

NAME: _____

ALGORITHMS QUALIFYING EXAM

This exam is open:

- books
- notes

and *closed*:

- neighbors
- calculators

The *upper bound* on exam time is 3 hours.

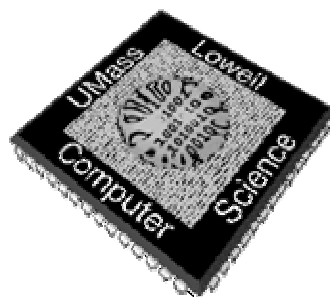
Please put all your work on the exam paper.

(Partial credit will only be given if your work is shown.)

If you use pseudocode from a reference, such as the Cormen *et al.* textbook, you can just give the page reference instead of writing that pseudocode on your exam paper.

All algorithms referred to here use a sequential model of computation, not a parallel one.

Good luck!



Part 1

(85 points)

Solve all of problems 1-7**1: (15 points) NP-Completeness**

Given: Undirected graph $G = (V, E)$, weight $w(e) \in \mathbb{Z}^+, \forall e \in E$, bound $B \in \mathbb{Z}^+$, a subset C of cliques of G .

Question: Does there exist a simple cycle in G that has total weight $\geq B$ and contains at most one edge from each clique in C ?

Either prove that this problem is NP-complete or provide a polynomial-time algorithm for solving it.

this page left blank for work

2: (10 points) Flow Networks

Given a flow network $G = (V, E)$ with source s and sink t , a cut (S, T) is a partition of V into S and $T = V - S$ such that $s \in S$, $t \in T$. Let (S, T) be a cut of G such that $|T| > 1$. Let s' be a vertex of T such that $s' \neq t$. Let $G' = (T, E')$ where E' contains every edge $e = (u, v)$ such that $e \in E$, $u, v \in T$. That is, E' is the subgraph of G induced by T . Suppose that there exists a cut (S', T') of G' such that $|S'| \geq 1$, $|T'| \geq 1$, $s' \in S'$, $t \in T'$.

Consider the following statement about the flow network:

$$f(S, T) = f(S, S') + f(S', T')$$

where $f(X, Y) = \sum_{x \in X} \sum_{y \in Y} f(x, y)$ and $f(x, y)$ denotes the flow from vertex x to vertex y .

Either prove the statement or provide a counterexample.

this page left blank for work

3: (10 points) Graph Algorithms

Given a directed, strongly connected graph $G=(V,E)$, we can determine whether or not it contains a directed cycle of odd length as follows:

- **Perform Depth-First-Search (DFS) on G starting from an arbitrary vertex v .**
- **During the DFS, mark each vertex as either even or odd. That is, we first mark v as even then, for each edge (u,w) , we mark w with the opposite mark of u .**
- **If we attempt to mark a vertex that is already marked by the opposite mark, then we conclude that there is a cycle of odd length.**

Can we use this approach to test for the existence of an *even* cycle in a directed, strongly connected graph? Why or why not?

this page left blank for work

4: (15 points) Dynamic Programming

An investor has \$8,000 to invest. This can be spread across 3 investment programs. Investment can only be in integral multiples of \$1,000. The investor's goal is to maximize total return on investment.

The return function $R_i(x)$ for $x = 0, \dots, 8$ for each of the 3 investment programs is given.

Example of $R_i(x)$:

Investment (in units of \$1,000)	0	1	2	3	4	5	6	7	8
Program 1	0	5	15	40	80	90	95	98	100
Program 2	0	5	15	40	60	70	70	74	75
Program 3	0	4	26	40	45	50	50	52	53

Show how this investment problem can be solved using dynamic programming by establishing the optimal substructure of the optimal solution.

this page left blank for work

5: (10 points) Number-Theoretic Algorithms.

- a) (4pts) Find all solutions to the equations $x \equiv 4 \pmod{5}$ and $x \equiv 5 \pmod{11}$.
- b) (3 pts) What do the Chinese Remainder Theorem and its Corollaries allow you to say about $x \pmod{55}$? Explain.
- c) (3 pts) Is the general result used in b) what you need to show that RSA is correct? Explain.

6: (10 points) Amortized Analysis. Common Lisp provides a hash table data structure which is automatically maintained by the system. At the beginning a “small” table is allocated. As the table becomes full, a new table twice the size is allocated, the old table is copied over and the memory allocated to it is released. Explain the comparative advantages and disadvantages of open addressing and chaining. In particular,

- a) (6 pts) explain (with appropriate references) why chaining will allow you to wait for the table to be full (load factor $\alpha = 1$) and still maintain an amortized linear cost for insertions (sketch the proof of this part), while the open addressing scheme will force you to create a new table well before the load factor is close to 1 (explain in detail) if you want reasonable amortized linear costs.
- b) (4 pts) What are the implications for the strategy required to support a Delete-Table operation that allows us to contract tables with too few entries? Explain carefully, since there are at least two interacting issues to address.

this page left blank for work

- 7: (15 points) Greedy Approximations.** Consider the problem of making change for n cents using the least number of coins.
- a) (3 pts) Describe a greedy algorithm to make change consisting of quarters, dimes, nickels and pennies.
 - b) (3 pts) Prove that your algorithm yields an optimal solution.
 - c) (2 pts) What is the time complexity of the greedy algorithm in a)?
 - d) (3 pts) Give a set of coin denominations for which the greedy algorithm does not yield an optimal solution. Prove your assertion.
 - e) (2 pts) What property of the set in part c) is crucial to the failure? Explain.
 - f) (2 pts) What is the time complexity of the general “make change” algorithm with arbitrary coin denominations? Explain.

this page left blank for work

Part 2
(15 points)

Solve one of problems 8-10.

8: (15 points) Linear Programming

A manufacturing company operates two plants. Three suppliers are willing to supply parts to the plants in the following amounts:

Supplier 1: 200 tons at \$10/ton
 Supplier 2: 300 tons at \$9/ton
 Supplier 3: 400 tons at \$8/ton

Shipping costs in dollars per ton are:

	TO Plant A	TO Plant B
FROM Supplier 1	2	2.5
FROM Supplier 2	1	1.5
FROM Supplier 3	5	3

Plant capacities and labor costs are:

	Plant A	Plant B
Capacity	450 tons	550 tons
Labor Cost	\$25/ton	\$20/ton

The assembled products are sold at \$50/ton to distributors. The company can sell at this price all that they can produce. How should the company plan its operations at the two plants so it maximizes its profits?

Formulate a linear program that solves this problem.

this page left blank for work

9: (15 points) Computational Geometry

The relative neighbor graph (RNG) of a set P of points in the plane is defined as follows: Two points p and q are connected by an edge in $\text{RNG}(P)$ if and only if:

$$d(p, q) \leq \min_{r \in P, r \neq p, q} (\max(d(p, r), d(q, r)))$$

where $d(p, q)$ is the Euclidean distance between p and q .

Design an algorithm that, given P , efficiently constructs $\text{RNG}(P)$.

Provide pseudocode, correctness justification and an upper bound on the worst-case asymptotic running time.

Hint: Two points p and q are connected by an edge in a Gabriel Graph if and only if the circle with diameter pq does not contain any other point of P in its interior. You may assume that a procedure $\text{GG}(P)$ is available. $\text{GG}(P)$ returns the Gabriel Graph of P using only $O(n \lg n)$ worst-case time. The number of edges in the Gabriel Graph is guaranteed to be in $O(n)$. You may decide which graph representation is used to store the result of $\text{GG}(P)$.

this page left blank for work

10: (15 points) Matrix Multiplication or FFT

Let $M(n)$ be the time to multiply two $n \times n$ Boolean matrices, and let $T(n)$ be the time to find the transitive closure of an $n \times n$ Boolean matrix. Show that an $M(n)$ -time Boolean matrix multiplication algorithm implies an $O(M(n) \lg n)$ -time transitive-closure algorithm, and a $T(n)$ -time transitive closure algorithm implies an $O(T(n))$ -time Boolean matrix multiplication algorithm. The proof involves filling in the details in the following steps:

- a) you are given that, if A is a Boolean matrix (e.g., the adjacency matrix of a graph), the Transitive Closure of A is $I + A + \dots + A^{n-1} = (I + A)^{n-1}$. Recall that, for Boolean matrices $A + A = A$ and that each power of A (in the adjacency matrix interpretation) provides us with the paths of length corresponding to the power. No need to prove anything here.
- b) (5 pts) Prove (e.g., by induction) that, if $P(n)$ is the fastest time to find the n -th power of an $n \times n$ matrix, then $P(n)$ is an $O(M(n) \lg n)$ -time algorithm.
- c) (5 pts) What is the Transitive Closure of the Boolean matrix $\begin{bmatrix} I_n & A & 0 \\ 0 & I_n & B \\ 0 & 0 & I_n \end{bmatrix}$ where A and B are Boolean matrices?
- d) (5 pts) Use the results in a), b) and c) to complete the proof.

this page left blank for work

this page left blank for work