Student Name: _____

Student Number: _____

School: _____

Year of Entrance: _____

ShanghaiTech University Final Examination Cover Sheet

Academic Year: ____2022__to___2023___ **Term:** ___Fall_____

Course-offering School: __School of Information Science and Technology__

Instructor: _____Quan Li_____

Course Name: ___Algorithm Design and Analysis___

Course Number: ___CS240_____

Exam Instructions for Students:

1. All examination rules must be strictly observed throughout the entire test, and any form of cheating is prohibited.

2. Other than allowable materials, students taking closed-book tests must place their books, notes, tablets and any other electronic devices in places designated by the examiners.

3. Students taking open-book tests may use allowable materials authorized by the examiners. They must complete the exam independently without discussion with each other or exchange of materials.

For Marker's Use:

Section	1	2	3	4	5	6	7	8	9	10	Total
Marks											
Recheck											

Marker's Signature: **Rechecker's Signature:**

Date: **Date:**

Instructions for Examiners:

1. The format of the exam papers and answer sheets shall be determined by the school and examiners according to actual needs. All pages should be marked by the page numbers in order (except the cover page). All text should be legible, visually comfortable and easy to bind on the left side. A4 double-sided printing is recommended for the convenience of archiving (There are all-in-one printers in the university).

2.The examiners should make sure that exam questions are correct and appropriate. If errors are found in exam questions during the exam, the examiners should be responsible to respond on site, which will be taking into account in the teaching evaluation.

SHANGHAITECH UNIVERSITY
School of Information Science and Technology

CS240: Algorithm Design and Analysis – Final Exam

Instructor: Quan Li

Fall 2022/12/27

Student Name: _____
Student ID: _____

---

1. This is an open-book exam with **only a A4 cheat sheet** and without any electronic devices.
2. Write your name and student ID on this page. This exam contains 15 pages (including this cover page and scratch pages) and 8 questions. Total of points is 100. Answer all questions in the space provided. **Answers should be given in English.**
3. You can use results from the slides without proof, provided you state precisely what the result is that you are using. This does not hold, of course, if the question is to prove a certain result from the slides.
4. When describing an algorithm, you do not have to give pseudocode unless this is explicitly asked for. However, **your algorithm description should be clear and unambiguous.** Add explanations to your algorithm description / pseudocode where appropriate; your solution should be easily understandable.
5. Make sure that you define all variables, even when you are using the same variables as in the book. (For example, if you use $c(u, v)$ for the capacity of an edge (u, v) in a flow network, and this notation has not been defined in the question already, then write something like "*Let c(u, v) denote the capacity of the edge (u, v).*"
6. Please read the following declaration and **sign your name**. Good luck and happy reading!

I confirm that I have answered the questions using only materials specifically approved for use in this examination, that all the answers are my own work, and that I have not received any assistance during the examination.
Signature: _____

Distribution of Marks

Question	Points	Score
1	12	
2	12	
3	12	
4	10	
5	12	
6	12	
7	15	
8	15	
Total:	100	

1. (12 points) **Short Questions**

   (a) (3 points) For all problems $X \in NP$, $X \leq_p$ 3D-MATCHING. Is this true or false? Give a brief justification for your answer.

   (b) (3 points) The vertex cover problem and the independent set problem are equivalent, since on any graph $G = (V, E), C \in V$ is a vertex cover iff $V - C$ is an independent set. So the 2-approximation algorithm for the vertex cover algorithm is also a 2-approximation algorithm for the independent set problem. Is this true or false? Give a brief justification for your answer.

   (c) (3 points) We know that vertex cover is a special case of set cover, i.e., for any vertex cover problem, we can transfer it to a set cover problem. Are they equivalent to each other? Give a brief justification for your answer.

   (d) (3 points) Prove the problem "*Given a bipartite graph, is there a perfect matching?*" has good characterizations.

2. (12 points) **FPT (Fixed Parameter Tractable) Algorithms**

   (a) (5 points) For a problem parametrized by k (in addition to the input size n), suppose we have an algorithm with running time $O(k^k \cdot poly(n))$. For what values of k is this algorithm a polynomial-time algorithm?

   (b) (5 points) Prove that if a problem can be solved in $f(k) \cdot poly(n)$ times, then it can also be solved in $g(k) + poly(n)$ time for some function g.

   (c) (2 points) We know that whether a problem belongs to FPT crucially depends on the choice of the parameter k. If we choose $k = n$, what would the class FPT become?

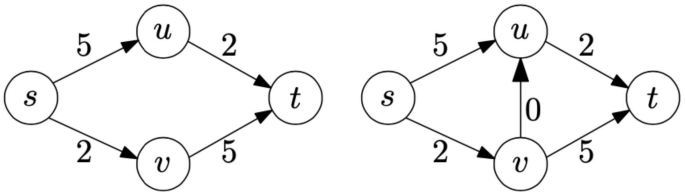
3. (12 points) **Amortized Analysis**

   Recall in class, we use a classic example – dynamic table to illustrate the idea of amortized analysis. That is, when we insert into a dynamic table, there are two possible behaviors: 1) There is still room in the array and we can simply add the value into the table; 2) There is no more room in the table so we need to make a new, larger table, copy over all the elements into the new table, and then finally add the new value. The first case is $O(1)$ whereas the second is $O(N)$, where N is the number of elements in the table. The average, or amortized order of appending to a dynamic table will depend on how often we have to resize. When we resize, we have two options for large to make the new internal array: 1) We could increase the size of the internal array by a constant k; 2) We could increase the size of the internal array by a constant factor k. Please perform an amortized analysis for **both cases**. We will add $N + 1$ elements to the lists (it makes the math a bit nicer than adding N). For both cases we assume $N \gg k$.

4. (10 points) **Selfish Multicast Routing with Congestion**

   Recall that in the *Multicast routing problem* discussed in class, we are given k player, each wanting to route one unit of flow from some source to some destination. Every edge e is associated with a cost c_e, and each player pays a price of $\frac{c_e}{x_e}$ where x_e is the number of players using edge e. Now suppose that the price each player pays is now $c_e \cdot x_e$. This for example can model the congestion caused by multiple players using the same edge, where the delay on an edge increases linearly as the number of players sharing that edge.

   For each of the following two networks, find a Nash equilibrium as well as the social optimum, including the corresponding price. Suppose there are two players, and both want to route from the same source s to the same destination t. Note that (b) is simply (a) but with one additional, zero-cost edge (which might correspond to a high-speed optical cable).



(a) Original network (b) Augmented network

5. (12 points) **Dominating Set**

   A *dominating set* for a graph $G = (V, E)$ is a subset D of V such that every vertex not in D is adjacent to at least one member of D. The dominating set problem asks to find the smallest dominating set for a given graph G. Define $x_u = 1$ if vertex u is picked in the dominating set and 0 otherwise. Write an integer linear program for this problem. Then use LP relaxation and rounding to give a $(d + 1)$-approximation for this problem, where d is the maximum degree of G.

6. (12 points) **FPTAS for Knapsack**

   Recall that in the FPTAS we gave for the *knapsack problem*, we rounded *up* all the values to the nearest multiple of θ, and then run the dynamic programming algorithm. Now consider the other two possible ways of rounding:

   (a) (6 points) If we round all values *down* to the nearest multiple of θ (for whatever θ that is necessary), and run the dynamic programming algorithm, does this still give an FPTAS?

   (b) (6 points) If we round all values *up* to the nearest power of θ (for whatever θ that is necessary), and run the dynamic programming algorithm, does this still give an FPTAS? If yes, give a proof. If no, say why.

7. (15 points) **Perfect Hashing**

You want to find a perfect hash function for mapping a given set of n items into a table of size m. A hash function is *perfect* if there are no collisions, i.e., each of the n items is mapped to a different slot in the hash table. Of course, a perfect hash function is only possible if $m \geq n$. Suppose you try to find a perfect hash function by brute force, namely, repeatedly pick idea random hash functions until you find one that happens to be perfect. Recall that an ideal random hash function maps each item to the slots uniformly and independently. Let $\alpha = \frac{n}{m}$ be the load factor.

(a) (3 points) Suppose you pick an ideal random hash function h. What is the exact (i.e., do not use asymptotic notation) expected number of pairs of items that collide, as a function of n and m?

(b) (3 points) Suppose you pick an ideal random hash function h. What is the exact expected fraction of items that do not collide with any other item. What is the limit of this fraction as m and n go to infinity, as a function of α?

(c) (3 points) What is the exact probability that a random hash function is perfect?

(d) (3 points) What is the exact expected number of ideal random hash functions you have to test before you find a perfect hash function?

(e) (3 points) Show that if $m \geq n^2$, then you expect to test a constant number of ideal random hash functions before you find a perfect hash function.

8. (15 points) **NP and Computational Intractability**

   Recall that the NP-completeness SUBSET-SUM problem asks, given a set of non-negative integers S and a target K, whether S has a subset S' with $\sum_{i \in S'} i = K$. The AVERAGE-SUM problem asks, given just a set of non-negative integers S, whether S has a subset S' with $\sum_{i \in S'} i = \frac{1}{|S|} \sum_{i \in S} i$, where $|S|$ is the number of elements in S. It is similar to the SUBSET-SUM problem, except now the target value is always the average value in S. Give a polynomial-time algorithm for AVERAGE-SUM or prove that it is NP-complete.

Intentionally left blank to accommodate work that wouldn't fit elsewhere and/or scratch work

Intentionally left blank to accommodate work that wouldn't fit elsewhere and/or scratch work

Intentionally left blank to accommodate work that wouldn't fit elsewhere and/or scratch work

Intentionally left blank to accommodate work that wouldn't fit elsewhere and/or scratch work

Intentionally left blank to accommodate work that wouldn't fit elsewhere and/or scratch work