National University of Singapore

School of Computing

CS4234 - Optimisation Algorithms

(Semester 1: AY2020/21)

Tuesday, 01 December 2020, AM (2 hours)

INSTRUCTIONS TO CANDIDATES:

1. Do **NOT** open this final assessment paper until you are told to do so.
2. This assessment paper contains FOUR (4) sections.
   It comprises SIX (6) printed pages, including this page.
3. This is an **Open Book/Open Laptop/Open PC Assessment**.
   But you are NOT allowed to use the Internet (web browser, messaging/cloud services, etc).
   You are free to use your Laptop/PC as you see fit *without* accessing the Internet other than for Zoom e-proctoring and to upload the (scanned) soft copy answer at the end of the paper.
4. There are 8 pages of answer sheets (**Answer Sheets.docx**, not password protected).
   If you have printer, you can print the blank answer sheets earlier.
   If you don't have printer, just mimic the format on 8 blank pages as best as you can.
   Answer **ALL** questions within the **given (boxed) space** to make grading easier.
   When you write your answers, you can do so using either pen or pencil, just write **legibly**!
   Scan the printed Answer Sheets and upload the scan to LumiNUS files at the end of the paper.
   Note that if you prefer to write your answers digitally, we can also type your answers at **Answer Sheets.docx** too. However, drawing test cases will be much more difficult in this case.
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, Kuhn-Munkres on graph G', etc.
7. All the best :)

A Short Answers (5x5 = 25 marks)

Q1). You are given a special implicit graph with K vertices labeled with [1, 2, .., K]. There is an edge between two vertices labeled with i and j if and only if $(i + j)$ is a prime. See Figure 1 for an example special implicit graph when $K = 5$.

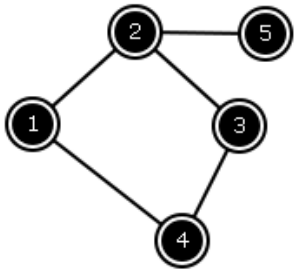

Figure 1: Implicit Graph with $K = 5$

Draw this special implicit graph when $K = 7$ (2 marks), then answer (3x1 mark = 3 marks) the `Min-Vertex-Cover` (the cardinality and the vertices in the `MVC`), the `Max-Independent-Set` (the cardinality and the vertices in the `MIS`), and the `Max-Clique` (the cardinality and the vertices in the `Max-Clique`) of this special implicit graph when $K = 7$. All valid answers are accepted.

Q2). Your company produces two products: A and B. This is the information that you know:

- Selling one unit of A/B gives you 3/5 SGD profit, respectively.
- You need to produce at least 10 As and at least 9 Bs weekly to satisfy minimum demands.
- Each A/B takes 4/6 hours to make, respectively and your only factory can obviously only run at most 24*7 = 168 hours per week.

With the information above and your current understanding of CS4234, what is your company's maximum *weekly* profit?. Marking scheme: 2 marks to express this problem as an ILP, 3 marks for the correct answer/the solution for that ILP (remember that A and B must be both integers).

Q3). Draw a small flow graph with $V = 7$ vertices, source s is vertex 0, sink t is vertex 6 such that *all* Ford-Fulkerson based methods (the DFS implementation of Ford-Fulkerson, Edmonds-Karp, or Dinic's algorithm) terminate after just **exactly one** iteration (give a short explanation), but even the relabel-to-front $O(V^3)$ Push-Relabel algorithm will spend many useless pushes or relabel operations (give a short explanation of at least 43 push or relabel operations) (mostly) to return undelivered unit of flows back to $s = 0$.

Q4). You are given a new NP-hard optimization problem (it's name is omitted, but it is real). This is the information that you know:

- $50 \leq n \leq 500$ for the (hidden) test cases that you have to solve,
- A standard Complete Search solution for this problem runs in $O(n! * n)$,
- There is no known Dynamic Programming solution,
- The run time limit per test case is approximately 10 minutes,
- It is OK to get 'good enough' results as long as the run time limit is not exceeded,
- You do not spot any special properties that can be usable,
- You are given a few sample test cases and the best known (not necessarily optimal) solutions for those sample test cases to train on, and
- Variability of Local Optimas quality that you found for each sample test case is very high.
- Fitness-Distance Correlation analysis for all sample test cases between the best known versus multiple Local Optimas that you found usually do NOT show positive correlation.

With the information above and your current understanding of CS4234, what will you do/implement to deal with this problem?

Q5). You have implemented a Tabu Search algorithm for another NP-hard optimization problem. This is the information that you know:

- The search space is all possible bit (0 or 1) strings of length n bits.
- You used the following Tabu Table mechanism: a 1D table `Tabu` that forbids the i-th bit to be toggled for `TT` (Tabu Tenure) iterations. **(initially, `Tabu[i]` is -INF** $\forall i \in [0..n-1]$**)**
- Setting an index i to be tabu can be done in $O(1)$ by storing current iteration number at `Tabu[i]`.
- Any local move that tries to toggle index i at iteration x will check against this Tabu Table in $O(1)$: is `x-Tabu[i] > TT` **(if True, then this toggle index** i **move is NOT tabu)**?

With the information above and your current understanding of Tabu Search algorithm, what do you think will happen if we set `TT` $\geq$ `n`?

B Tracing (10 marks)

Kuhn-Munkres algorithm is one of the possible algorithm that can be used to find the solution of (max) weighted `Max-Cardinality-Bipartite-Matching` (MCBM). In this tracing question, we will check your understanding about this algorithm. You are given a 7×7 matrix M that describes a (transformed into) complete Bipartite Graph $K_{7,7}$. The left set contains vertex [0/1/../6] (the rows in M) and the right set contains vertex [7/8/../13] (the columns in M). A non-negative value $M[i][j]$ signifies that if we match vertex i with vertex j, we will obtain $M[i][j]$ profit. Value $M[i][j] = -1$ signifies that there is actually no (directed) edge between vertex i to vertex j in the initial Bipartite Graph (see part 1). However, assuming that there exists edge $i \rightarrow j$ with weight -1 will not change the final answer (see part 4) for this test case.

	7	**8**	**9**	**10**	**11**	**12**	**13**
0	*-1*	8	6	*-1*	*-1*	*-1*	*-1*
1	1	4	*-1*	*-1*	*-1*	*-1*	*-1*
2	4	*-1*	1	*-1*	*-1*	*-1*	*-1*
3	*-1*	*-1*	*-1*	9	7	8	6
4	*-1*	*-1*	*-1*	7	*-1*	3	*-1*
5	*-1*	*-1*	*-1*	6	*-1*	*-1*	*-1*
6	*-1*	*-1*	*-1*	*-1*	*-1*	*-1*	0

1. (1 mark). Draw the initial Bipartite Graph where edge $i \rightarrow j$ with $M[i][j] = -1$ does not exist.
2. (1 mark). How many different perfect MCBMs are there in a complete Bipartite Graph $K_{7,7}$?
3. (3 marks). What is the maximum weight of the perfect MCBM? (if you run Kuhn-Munkres algorithm (see below) or via Complete Search (see above), this answer should be correct, i.e., OPT; otherwise if you has a bug (or you guess this answer) and your answer is X, then your score is max(0, 3 - abs(OPT-X)).
4. (5 marks). Show the workings of your Kuhn-Munkres algorithm.
   1 mark for the *initial* equality graph and the initial labels of each vertex.
   2 marks to show the progress of Kuhn-Munkres algorithm
   2 marks for the *final* equality graph with the selected perfect matching with max total profit.

C Simple Application (10 marks)

Steven needs to create n questions for this paper, one for each of the n topics (labeled topic 0 to $n-1$) of CS4234. He has m question ideas and fortunately $m \geq n$. Each question is suitable for at least one topic (a superbly set question, e.g., see section D, can be suitable for multiple topics) and has appropriateness rating of [1/2/../9] with 1 being 'least appropriate' and 9 being 'the most appropriate'. He wants to setup the paper with two constraints: each of the n topics of CS4234 need to have exactly 1 chosen question **and each question can only be assigned to exactly** 1 **topic** (output "CREATE AT LEAST X NEW QUESTION(S)" where X is the *minimal* number of new question(s) that Steven has to create to satisfy the first constraint) and if the first constraint is satisfiable, choose the setup with the highest total appropriateness rating.

For example, if Steven has $m = 3$ question ideas for $n = 2$ topics (question idea A and B have appropriateness rating 7 and 6, respectively; both A and B are only suitable for topic 0; question idea C only has appropriateness rating 1 but is the only one suitable for topic 1), then he shall pick the question idea A with appropriateness rating 7 for topic 0 and has no choice to pick the question idea C with appropriateness rating 1 for topic 1, thus total = 7+1 = 8 appropriateness rating. Note that if Steven doesn't have question idea C for the earlier example, then the output is "CREATE AT LEAST 1 NEW QUESTION(S)" (that is, create 1 new question for the missing topic 1).

Solve the general version of this problem for $1 \leq n \leq m \leq 200$. Describe the fastest possible algorithm (you can use pseudocode) and analyze its type complexity.

D `Max-Prime-Sum-Matching` (55 marks)

In the problem of `Max-Prime-Sum-Matching`, there are K positive integers, $N_1, N_2, ..., N_K$. For each integer N_i, there are C_i copies of the integer. The input must satisfy the following limits:

- $1 \leq N_1 < N_2 < ... < N_K \leq K^2$
- $1 \leq C_i \leq K^2$ for all i from 1 to K

You can match two integers N_i and N_j together if their sum is a prime number, but each copy can only be matched at most once. The problem of `Max-Prime-Sum-Matching` is to find the maximum number of matches you can make.

For example, for the instance $N = [2, 3, 4, 5], C = [2, 3, 2, 4]$ the maximum number of matches is 4 and one possible set of matches is $\{(2, 3), (3, 4), (3, 4), (2, 5)\}$.

Answer the following questions about `Max-Prime-Sum-Matching`. For all the questions, you may assume that there is a boolean function $isPrimeSum(i, j)$ that takes in 2 integers $1 \leq i, j \leq K$ and tells you whether $N_i + N_j$ is prime in $O(1)$ time.

1 Solve these instances of `Max-Prime-Sum-Matching`. For each instance, state the maximum number of matches and give a possible set of matches. (1+2+2+2 = 7 marks)

   (a) $N = [1, 2, 3, 4, 5], C = [1, 1, 1, 1, 1]$, i.e., see Q1 of Section A

   (b) $N = [1, 2, 3, 4, 5, 6, 7], C = [2, 1, 1, 1, 1, 1, 1]$

   (c) $N = [2, 3, 4, 5, 10, 11], C = [3, 3, 1, 1, 1, 1]$

   (d) $N = [1, 3, 4, 8, 17, 31], C = [3, 2, 2, 1, 4, 5]$

2 Express `Max-Prime-Sum-Matching` as an ILP. (5 marks)

3 Design a 2-approximation algorithm for `Max-Prime-Sum-Matching` that has a time complexity of $O(K^2)$. Prove that your algorithm gives a 2-approximation. (8 marks)

4 Consider a special case of `Max-Prime-Sum-Matching` called `Max-Prime-Sum-Distinct-Matching` where only 2 **different** integers with a prime sum can be matched (i.e. you cannot match 2 copies of the same integer). The input format is the same for both problems.

   (a) Given an instance of `Max-Prime-Sum-Distinct-Matching`, we can construct a matching graph G where each copy of an integer is a vertex and 2 vertices are connected by an edge if and only if they can matched. Explain why G is **bipartite**. (2 marks)

   (b) The worst-case time complexity of running Hopcroft-Karp algorithm on G is $O(K^{n_1})$. Find n_1. (2 marks)

   (c) However, if $C_i = 1$ for all i from 1 to K, then the worst-case time complexity of running Hopcroft-Karp algorithm on G is a lower $O(K^{n_2})$. Find n_2. (2 marks)

(d) Given an instance of `Max-Prime-Sum-Distinct-Matching`, explain how to construct a network flow graph H with $O(K)$ vertices such that the Max Flow of H is the same as the answer to `Max-Prime-Sum-Distinct-Matching` on the instance. (4 marks)

(e) Which Max Flow algorithm should you use to find the Max Flow of H in $O(K^3)$ time? (1 mark)

(f) After running Max Flow, how can you obtain the number of unmatched copies of N_i from the residual graph H'? (2 marks)

5 Define the $O(K^3)$ time algorithm used to solve `Max-Prime-Sum-Distinct-Matching` in the previous part as algorithm X (i.e., generating flow graph H and then running Max Flow).

Mr. Panda proposes a few different ways to use algorithm X to solve `Max-Prime-Sum-Matching` in $O(K^3)$ time but they are all wrong. For each algorithm, show that he is wrong by giving a counterexample with the **smallest possible** K. You should provide the input, correct answer, and a possible wrong answer given by the algorithm.

(a) Run algorithm X and just return the answer. (2 marks)

(b) If $N_1 > 1$, run algorithm X and return the answer. Otherwise, set $C_1 = 0$ and run algorithm X to get an answer M. Return $M + \lfloor C_1/2 \rfloor$. (2 marks)

(c) If $N_1 > 1$, run algorithm X and return the answer. Otherwise, run algorithm X to get an answer M. If there are multiple valid maximum flows, pick **any**. Get $L =$ the number of unmatched copies of N_1. Return $M + \lfloor L/2 \rfloor$. (2 marks)

6 Design an algorithm to solve `Max-Prime-Sum-Matching` and prove that it is correct. You can use algorithm X in your answer if needed. (16 marks)

- If your algorithm runs within $O(K^3)$ time, you can get **up to 16 marks**.
- Otherwise, if your algorithm runs within $O(K^3 \log K)$ time, you can get **up to 12 marks**.
- Otherwise, if your algorithm runs within $O(K^5)$ time, you can get **up to 6 marks**.
- Otherwise, no marks will be awarded.

Hint: You may have to run algorithm X more than once, and change the flow graph slightly each time.

– End of this Paper, All the Best –

– The last few pages are the blank answer sheets —