National University of Singapore

School of Computing

CS4234 - Optimisation Algorithms

(Semester 1: AY2016/17)

Date and Time: Tuesday, 04 October 2016, 12.05-13.35 (90m)

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 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 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 G, run Kruskal's on G', etc.
7. 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	25		
C	30		
Total	100		

A Be the Computer (45 marks)

For **Part 1-4** of this **Section A**, you are given a 'small' instance of a disconnected graph. In **Part 1** and **Part 3**, the graph is weighted and the weights are indicated as integers inside the circles/vertices. In **Part 2** and **Part 4**, the graph is unweighted, i.e. you can treat each vertex as having weight 1.

A.1 Part 1 (8 marks)

Solve the Min-Weight-Vertex-Cover (MWVC) on Figure 1 below by **circling the vertices** that you want to put inside the minimum weight vertex cover **and write down** the value of the optimal answer in the box provided. Answer that is a vertex cover but **not** the minimum weight will be given partial marks only if it is not worse than 2-times the minimum weight.

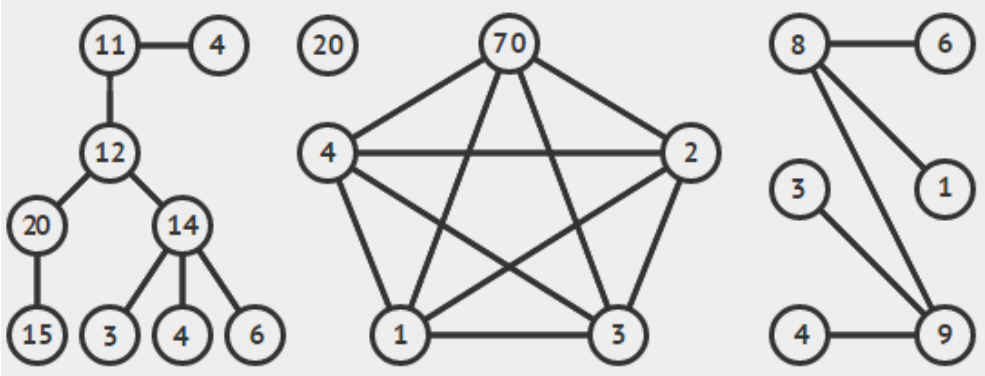

Figure 1: Instance for Part 1, circle the vertices in your MWVC and write the minimum weight below:

A.2 Part 2 (6 marks)

Please redo the question from **Part 1** above, but now the vertices in Figure 2 are unweighted.

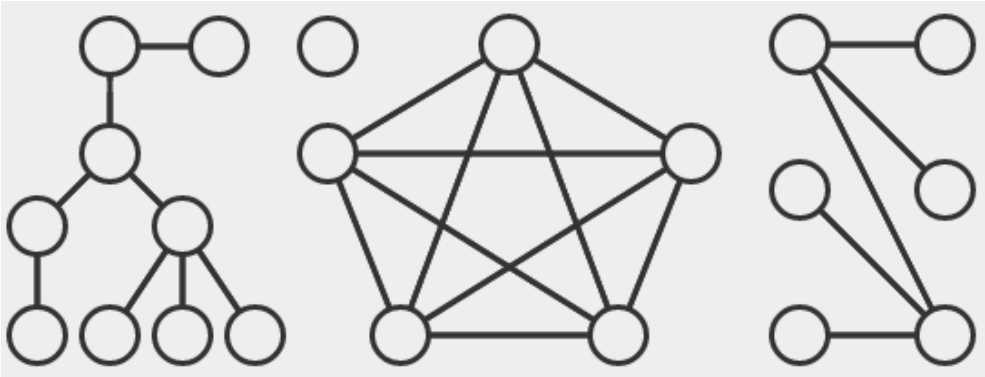

Figure 2: Instance for Part 2, circle the vertices in your MWVC and write the minimum weight below:

Page 2 Marks = ___

A.3 Part 3 (4 marks)

Now, what if we have to solve a different problem instead: The MAX-WEIGHT-INDEPENDENT-SET (MWIS) that is mentioned in T04. If you forget the definition, here it is: "Given a graph $G = (V, E)$, pick the maximum-weight set $I \subseteq V$ so that no two vertices in I share an edge.". Solve the problem on the graph in Figure 3 (which is exactly the same as in **Part 1** above).

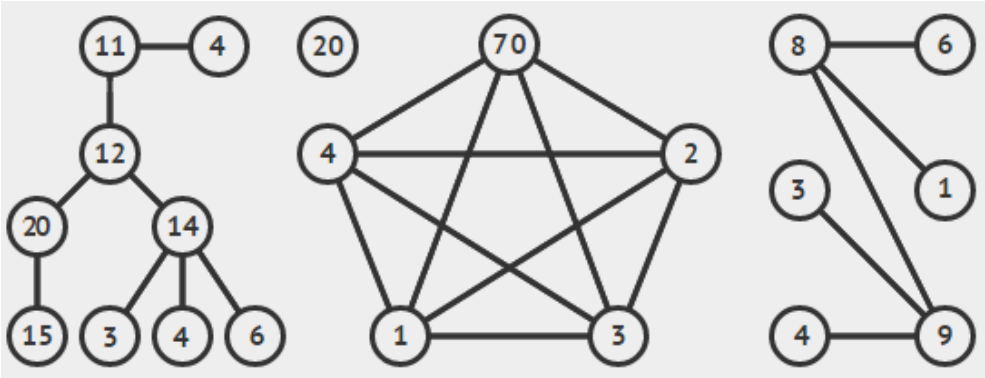

Figure 3: Instance for Part 3, circle the vertices in your MWIS and write the maximum weight below:

A.4 Part 4 (2 marks)

Now solve the MWIS problem on the unweighted graph in Figure 4 (which is exactly the same as in **Part 2** above).

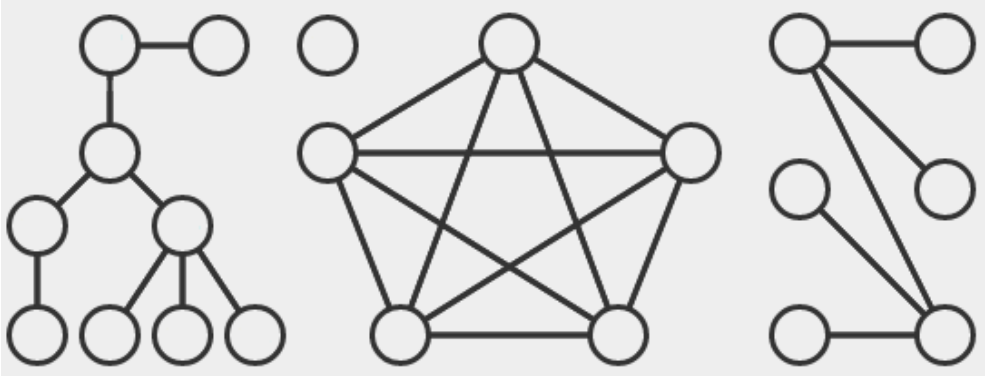

Figure 4: Instance for Part 4, circle the vertices in your MWIS and write the maximum weight below:

Page 3 Marks = ___

A.5 Part 5 (9 marks)

Now we switch to Lecture 3b and T2: The EUCLIDEAN-STEINER-TREE problem. Based on your understanding on the structure of an optimal EUCLIDEAN-STEINER-TREE solution, try your best to derive a manual solution for this instance below with 8 points on 2D Euclidean space. The actual coordinates of the 8 points are ignored so that you can concentrate on the *structure* of the optimal Euclidean Steiner Tree. Answer that is an Euclidean Steiner Tree but not the minimum Euclidean Steiner Tree will be given some partial marks if it is not worse than 2-times the optimal answer.

Figure 5: Instance for Part 5, draw your best Euclidean Steiner Tree of these 8 points.

A.6 Part 6 (4 marks)

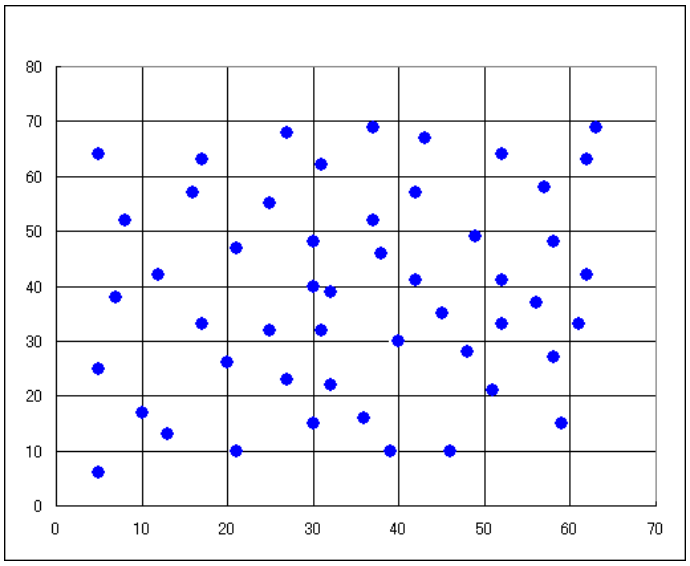

Figure 6: Instance for Part 6, Human performance on TSP. Draw your tour in this picture.

Now is about Lecture 4 and T3: The TRAVELLING-SALESMAN-PROBLEM. Research, e.g. Macgregor, J.N. & Ormerod, T. "Human performance on the traveling salesman problem", Perception & Psychophysics (1996) 58: 527, shows that human, e.g. you and Steven 10 years ago when he experimented

Page 4 Marks = ___

on TSP, is actually rather good at solving TSP instances. Given a 51 points test case in Figure 6 below, try to connect them with 51 edges to produce a **valid** TSP tour (validity is a hard constraint). Grading scheme will be determined *after* seeing your answers.

A.7 Part 7 (7 marks)

Finally we use manual question to verify your understanding of Lecture 5+6: Run FORD-FULKERSON's algorithm that uses the *capacity-scaling* heuristic with $O(m^2 \times \sqrt{U})$ time complexity (this was what we discussed in more details: Faster implementation that runs in $O(m^2 \log(U))$ exists). First, let's determine U, the weight of the outgoing edge from source vertex s with the max capacity. In Figure 7, left, U is: _____ and $k = \sqrt{U}$ is _____. Now, divide all edge capacities by k, write down the new edge capacities in the spaces provided in Figure 7, right, and then run the basic FORD-FULKERSON's algorithm on it to get $f =$ _____ and $k \times f =$ _____ (this value may not be the max flow).

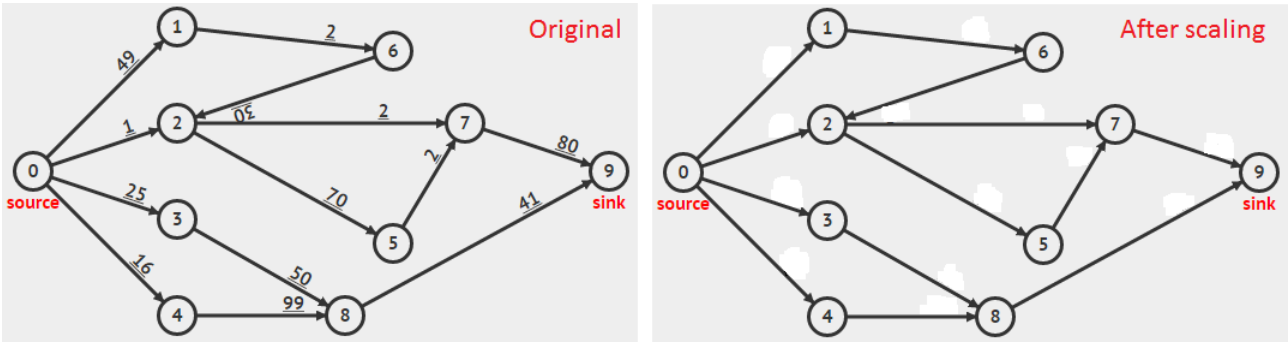

Figure 7: Instance for Part 7, determine U, choose the scaling factor $k = \sqrt{U}$.

A.8 Part 8 (5 marks)

Now, let's run the shortest-path heuristic of FORD-FULKERSON's algorithm instead, i.e. the EDMONDS-KARP's algorithm, on Figure 7, left. Please write the series of augmenting paths (APs) that will be found in the spaces below (break ties, if any, by selecting path involving lower vertex numbers first).

- AP1 = 0 → _____; Bottleneck Capacity = __; Current flow f = __.
- AP2, if any = _____; Bottleneck Capacity = __; Current flow f = __.
- AP3, if any = _____; Bottleneck Capacity = __; Current flow f = __.
- AP4, if any = _____; Bottleneck Capacity = __; Current flow f = __.
- AP5, if any = _____; Bottleneck Capacity = __; Current flow f = __.
- AP6, if any = _____; Bottleneck Capacity = __; Current flow f = __.
- AP7, if any = _____; Bottleneck Capacity = __; Current flow f = __.

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

B Decision Problems (25 Marks)

B.1 Boss Request (7 Marks)

A few years after you graduate from NUS School of Computing, you work for a certain small startup company as a software engineer. One day, your boss, a senior citizen who graduated with a non computing degree two decades earlier than you (hence he is now your boss) asks you to help him check among his $n = 50$ employees, what is the *largest subset* of these n employees who share *the same birthday week*. He said that he wants to select that week for the next performance bonus giveaway week. Your answer is (select *only one* and elaborate):

1. "Boss, this problem is an NP-hard optimization problem by reduction of another NP-hard problem known as _____, so (elaborate your defense in the box below)."
2. "Boss, this problem is an easy optimization problem, I will solve it by
(write your solution in the box below).

B.2 Theory versus Practice? (9 Marks)

We were told in Lecture 3b that the Metric-Steiner-Tree can be approximated by running the Min-Spanning-Tree algorithm only on the set of required vertices R. The rather long analysis of this approximation algorithm during Lecture 3b yields approximation ratio of 2, i.e. MST on set R is a 2-approximation algorithm for Metric-Steiner-Tree. Then in PS2 - Rubble (Version V), Steven asked you to empirically test if this bound is tight or loose... Now based on what you have learned and experiment so far, you say (select *only one* and elaborate):

1. "Steven, your 2-approximation analysis is actually not tight, a tighter approximation ratio should be less than 2, i.e. 1._____. We can prove that the approximation algorithm: MST on set R has that tighter approximation ratio by (write a sketch of your stronger proof in the box below)"
2. "Steven, the approximation ratio of 2 for this MST-based approximation algorithm for Metric-Steiner-Tree is actually already tight. Because: (write your answer in the box below).

B.3 Non-Integer Flows (9 Marks)

We were told in Lecture 6 that the $O(m^2 \times U)$ time complexity and termination of the *basic* FORD-FULKERSON'S algorithm (without any heuristic applied) relies on the assumption that all edge capacities are integers. But what if we are given a flow network $GR_{eal} = (V, E)$ where all the edge capacities in GR_{eal} are floating point numbers but *the capacities are all rational*, e.g. 1/16, 2/5, 7/14. Please decide whether the basic FORD-FULKERSON'S algorithm can terminate on GR_{eal}?

Section B Marks = ___ + ___ + ___ = ___

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

C.1 Max-Clique on VisuAlgo (15 Marks)

The year is 2020. Steven has added a lot of features in VisuAlgo and he has started the foray towards visualization of various NP-hard optimization problems (assume that nobody has proven whether $P = NP$ (or not) by then). You are one of his Final Year Project (FYP) student who is assigned to do Max-Clique visualization. If you forget, the decision version of Clique is defined as follows: "Given a graph $G = (V, E)$ and an integer k, is there a subset $C \subseteq V$ of size k such that C is a clique in G? The Max-Clique optimization variant seeks for the subset C with the largest possible k in G. Here are Steven's requirements:

1. VisuAlgo user *must* be able to draw his/her own graph input, as per VisuAlgo standard, so you **can/cannot (circle one)** pre-compute the answers on pre-scripted example graphs only.
2. Steven dislikes graphs that are drawn with edge crossings, so he has put on a check in his "Draw Graph" feature so that only graphs without edge crossing are allowed in VisuAlgo.
3. VisuAlgo must be very responsive, i.e. upon clicking "Show Max-Clique" button, a Java?Script (the year is 2020, remember) routine has to quickly compute the answer in not more than 1 second. Longer than that, Steven says that the visitors will press Alt+F4 instead.

What is the *best* algorithm that will you implement as Java?Script routine of your Max-Clique visualization in VisuAlgo? What is the time complexity? Is your algorithm seeks for an approximate solution or an optimal solution? What are the rough limit of n and m, the number of vertices and edges, that you will set in your visualization?

C.2 Min-Size Min-Cut (15 Marks)

We were told in Lecture 5 that by running a the Max-Flow algorithm on the flow graph (with integer capacities), we also simultaneously find the Min-Cut of the flow graph by running graph traversal (either DFS/BFS) from source vertex s on the final residual graph. Vertices that are still reachable from s are in the S-component and the other vertices are in the T-component. Edge(s) that cross(es) S-component to T-component is/are the edge(s) that form the minimum cut. However, this technique may not yield the minimum cut *of minimum-size*. For example, Figure 8 shows an example where the max flow = min cut value is $f = 2$ which can be found by either deleting two edges $1 \rightarrow 3$ with capacity 1 and $0 \rightarrow 2$ also with capacity 1. The minimum-size min-cut for this example should be deleting *just one* edge $3 \rightarrow 4$ with capacity 2.

If you find that this variant is in P, design an algorithm using any related? techniques that you have learned so far in this class (or beyond?) to solve this variation of the Min-Cut problem that we learned in Lecture 5 and *prove its correctness*.

However, if you find that this variant is NP-hard, prove that it is and then write a simple brute force algorithm to solve this problem when $m \leq 20$, i.e. the number of edges $m = |\ E\ |$ in the flow graph is small.

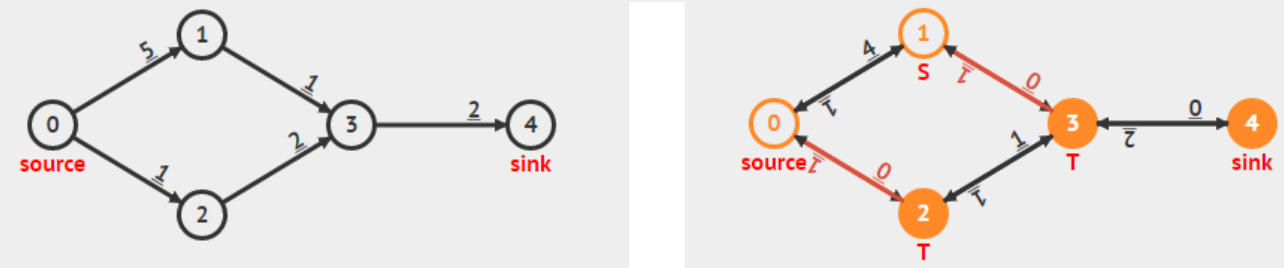

Figure 8: The Min-Cut of the flow graph G on the left is usually like the one shown on the right. But how to find the Min-Size Min-Cut instead?

Section C Marks = ___ + ___ = ___

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