National University of Singapore

School of Computing

# CS4234 - Optimisation Algorithms

(Semester 1: AY2018/19)

Date and Time: Wednesday, 10 October 2018, 12.05-13.40 (95m)

---

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 TWELVE (12) 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 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, 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. `Min-Set-Cover` Instance (6 marks)

Let's assume that we have a set $X$ with $n = 32$. To simplify the question, assume that $X = \{1, 2, 3, \ldots, 30, 31, 32\}$. Create a `Min-Set-Cover` instance with that $X$ and your chosen set $S = \{S_1, S_2, \ldots, S_m\}$ (The number of subsets $m$ is up to you) so that your test case makes the $O(\ln n)$-approximation `Greedy-Set-Cover` as discussed in class produces a solution that requires $\lfloor \ln 32 \rfloor = \lfloor \log_e 32 \rfloor = \lfloor 3.46 \rfloor = 3$ times more subsets than the optimal answer.

## Q2. `Euclidean-Steiner-Tree` Instance (6 marks)

Draw a good solution (it does *not* have to be optimal) for this `Euclidean-Steiner-Tree` instance.

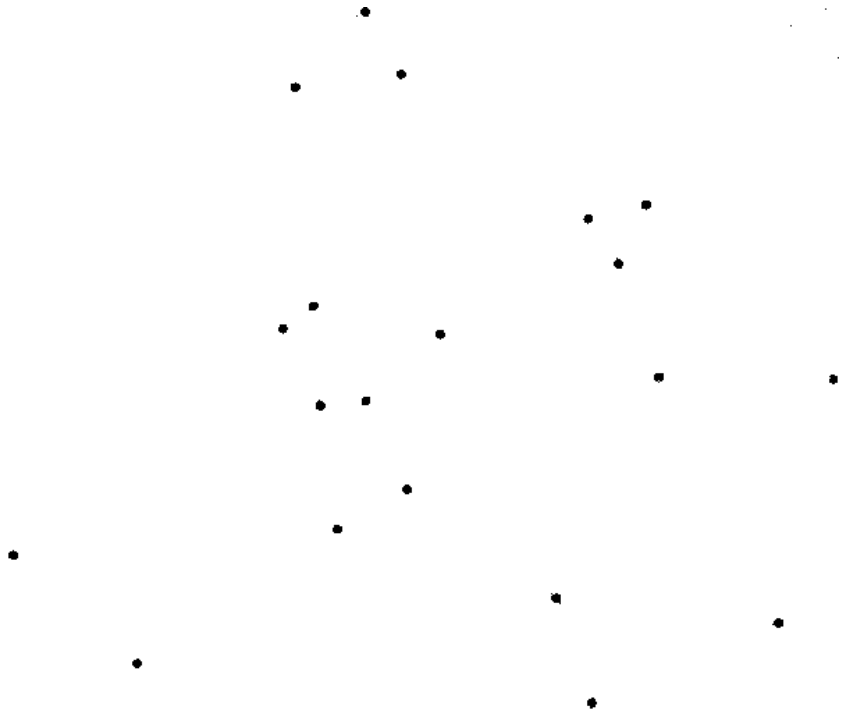


Figure 1: Draw Your Best Form of `Euclidean-Steiner-Tree` Here (20 (TWENTY) Dots)

**Q3.** `Max-Flow` **(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 Dinic's algorithm will find all level graphs 1 (the lowest level graph where the source and the sink are connected via a direct edge), 2, 3, 4, 5, 6 (the highest level graph) and find that the max flow value is *exactly* 6 units.

**Q4.** `Min-Cut` **(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) so that there are *exactly* 36 different `Min-Cuts`.

## Q5. Special Flow Graphs (6 marks)

There are three special flow graphs in Figure 4, 5, and 6. Find the `Max-Flow` **value** from the source vertex $s$ to the sink vertex $t$ on each one of them using *the best* algorithm that solves each of the special cases. Explain in one short sentence on why the algorithm (of your choice) is correct and write down its tightest time complexities.


Figure 2: A Connected Directed +ve Weighted (notice varying weights) Line Graph with $n+2$ Vertices

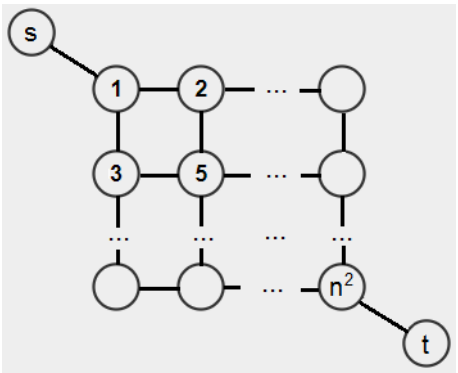



Figure 3: A Connected Un-directed Un-weighted (weight = 1) Grid Graph with $n^2 + 2$ Vertices

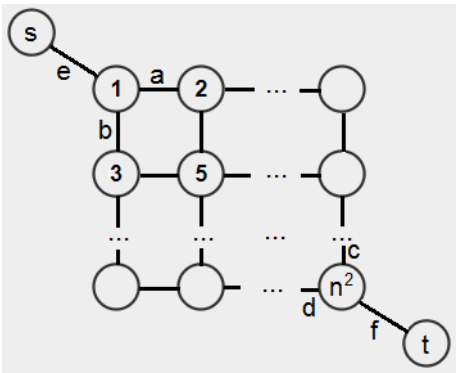



Figure 4: A Connected Un-directed +ve Weighted (notice $a$, $b$, ..., $f$) Grid Graph with $n^2 + 2$ Vertices



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

# B CS4234 Review (40 Marks)

## B.1 Review 1 (25 Marks)

In tutorial, we discussed the `Min-Weight-Feedback-Edge-Set` (`MWFES`) problem which involves removing edges with the minimum total weight so that the graph is acyclic. In this question, we assume the `MWFES` problem is posed on an **undirected**, weighted graph and we do not consider bidirectional edge as a trivial cycle.

Now consider the following similar problem: Given an undirected, weighted graph $G = (V, E)$, find the minimum total weight of edges that need to be removed so that the `Max-Clique` of the graph is at most 2.

### Part A (4 marks)

Does a solution to the `MWFES` problem always satisfy the constraints of the given problem? Explain your answer.

### Part B (4 marks)

Conversely, does a solution to the given problem always satisfy the constraints of the `MWFES` problem? Explain your answer.

### Part C (8 marks)

Formulate the given problem as an ILP problem and write out the set of constraints (you can use 'pseudo-code' to help you write the constraints). Show how to construct the solution to the given problem out of the solution to the ILP and explain briefly on why it is correct.

**Part D (9 marks)**

Relax the Integer constraint to get an LP problem. Show how you can use rounding to obtain a 3-approximation to the given problem and explain why the solution is a valid 3-approximation. (You may have to slightly modify your ILP from the previous part)

## B.2 Review 2 (15 Marks)

In the area of computational biology, DNA sequences are studied in-depth and many algorithms have been developed for this purpose. One of the *common* problems is as follows: Given a set of DNA sequences (strings) $S$, find the shortest DNA sequence (string) that contains all the sequences (strings) in $S$. A sequence $A$ contains a sequence $B$ if $B$ is a contiguous substring of $A$. For example, if the sequences are $\{TAT, GTA, ATACG\}$ then the answer would be $GTATACG$.

In DNA sequences there will only be at most 4 proteins that can appear in the sequence $G$, $T$, $A$ and $C$. Here, we will consider a general version of this problem where there are infinitely many different types of proteins and we will assume that there is no sequence is $S$ which is a substring of another sequence.

### Part A (1 mark)

What is the name of this *common* computational biology problem?

### Part B (1 mark)

Is this *common* computational biology problem an NP-hard optimization problem?
(Just state yes/no, the proof is not needed).

### Part C (2 marks)

Suppose (**hypothetically**) that an approximation algorithm $P$ is found that gives a 3-approximation to the `Min-Weight-Set-Cover` (`MWSC`) problem in polynomial time with respect to the number of subsets and number of items covered by the sets.

Is this **hypothetical** 3-approximation algorithm $P$ for the `MWSC` problem a better algorithm than the $O(\log n)$-approximation `Greedy-Set-Cover` algorithm that we have discussed in class on the unweighted version of `Min-Set-Cover` problem? Why?

**Part D (8 marks)**

Now with the **hypothetical** 3-approximation algorithm $P$ for the `MWSC` problem, please show that you can use $P$ to construct a 6-approximation to the given problem in polynomial time with respect to the sum of the length of the sequences in the set $S$. (Hint: Consider a set containing all possible ways to overlap two of the sequences with weight equal to the length of the combined sequence).

**Part E (3 marks)**

Show that the reduction to the `MWSC` problem from the given problem runs in polynomial time.



Section B Marks = ___ + ___ = ___

# C Decision Problem (30 Marks)

Flow Free (**NOT** `Max-Flow`) is a puzzle that is played on a 2D grid of cells, with some cells marked as endpoints of certain colors and the rest of cells being blank.

To solve the puzzle, you have to connect each pair of colored endpoints with a path, following these rules:

1. There is a path connecting two points with the same color, and that path also has that color,
2. All cells in the grid are used and each cell belongs to exactly one path (of the same color as the endpoints of the path)
3. The different colored paths cannot intersect.

The path is defined as a connected series of edges where each edge connects two neighboring cells. Two cells are neighbors if they share a side. Therefore, each colored cell will be an endpoint of exactly one edge and each blank cell will be an endpoint of exactly two edges.

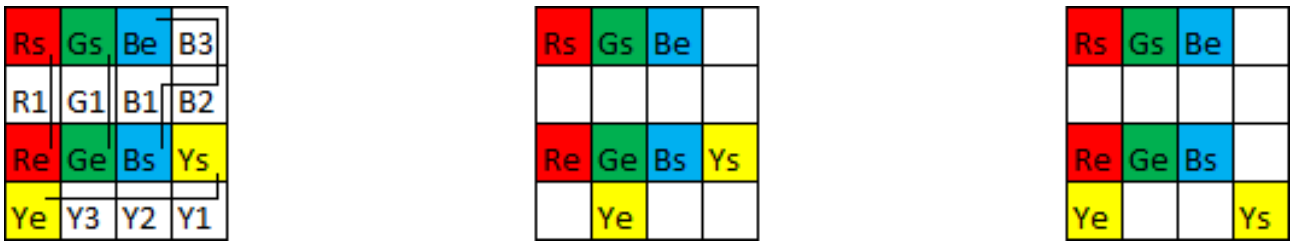



Figure 5: Left Example: `Solvable` + the Certificate; Middle and Right Examples: `Not-Solvable`

Your task is to determine if a given puzzle is solvable or not.

### Part A (3 Marks)

Let's consider an $N \times M$ grid ($N > 1$ and $M > 1$) and **assume that there is only one pair of neighboring endpoints with the same color at the side of the grid, e.g. Red**. Let's call this the `Special-Flow-Free` puzzle. To show that you have understood the requirements, please decide the answers for these 3 test cases and draw the required 'certificate' for all '`Solvable`' test cases in Figure 8 directly.

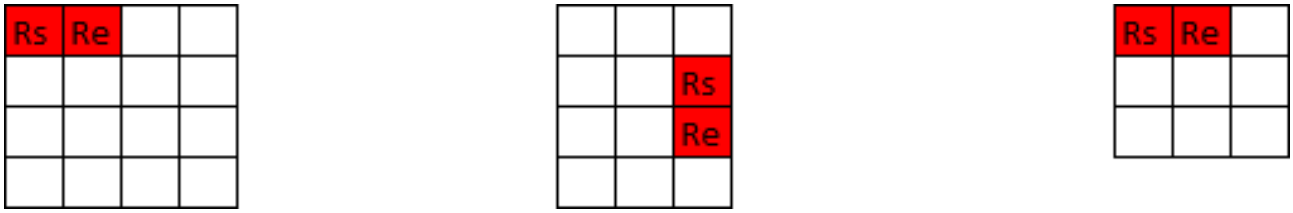



Figure 6: Decide The Answers and Draw the 'Certificates' for all '`Solvable`' test cases

**Part B (12 Marks)**

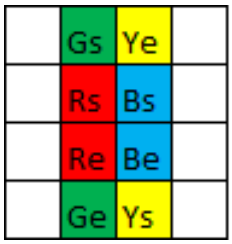
Is `Flow-Free-One` an NP-complete decision problem?

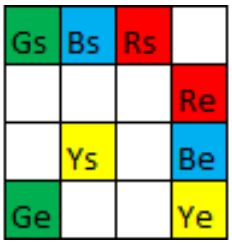
- If yes, prove that it is NP-complete and then propose an exponential algorithm to solve it
  This route worth full 12 marks.
- If no, please solve `Flow-Free-One` fully using a polynomial time algorithm
  This route worth only 11 marks.

Your (exponential or polynomial) algorithm must give a certificate for all 'yes' test cases or print '`Not-Solvable`' otherwise.

**Part C (3 Marks)**

Let's consider an $N \times M$ grid ($N > 1$ and $M > 1$) and **assume that there are exactly four pairs of endpoints (not necessarily neighboring/at the side of the grid) with the same color, e.g. Red, Green, Blue, Yellow**. Let's call this the `Flow-Free-Four` puzzle. Figure 7 at the beginning of this question is exactly this variant. To show that you have understood the requirements, please decide the answers for these 3 other test cases that are different from Figure 7 and draw the required 'certificate' for all '`Solvable`' test cases in Figure 9 directly.



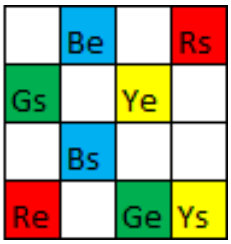




Figure 7: Decide The Answers and Draw the 'Certificates' for all '`Solvable`' test cases

**Part D (12 Marks)**

Is `Flow-Free-Four` an NP-complete decision problem?

- If yes, propose an exponential algorithm to solve it for $M = 4$ and $N = 4$ case.
  This route worth only 11 marks.
- If no, please solve `Flow-Free-Four` using a polynomial time algorithm.
  This route worth full 12 marks.

Again, your (exponential or polynomial) algorithm must give a certificate for all 'yes' test cases or print '`Not-Solvable`' otherwise.

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