National University of Singapore

School of Computing

CS4234 - Optimisation Algorithms

(Semester 1: AY2022/23)

Tuesday, 29 November 2022, 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 FOUR (4) sections.
   It comprises FOURTEEN (14) printed pages, including this page.
3. This is an **Open Book + Open Laptop (in Airplane Mode) Assessment**.
   You can use your Laptop *offline* as you see fit (to read notes, use Excel, write/run code, etc).
4. For Section A, use the OCR form provided (use 2B pencil).
   For Section B, C, and D, answer **ALL** questions within the **boxed space** in the answer sheet.
   The answer sheet is at page 11-14 but you will still need to hand over the entire paper.
   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 Kuhn-Munkres on graph G, Edmonds' Matching on graph G', etc.
7. Please write your Student Number only. Do **not** write your name.

A	0							

This portion is for examiner's use only

Section	Maximum Marks	Your Marks	Remarks
A	40		
B	22		
C	16		
D	22		
Total	100		

A MCQs ($20 \times 2 = 40$ marks)

Select the **best unique** answer for each question.
Each correct answer worth 2 marks.

The MCQ section will not be archived to open up possibilities of reuse in the future.

[PAGE 3 IS NOT ARCHIVED]

[PAGE 4 IS NOT ARCHIVED]

[PAGE 5 IS NOT ARCHIVED]

[PAGE 6 IS NOT ARCHIVED]

[The upper side of PAGE 7 IS NOT ARCHIVED]

B `Unbounded-Knapsack-Problem` (22 marks)

The `Unbounded-Knapsack-Problem` (`UKP`) is a variation of the classic 0/1-`Knapsack` problem that is formally defined as follows: There are n items with weights w_1, w_2, ..., w_n and values v_1, v_2, ..., v_n (all positive integers), and the goal is to pick the most valuable combination of these n items subject to the constraint that their total weight is at most S (you can assume that $w_i \leq S, \forall i \in [1..n]$, i.e., no single item is heavier than S). In `UKP`, there is no upper bound on the number of copies of each kind of item, i.e., you can pick the same item more than once (unlike the classic 0/1 variant).

For example, if $n = 4$, $w = \{10, 4, 6, 12\}$, $v = \{100, 70, 50, 10\}$, and $S = 12$, then the most valuable combination is to pick item 2 three times with total weight of $(3 \times 4 = 12) \leq (S = 12)$ and with the highest total value of $3 \times 70 = 210$. Contrast this with the classic 0/1 variant where the optimal combination is to pick item 2 and 3 (both can only be picked once) with total weight of $(4 + 6 = 10) \leq (S = 12)$ and with the highest total value of $70 + 50 = 120$.

B.1 Three Manual Test Cases ($3 \times 1 = 3$ marks)

What are the answers (the value of the most valuable combination — remember that you can pick any item more than once) if you are given:

1. $n = 4$, $w = \{6, 3, 4, 2\}$, $v = \{30, 14, 16, 9\}$, and $S = 10$.
2. $n = 8$, $w = \{1, 2, 4, 8, 16, 32, 64, 128\}$, $v = \{1, 3, 1, 1, 200, 1, 1000, 1\}$, and $S = 243$
3. $n = 7$, $w = \{2387, 8787, 4323, 7879, 1298, 877, 1232\}$, $v = \{1076, 1392, 1750, 3088, 5109, 4298, 8956\}$, and $S = 21\,823$.

B.2 Formulate as ILP (3 marks)

Formulate `UKP` as an ILP. You can use pseudocode.

B.3 Dynamic Programming (DP) Solution (4 marks)

`UKP` also has DP solution that can optimally solve all instances of this problem. Show this DP solution and analyze its *pseudo-polynomial* time complexity in terms of n and S.

B.4 What Is The Issue? (1 mark)

What is the is main issue with the pseudo-polynomial solution in Section B.3 above?

B.5 An Approximation Algorithm (4 marks)

Long ago (in 1957), George Dantzig proposed the following approximation algorithm: Sort the n items by non-increasing value-to-weight ratio. Insert items into Knapsack following this sorted ratio, i.e., repeatedly take the item with the highest value-to-weight ratio until we cannot take that item anymore, then we do the same with the next item with the second highest value-to-weight ratio, and so on until the Knapsack is full or we have considered all items. We are not going to prove it in this question, but this algorithm is guaranteed to find a solution that is no worse than half of the optimal maximum value, i.e., a 2-approx algorithm.

Now compare this approximation algorithm versus the DP solution in Section B.3 in terms of speed (time complexity), universality (does the approximation algorithm work on all `UKP` instances?), and optimality (show at least one small test case where this approximation algorithm does not able to find the optimal solution).

B.6 Local Search for `UKP` (7 marks)

Design a (Stochastic) Local Search for the `UKP` that is at least 2-approx or better. Describe the rough ideas of your algorithm. Your answer will be graded based on how sound the heuristics/local search ideas that you propose, albeit we cannot test your ideas empirically in this theory paper.

C Duìlián/Couplet (Chinese Poetry) (16 marks)

In Chinese poetry, a duìlián/couplet is a pair of lines of poetry usually seen on the sides of doors leading to people's homes. Usually when the Chinese New Year arrives, people write new couplet and paste them on the sides of their doors. Generally, a couplet is required to obey the following rules:

- A couplet has exactly two lines (to be placed on the left and right doors, respectively).
- Both lines have exactly the same number of characters.
- One character in the left line must be suitably paired to its corresponding character in the right line, in the sense of the lexical category, tunes, and meanings.

Writing a good couplet is a challenging job, so you need help. First, out of n different characters from the Chinese dictionary, you have asked your father, a Chinese language expert, to figure out m suitable Chinese character pairs out of these n different characters. There is no implicit transitivity, e.g. character 'Moon' (in Chinese, this is really just one single character) is suitable to be paired with either 'Sun' or 'Flower', but 'Sun' is not suitable to be paired with 'Flower'.

Now you want to pick as many character pairs as possible to be considered in your new couplet, subject to one constraint: No character is allowed to belong to two different pairs that are picked.

For example, given $n = 6$ characters {'Moon', 'Sun', 'Flower', 'Wind', 'Star', 'Mountain'} and $m = 5$ suitable pairs: ('Moon', 'Sun'), ('Moon', 'Flower'), ('Moon', 'Wind'), ('Flower', 'Wind'), ('Sun', 'Star'), you can pick at most 2 suitable pairs, e.g., ('Moon', 'Sun') and ('Flower', 'Wind').

You just need to output a single integer as your answer: the maximum number of suitable pairs that you can pick.

C.1 Three Other Solutions (2 marks)

If you have understood this question, give three other solutions for the same example test case that are different from the solution shown in Section C above: ('Moon', 'Sun') and ('Flower', 'Wind'). These three other solutions still contain at most 2 suitable pairs.

C.2 Do you know the actual problem name? (1 mark)

If you know the real name of this problem, mention it.

C.3 Is this an NP-hard optimization problem? (1 mark)

If you say it is, give a short justification, and you are allowed to use an exponential algorithm for Subsection C.4. If you say it is not, you must use a polynomial solution for Subsection C.4.

C.4 Solve This Problem (12 marks)

One of the route in Subsection C.3 is correct.
The correct route worth the full (12) marks.
The other one worth only 6 marks (2-opt of the optimal marks).

D `Traveling-Salesperson-Problem` Variant (22 marks)

Being a Traveling Salesperson[1] is tough work. You are one such salesperson and would like to find an efficient way to visit all the cities in a foreign country *exactly once*. You define efficiency in a weird way as you are actually afraid of flying on planes. Your favorite mode of transportation is trains and you will gladly take trains for as long as possible.

However, the train system in this foreign country is very weird as the train lines are all *one-way*, and once anyone takes a train out of a city, there is no sequence of train lines that return to that city[2]. Fortunately in this country, every city has exactly one airport, so you can travel by plane (that you dread) from any city to any other city.

Now you want to know the *minimum* number of flight(s) that you need to complete your trip. Note that you can start your sales trip from any city.

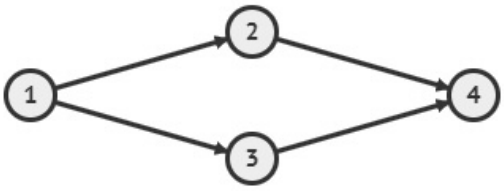

Figure 1: An Example With 4 Cities

Consider this foreign country with four cities with the directed arrows representing one-way train routes (see Figure 1). There are several possible trips that you could take, but you will need to fly *at least once*. Here are some (but not all) possible routes with fewest (one for this example) flight(s), with $\rightarrow$ indicating a train trip and $\Rightarrow$ indicating a flight:

- $1 \rightarrow 2 \rightarrow 4 \Rightarrow 3$ (start from 1, train $1 \rightarrow 2 \rightarrow 4$, then fly once $4 \Rightarrow 3$, done)
- $2 \rightarrow 4 \Rightarrow 1 \rightarrow 3$ (start from 2, train $2 \rightarrow 4$, fly once $4 \Rightarrow 1$, then train again $1 \rightarrow 3$, done)

You will be given two space-separated integers n and m, where n is the number of cities and m is the number of train lines. The cities are numbered from 1 to n. Then, you will be given m more pairs of integers u and v ($1 \leq u, v \leq n; u \neq v$), which indicates that there is a train line from city u to city v (one direction). All train lines will be distinct.

You just need to output a single integer as your answer: the minimum number of flights that you must take to visit all of the cities exactly once.

D.1 Two Other Routes ($2 \times 1 = 2$ marks)

If you have understood this question, give two other routes for Figure 1 that are different from the two that have been shown above ($1 \rightarrow 2 \rightarrow 4 \Rightarrow 3$ or $2 \rightarrow 4 \Rightarrow 1 \rightarrow 3$). These two other routes still need at least one flight. Please use the similar explanation format as above.

[1] Gender neutral term.

[2] If you find this very weird, just assume that trains return to their origin city *without* carrying any passenger.

D.2 Four Manual Test Cases ($4 \times 1 = 4$ marks)

What are the answers (the minimum number of flights needed) if you are given:

1. $n = 20$, $m = 0$ (i.e., no train).
2. $n = 20$, $m = 19$, and the train lines have this property: $v = u + 1$, $\forall u \in [1..19]$.
3. $n = 5$, $m = 4$, and the train lines are $\{(1 \rightarrow 3), (2 \rightarrow 3), (3 \rightarrow 4), (4 \rightarrow 5)\}$
4. $n = 10$, $m = 8$, and the train lines are $\{(2 \rightarrow 1), (3 \rightarrow 1), (4 \rightarrow 2), (5 \rightarrow 2), (6 \rightarrow 3), (7 \rightarrow 3), (8 \rightarrow 4), (9 \rightarrow 4)\}$

D.3 Do you know the actual problem name? (1 mark)

The name `Traveling-Salesperson-Problem` Variant in this section is not the real problem name. If you know the real name of this problem, mention it.

D.4 Is this an NP-hard optimization problem? (1 mark)

If you say it is, give a short justification, and you are allowed to use an exponential algorithm for Subsection D.5. If you say it is not, you must use a polynomial solution for Subsection D.5.

D.5 Solve This Problem (14 marks)

One of the route in Subsection D.4 is correct.
The correct route worth the full (14) marks.
The other one worth only 7 marks (2-opt of the optimal marks).

E Answer Sheets

Box B-1. Three manual test cases (write three different integers in B.1.1, B.1.2, and B.1.3 order)

Box B-2. ILP of `UKP`

Box B-3. DP for `UKP`

Box B-4. DP Issue

Box B-5. DP vs Approximation Algorithm

Box B-6. Local Search for `UKP`

Box C.1. Three other solutions (write three lines with similar explanation format as in the example)

Box C.2. Say the actual problem name

Box C.3. NP-hard?

Box C.4. Solve this!

Box D.1. Two other routes (write two lines with similar explanation format as in the example)

Box D.2. Four manual test cases (write four different integers in D.2.1, D.2.2, D.2.3, and D.2.4 order)

Box D.3. Say the actual problem name

Box D.4. NP-hard?

Box D.5. Solve this!

– END OF PAPER; All the Best –