

National University of Singapore
School of Computing
CS4234 - Optimisation Algorithms
(Semester 1: AY2023/24)

Date and Time: Thursday, 12 October 2023, 12.02-13.32 (90m)

INSTRUCTIONS TO CANDIDATES:

1. You can start immediately after you are given the password to open this file.
2. This assessment paper contains FOUR (4) sections.
It comprises TWELVE (12) printed pages, including this page.
3. This is an **Open Book Assessment**.
Additionally, you can also use your laptop (but in airplane/no Internet mode).
4. Answer **ALL** questions within the **boxed space** of the answer sheet (page 7-12).
There are a few starred (*) boxes: free 1 mark if left blank but 0 for wrong answer (no partial).
You will only need to hand over page 7-12 after this quiz.
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 :)

A The Simpler Questions (25 marks)

A.1 MVC (5 marks)

What's wrong with the left side of Figure 1 so that Prof Halim has to add edge (1, 5)?

Context: This slide introduces the MVC problem.

The intention is to show that MVC is NP-hard.

To solve the instance shown optimally, one needs to run a $O(2^{|V|} \times |E|)$ complete search algorithm.

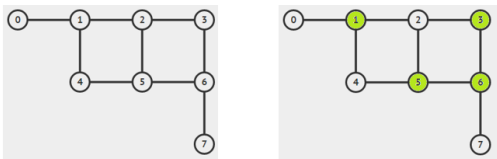
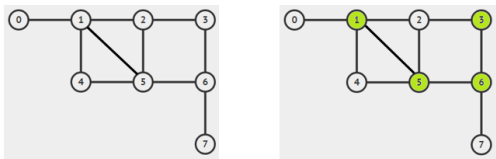
Min-Vertex-Cover (MVC)	Min-Vertex-Cover (MVC)
<p>Definition:</p> <ul style="list-style-type: none"> Given a graph $G = (V, E)$, find a minimum-sized set S that is a vertex cover for G <p>Analogy: A certain coffee brand in Singapore</p> <p>Try Brute Force ("Complete Search") live at https://visualgo.net/en/mvc</p> 	<p>Definition:</p> <ul style="list-style-type: none"> Given a graph $G = (V, E)$, find a minimum-sized set S that is a vertex cover for G <p>Analogy: A certain coffee brand in Singapore</p> <p>Try Brute Force ("Complete Search") live at https://visualgo.net/en/mvc</p> 

Figure 1: Left: The opening slide about MVC before AY 2021/22; Right: The same slide afterwards

A.2 ILP (5 marks)

Express the NP-hard optimization problem: MAX-CLIQUE problem that we discussed in Lecture 01 and Tutorial 01 as an ILP!

To avoid ambiguity, a *clique* in a graph $G = (V, E)$ is a subset $C \subseteq V$ such that the subgraph of G induced by those vertices is a complete graph. The decision version of CLIQUE is as follows: "Given a graph $G = (V, E)$ and an integer k , is there clique of size k (vertices)?" The optimization version MAX-CLIQUE asks for the largest possible clique in G .

Hint: Think about the complement graph $\bar{G} = (V, \bar{E})$.

A.3 ILP and Relaxed ILP (10 marks)

A wood factory has many 6-meters boards (for this problem, assume the factory has sufficiently many such 6-meters boards). Suppose a customer needs 6 one-meter boards, 7 two-meters boards, and 17 three-meters boards. The factory can produce the required boards by cutting up their default 6-meters boards in various ways as shown in Table 1, e.g., combination 5: we can cut one 6-meters board into 3 one-meter boards and 1 three-meters board.

Combination ID	Number of 1-meter board(s)	Number of 2-meters board(s)	Number of 3-meters board(s)
1	0	0	2
2	0	3	0
3	1	X	1
4	2	2	Y
5	3	0	1
6	4	1	0
7	Z	0	0

Table 1: All Combinations Without Wastage

How should the factory cut their 6-meter boards so that it can satisfy all the customer's demand *while minimizing the number of 6-meter boards used*. There are three sub-parts:

- For 3 marks, what are the values of X , Y , Z in Table 1?
- For 5 marks, write an Integer Linear Program (ILP) for this optimization problem.
- For the last 2 marks, relax this ILP into an LP by ignoring the integer constraints, run any LP solver that you have in your laptop (Excel, `lp_solve`, or Simplex code in C++/Java/Python). Then apply rounding technique (if necessary) to find the minimum number of 6-meter boards used by the factory to satisfy the customer's demand, i.e., the value of **OPT**.

Marking scheme: **2 marks for OPT**, **1 mark for OPT+1**, **0 mark otherwise**.

A.4 Christofides's Implementation (5 marks)

The hardest part for correctly implementing the Christofides's 1.5-approximation algorithm for (M-NR) TSP of a complete weighted graph G is on how to get a perfect matching on a general (complete) weighted graph (on subgraph of G induced by the set of vertices of MST of G that has odd degree). We can theoretically use the weighted form of Edmonds' matching (Blossom) algorithm, but that algorithm is still out of scope for this course (and for this midterm).

One student suggests that we can restrict $V \leq 20$ so that we can use $O(2^V \times V)$ Dynamic Programming (DP) with bitmask to get the perfect matching. What is now the issue of this suggestion?

B Inheritance (25 Marks)

There are two siblings, let's call them Steven and Felix, who want to **equally** divide their INHERITANCE (so as to avoid civil war). For the following three scenarios, design *the best* algorithm to help the two siblings on whether they can do this equal division. You have to analyze the time complexity of your algorithm, be it polynomial or exponential algorithm. For all three scenarios, you do not need to provide proof of correctness.

Background: The PARTITION problem is the task of deciding whether a given multiset S of n positive integers can be partitioned into two subsets S_1 and S_2 such that the sum of the integers in S_1 equals the sum of the numbers in S_2 . The general version PARTITION of is proven to be NP-complete.

B.1 Scenario 1 - Floating Point (6 marks)

Their inheritance consists of n cheques that can be cashed by *either* of them. Each cheque has a positive *floating-point* dollar values (correct to two digits after decimal point, to represent cents). It is guaranteed that no cheque value is greater than one hundred SGD.

B.2 Scenario 2 - Two Denominations (6 marks)

Their inheritance consists of n coins, but curiously only in two different denominations: x dollars and y dollars. Both are integers up to $2^{64} - 1$.

B.3 Scenario 3 - Powers of Two (7 marks)

Their inheritance consists of n (bit)coins, with arbitrary number of different denominations, but curiously each denomination is a non-negative powers of two, i.e., $2^0 = 1$ dollar, $2^1 = 2$ dollars, ..., $2^7 = 128$ dollars, etc up to 2^{63} dollars.

B.4 Six Sample Test Cases ($6 \times 1 = 6$ marks)

There are six test cases, two for each scenario above.

For each test case, output 'NO' if equal division is impossible, or output 'YES' and show a certificate (i.e., the portion of one of the sibling) otherwise.

1-1. $n = 7$, $S = \{10.01, 10.70, 10.04, 25.47, 25.30, 10.01, 10.01\}$.

1-2. $n = 7$, $S = \{10.01, 10.63, 10.04, 25.47, 25.30, 10.01, 10.01\}$.

2-1. $n = 11$, $S = \{7, 4, 4, 4, 4, 7, 4, 4, 7, 7, 7\}$.

2-2. $n = 11$, $S = \{7, 4, 4, 4, 4, 7, 4, 4, 7, 7, 4\}$.

3-1. $n = 5$, $S = \{32, 64, 32, 4, 8\}$.

3-2. $n = 5$, $S = \{32, 64, 32, 8, 8\}$.

C Min-Cardinality-Min-Cut (25 Marks)

In Max Flow Lecture on Week 05, we have proved that by running a MAX-FLOW algorithm on the flow graph G (with Integer capacities) from the source vertex s to the sink vertex t , we find the MAX-FLOW and also simultaneously the MIN-CUT of the flow graph. We can construct the MIN-CUT by running a graph traversal algorithm (DFS/BFS) from 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 (including t). Edge(s) that cross(es) S -component to T -component is/are the edge(s) that form the MIN-CUT.

However, this technique may not yield the minimum cut of *minimum cardinality*. For example, Figure 2 shows an example where the max flow = min cut value is $f = 2$ which can be found by cutting two edges: $0 \rightarrow 2$ with capacity 1 and $1 \rightarrow 3$ also with capacity 1 (two cyan dashed lines). However, the MIN-CARDINALITY-MIN-CUT for this example should be cutting *just one* edge $3 \rightarrow 4$ with capacity 2 (one green dotted line).

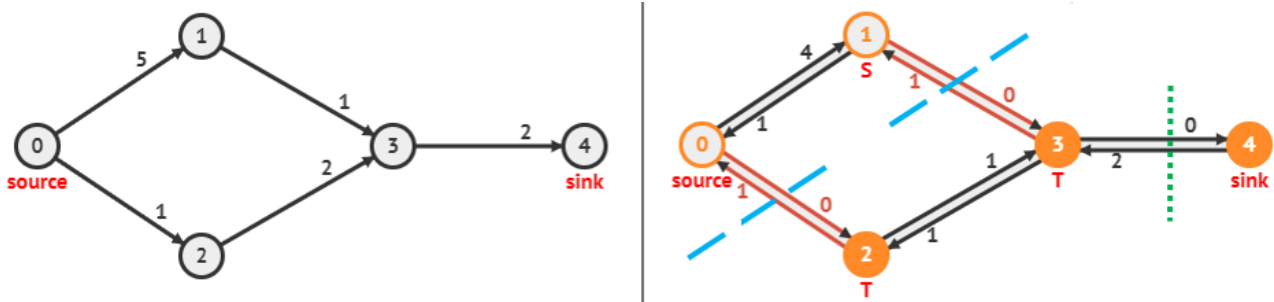


Figure 2: The MIN-CUT of the flow graph G on the left is usually like two cyan dashed lines shown on the right. But how to find the MIN-CARDINALITY-MIN-CUT instead one green dotted line?

C.1 Two Sample Test Cases (2 + 2 = 4 Marks)

If you have understood this MIN-CARDINALITY-MIN-CUT problem, highlight the edge(s) that has/have to be cut in Figure 3 and Figure 4 in the answer sheet. If there are multiple possible valid answers, you can select any (but *preferably* pick the lexicographically smallest answer).

C.2 Solve Unique-Min-Cut (10* Marks)

The full version of MIN-CARDINALITY-MIN-CUT may still be in P or actually really NP-hard, see Section C.3. So, solve a simpler variant UNIQUE-MIN-CUT for 10 marks. Decide if a given flow graph G has a *unique* min cut (and thus also the one with the minimum cardinality) or not (i.e., there are 2 (or more) possible st-cuts with minimum capacity). There are a few P solutions for this simpler variant (you can answer any). You are not required to provide the proof of correctness.

Marking scheme: 1 if left blank, 0 if solution, 10 if correct.

C.3 Solve Min-Cardinality-Min-Cut (11* Marks)

Choose ONE:

1. If you find that MIN-CARDINALITY-MIN-CUT is in P, design an algorithm using any techniques that you have learned so far in this class (or beyond?) to *optimally* solve this MIN-CUT problem variant in *polynomial time*. This time, you also have to *prove its correctness*.
2. If you find that this variant is NP-hard, prove that it is so, and then write a simple complete search algorithm to solve this problem when $m \leq 20$, i.e., the number of edges $m = |E|$ in the flow graph is small.

Marking scheme: 1 if left blank, 0 if very wrong, 11 if correct,

For this subsection, partial 5 marks may still be given for a few possible partial solutions.

D Randomized Greedy Pre-Processing for MC(B)M (25 marks)

D.1 Maximal versus Maximum Matching (2 + 2 = 4 Marks)

At the beginning of Graph Matching lecture on Week 06, we have two similar but different terms. Given a graph $G = (V, E)$, a maximal matching M of graph G is “a matching that is not a subset of any other matching” and a maximum (cardinality) matching is “a matching that contains the largest possible number of edges”.

Given two copies of the same bipartite graph G shown in Figure 5 (see the answer sheet), circle *any* maximal matching in G that *is not maximum* on the left figure, and circle *any* maximum matching on the right figure (but *preferably* pick the lexicographically smallest answer).

D.2 2-Approximation (7* marks)

Prove that any *maximal* matching is a 2-approximation of a *maximum* matching.

Note that the underlying graph does not necessarily has to be bipartite for this proof.

Marking scheme: **1 if left blank, 0 for wrong proof, 7 for correct proof.**

D.3 RGPP for MCBM Produces a Maximal Matching (7* marks)

The simple Augmenting Path algorithm that directly implements Berge’s theorem shown on Week 06 has time complexity of $O(V \times E)$ or $O(V^3)$ if $E = O(V^2)$ in a (near-)Complete Bipartite Graph – far slower than the better Hopcroft-Karp/Dinic’s algorithms that run in $O(V^{2.5})$.

A potential improvement is to run the $O(V + E)$ “Randomized Greedy Pre-Processing” (abbreviated as RGPP): For every vertex on the left set (from the lowest vertex number to the highest), match it with a *randomly chosen* unmatched neighbouring vertex on the right set. This way, we eliminate many trivial (one-edge) Augmenting Paths that consist of a free vertex u , an unmatched edge (u, v) , and a free vertex v . Thus the Augmenting Path *Plus* algorithm just need to execute Berge’s theorem k times, for $O(k \times E)$ time overall.

Prove that RGPP produces a *maximal* matching.

Marking scheme: **1 if left blank, 0 for wrong proof, 7 for correct proof.**

D.4 Implication (3 marks)

As any *maximal* matching is a 2-approximation of a *maximum* matching (from Subsection D.2) and the output of RGPP on a Bipartite Graph is a *maximal* matching (from Subsection D.3), what is the implication of the maximum k of the $O(k \times E)$ Augmenting Path *Plus* algorithm?

D.5 Omitting the Randomization of RGPP (4 marks)

Suppose the Randomized part of the RGPP is omitted, i.e., the algorithm becomes GPP: For every vertex on the left set, match it with an unmatched neighbouring vertex on the right set *with the largest vertex number*. Be an adversary and draw any Bipartite Graph G with carefully numbered vertices so that GPP’s output of G is *exactly half* of the MCBM of G .

The Answer Sheet

Write your Student Number in the box below using **(2B) pencil**:

STUDENT NUMBER											
A											
U	<input type="radio"/>	0	0	0	0	0	0	0	0	A	U
A	<input checked="" type="radio"/>	1	1	1	1	1	1	1	1	B	R
HT	<input type="radio"/>	2	2	2	2	2	2	2	2	E	U
NT	<input type="radio"/>	3	3	3	3	3	3	3	3	H	U
		4	4	4	4	4	4	4	4	J	X
		5	5	5	5	5	5	5	5	L	Y
		6	6	6	6	6	6	6	6	M	
		7	7	7	7	7	7	7	7		
		8	8	8	8	8	8	8	8		
		9	9	9	9	9	9	9	9		

Box A.1. What's wrong with the left side of Figure 1?

Box A.2. Express MAX-CLIQUE problem as an ILP!

Box A.3.1. What are the values of X , Y , Z in Table 1?

Box A.3.2. Express this problem as an ILP!

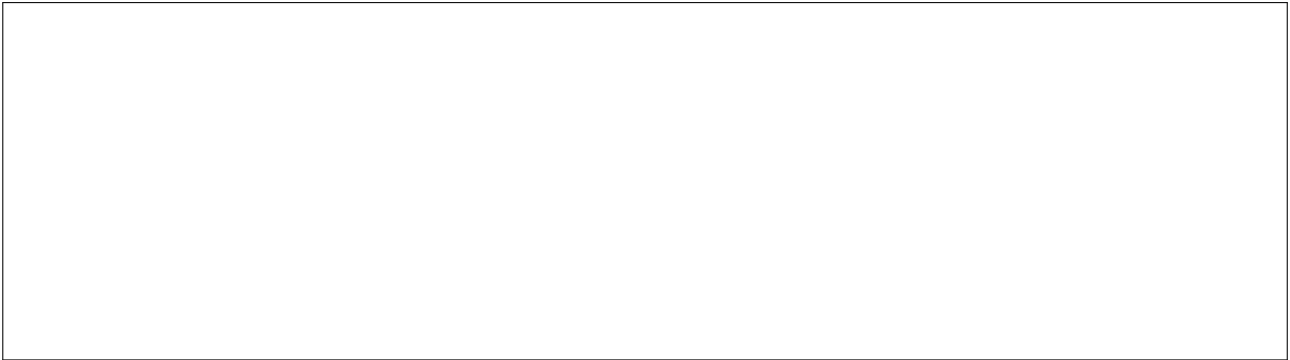
Box A.3.3. What is the minimum number of 6-meter boards used?

Box A.4. What is now the issue of this suggestion?

Box B.1. Scenario 1 - Floating Point

Box B.2. Scenario 2 - Two Denominations

Box B.3. Scenario 3 - Powers of Two



Box B.4. Six Sample Test Cases. Circle 'NO' or 'YES' accordingly (provide certificate for 'YES').

- 1-1. 'NO' or 'YES' (and the certificate: Steven = {.....})
- 1-2. 'NO' or 'YES' (and the certificate: Steven = {.....})
- 2-1. 'NO' or 'YES' (and the certificate: Steven = {.....})
- 2-2. 'NO' or 'YES' (and the certificate: Steven = {.....})
- 3-1. 'NO' or 'YES' (and the certificate: Steven = {.....})
- 3-2. 'NO' or 'YES' (and the certificate: Steven = {.....})

Box C.1. Draw (dotted) line(s) directly in Figure 3 and Figure 4.

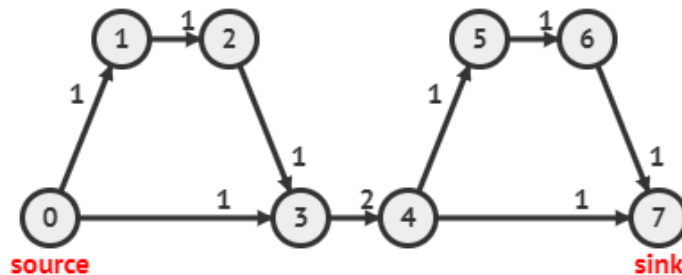


Figure 3: Find the MIN-CARDINALITY-MIN-CUT of this flow graph 1, $s = 0, t = 7$.

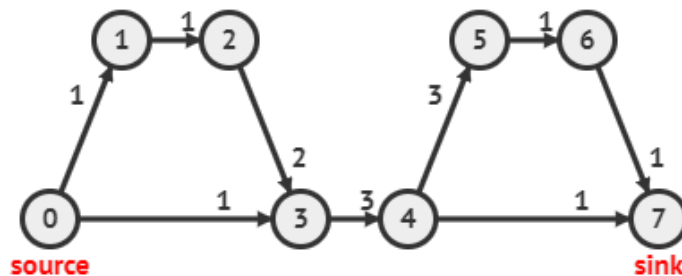


Figure 4: Find the MIN-CARDINALITY-MIN-CUT of this flow graph 2, $s = 0, t = 7$.

Box C.2.* Solve UNIQUE-MIN-CUT.

Box C.3.* Solve MIN-CARDINALITY-MIN-CUT.

Box D.1. Circle your answer directly in Figure 5.

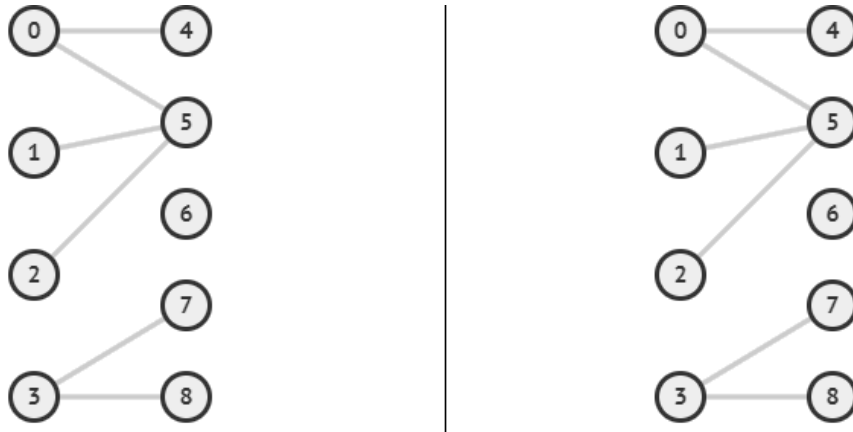


Figure 5: Circle Maximal (but not Maximum) Matching (Left) and Maximum Matching (Right)

Box D.2.* Maximal is a 2-Approximation of Maximum Matching

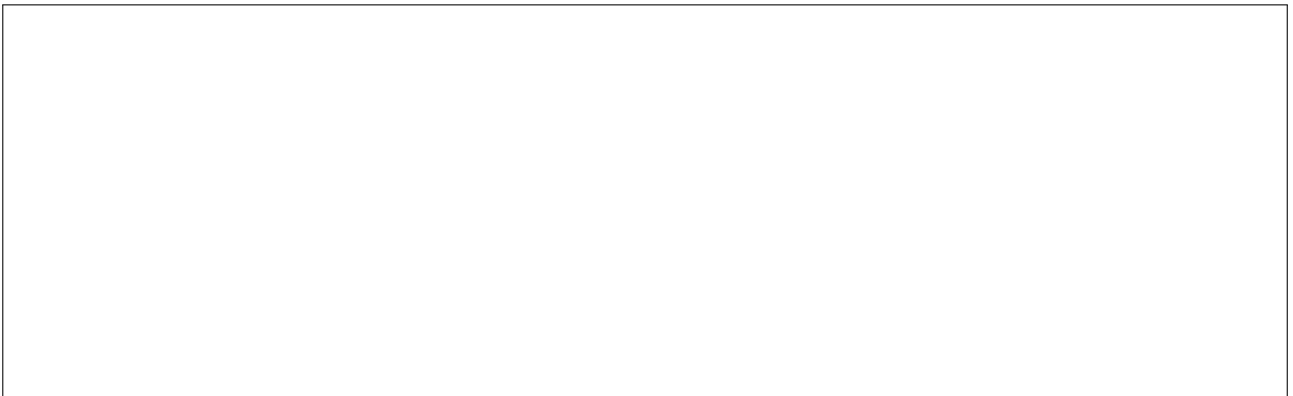
Box D.3.* RGPP for MCBM Produces a Maximal Matching



Box D.4. Implication



Box D.5. Omitting the Randomization of RGPP



End of this Paper, All the Best