) with largest cardinality 2.

To ensure your understanding of this problem, what are the answers for $S_1 = \{1, 2, 4, 8, 16, 32, 63\}$, $S_2 = \{13, 3, 2, 7, 11, 5, 17\}$, and $S_3 = \{7, 7, 7, 7, 7, 7, 7\}$ (**3 marks**)?

Is this problem NP-hard? If so, prove it and provide a working exponential algorithm to solve it. Otherwise, provide a polynomial time algorithm to solve it (both routes worth **12 marks**).

Section C Marks = (___ + ___) + (___ + ___) = ___

– End of this Paper, All the Best, You can use this Page 12 for extra writing space –