Final Exam Solutions

- Do not open this quiz booklet until you are directed to do so. Read all the instructions first.
- The quiz contains 12 problems, several with multiple parts. You have 180 minutes to earn 150 points.
- This quiz booklet contains 15 pages, including this one, and a sheet of scratch paper.
- This quiz is closed book. You may use two double-sided letter ($8\frac{1}{2}'' \times 11''$) or A4 crib sheet. No calculators or programmable devices are permitted. Cell phones must be put away.
- Write your solutions in the space provided. If you run out of space, continue your answer on the back of the same sheet and make a notation on the front of the sheet.
- Do not waste time deriving facts that we have studied. Just cite results from class.
- When we ask you to "give an algorithm" in this quiz, describe your algorithm in English or pseudocode, and provide a short argument for correctness and running time. You do not need to provide a diagram or example unless it helps make your explanation clearer.
- Do not spend too much time on any one problem. Generally, a problem's point value is an indication of how many minutes to spend on it.
- Show your work, as partial credit will be given. You will be graded not only on the correctness of your answer, but also on the clarity with which you express it. Please be neat.
- Good luck!

Problem	Points	Parts	Grade	Problem	Points	Parts	Grade
0	3	1		7	10	1	
1	22	20		8	20	1	
2	10	2		9	10	1	
3	10	2		10	5	1	
4	15	3		11	10	2	
5	10	1		12	15	2	
6	10	1		Total	150		

Name: _____

Circle your recitation:

F10	F11	F11	F12	F12	F1	F2	F3
R01	R02	R07	R03	R08	R04	R05	R06
Yotam	Boon Teik	Aizana	Annie	Aizana	Annie	Katherine	Heejung

Problem 0. Name. [3 points] Write your name on every page of this exam booklet! Don't forget the cover.

Problem 1. True or False. [22 points] (20 parts)

Circle **T** or **F** for each of the following statements to indicate whether the statement is true (T) or false (F). **No explanation is needed.**

(a) T F Las Vegas algorithms are a class of randomized algorithms that always give the correct answer.

Solution: True.

(b) T F A Monte Carlo algorithm runs in worst-case polynomial time.

Solution: True.

(c) T F Prim's algorithm on a graph $G(V, E)$ takes $O(|V| \log |V|)$ time if implemented with a binary heap.

Solution: False. It takes $O(|E| \log |V|)$ time.

(d) T F If a flow f on a graph G has no augmenting paths, then $|f|$ is equal to the capacity of the minimum cut in G.

Solution: True.

(e) T F Paths along a minimum spanning tree are shortest paths.

Solution: False.

(f) T F Solving a linear program whose variables are constrained to be real numbers is NP-hard.

Solution: False. ILP is NP-hard, while (real-valued) LP is in P.

(g) T F The simplex algorithm is a polynomial-time algorithm.

Solution: False. The worse-case running time is exponential time, although it is often more efficient than known polynomial-time LP algorithms in practice.

(h) T F [3 points] Consider the following snapshot of the simplex algorithm, where z denotes the objective value, and $x_1, ..., x_5$ are variables constrained to be nonnegative. The current basic variables are x_4 and x_5. If x_2 is chosen as the entering variable for the next pivoting operation, then x_4 is the leaving variable.

$$\begin{array}{rcrcrcrcr} z & = & & & x_1 & + & 4x_2 & + & x_3 \\ x_4 & = & 6 & - & x_1 & - & 2x_2 & - & 3x_3 \\ x_5 & = & 10 & - & 2x_1 & - & 5x_2 & - & x_3 \end{array}$$

Solution: False. x_5 is the leaving variable. The second equation gives $x_2 \leq 3$, while the third equation gives $x_2 \leq 2$, a tighter bound.

(i) T F For NP-complete decision problems, there always exist certificates that can be verified in polynomial time.

Solution: True.

(j) T F NP-hard problems are problems that are strictly harder than all NP problems.

Solution: False. NP-hard problems are at least as hard as all NP problems.

(k) T F If problem A is NP-hard and can be reduced by Karp reduction to problem B, then B is also NP-hard.

Solution: True.

(l) T F Let A and B be optimization problems where it is known that A reduces to B in quadratic time. If there exists a polynomial-time 2-approximation for B, then there also exists a polynomial-time 2-approximation for A.

Solution: False. Approximation factor is not necessarily carried over in polynomial-time reduction. e.g., set cover vs. vertex cover.

(m) T F Assume P $\neq$ NP. The general Traveling Salesman Problem has a polynomial-time α-approximation algorithm for some constant $\alpha > 1$.

Solution: False. Assuming P $\neq$ NP, there is a polynomial-time approximation algorithm for TSP with the triangular inequality, but not the general TSP. For example, the case where all weights are 0 corresponds to the Hamiltonian Cycle problem, which is NP-complete.

(n) T F If a sequence of n operations on a data structure cost $T(n)$, then the amortized runtime of each operation in this sequence is $T(n)/n$.

Solution: True. This is the aggregate method.

(o) T F The amortized cost of any single operation should always be greater than or equal to the actual cost of that operation, in order to bound the total actual cost for any sequence of operations.

Solution: False. It is true for the total cost of a sequence of operations, but not necessarily for any single operation.

(p) T F Leader election in a bidirectional ring of n processes can be performed in $O(\log n)$ rounds.

Solution: True. Use the hierarchical algorithm from lecture.

(q) T F Using the simplified Luby's algorithm presented in lecture, a maximal independent set of vertices of an arbitrary undirected graph $G(V, E)$ can be found in expected $O(|V| \log |V|)$ time.

Solution: True. The maximum degree is bounded above by $|V|$.

(r) T F A hash function that is one-way (OW), target collision resistant (TCR) and non-malleable (NM) is guaranteed to be collision resistant (CR).

Solution: False.

(s) T F There is an existing deterministic polynomial-time algorithm that determines whether or not a number is composite.

Solution: True. Test if it is a prime, which takes polynomial time.

(t) T F A computational problem in P is guaranteed to have an interactive proof.

Solution: True. The execution of the polynomial-time algorithm itself is a legal proof.

Problem 2. Selection [10 points]

Consider the recursive expression for the running time $T(n)$ of SELECT(A, i), an algorithm that returns the ith smallest element in an array A of n distinct elements:

$$T(n) \leq \begin{cases} O(1) & \text{if } n < 140, \\ T(\lceil n/5 \rceil) + T(7n/10 + 6) + O(n) & \text{if } n \geq 140. \end{cases}$$

For each of the three terms in the latter case where $n \geq 140$, specify which step(s) in the following outline of SELECT contributed to the term. Briefly explain why.

SELECT(A, i)

1. Divide the n elements into groups of 5 elements each, sort each group, and find the median within each group.
2. Use SELECT recursively to find the median x of the medians found in step 1.
3. Partition the input array A around x. Suppose x is the kth largest element, i.e., there are $k - 1$ elements on the low side of the partition, and $n - k$ elements on the high side.
4. If $i = k$, then return x. Otherwise, use SELECT recursively to find the ith smallest element on the low side if $i < k$, or the $(i - k)$th smallest element on the high side if $i > k$.

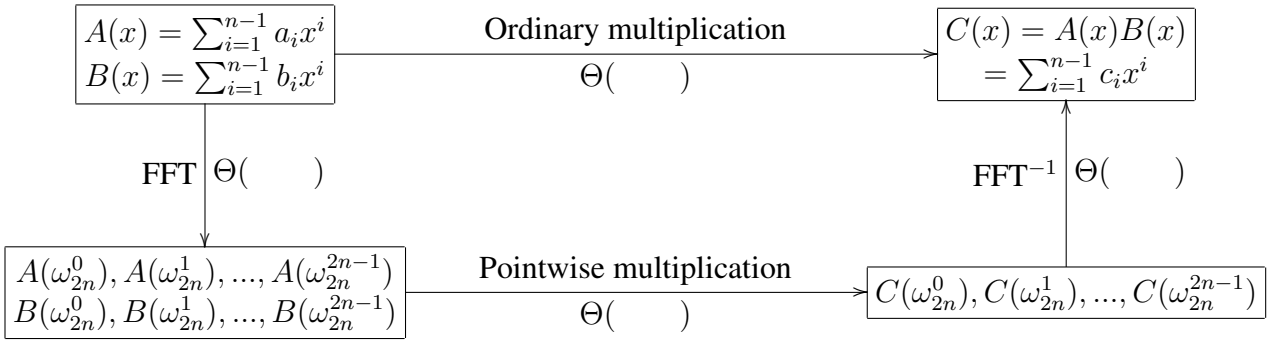
Term	Step(s)	Brief explanation
$T(\lceil n/5 \rceil)$		
$T(7n/10 + 6)$		
$O(n)$		

Solution:

- $T(\lceil n/5 \rceil)$: from step 2, recursively finding the median-of-medians;
- $T(7n/10 + 6)$: from step 4, recursively performing SELECT. The eliminated side contains at least $3\left(\lceil \frac{1}{2} \lceil \frac{n}{5} \rceil \rceil - 2\right) \geq \frac{3n}{10} - 6$ elements, so the chosen side contains at most $7n/10 + 6$ elements.
- $O(n)$: from step 1, sorting n groups of 5 elements each, and from step 3, partitioning the input around the median-of-medians.

Problem 3. Fast Fourier Transform [10 points] (2 parts)

The following diagram illustrates the outline of an efficient polynomial-multiplication process, using the Fast Fourier Transform (FFT):



(a) Fill in the running times above.

Solution: Ordinary multiplication: $\Theta(n^2)$, FFT and FFT^{-1}: $\Theta(n \log n)$, pointwise multiplication: $\Theta(n)$.

(b) What is $\omega_{2n}^k, k \in \{0, 1, ..., 2n-1\}$, in the lower two boxes? Describe it with a concise verbal explanation or terminology, as well as a mathematical expression. (Caution: Note that the expression involves $2n$ and not n.)

Solution: ω_{2n}^k is one of the $2n$ complex $(2n)$th roots of unity; in particular,

$$w_{2n}^k = e^{\frac{2\pi i k}{2n}}.$$

Problem 4. Modifying Paths [15 points] (3 parts)

You are given a weighted directed graph $G = (V, E)$ with weights $w\colon (u, v) \to \mathbb{R}$. Your friend has already computed the all pairs shortest path problem for this graph.

How long would it take to recompute the all pairs shortest paths if you only changed the weight of the $(n-1, n)$ edge in the following circumstances.

(a) Your friend gives you the answers to the subproblems from the $O(n^4)$ dynamic programming algorithm.

Solution: $O(n^4)$ time. While the first two iterations (filling out $d_{uv}^{(1)}$ and $d_{uv}^{(2)}$) can be done without recalculating most of the entries, the third iteration onwards may have many entries differing from the original problem. Thus, it will still take $O(n^4)$ time.
Using Johnson's algorithm for a time of $O(nm + n^2 \log n)$ or Floyd-Warshall for a time of $O(n^3)$ was also acceptable.

(b) Your friend gives you the intermediate matrices from the fast matrix multiplication algorithm.

Solution: $O(n^3 \log n)$ time. After taking the matrix to the fourth power, all entries may be different, which means that all the matrices have to be redone.
Using Johnson's algorithm for a time of $O(nm + n^2 \log n)$ or Floyd-Warshall for a time of $O(n^3)$ was also acceptable.

(c) Your friend gives you the answers to the subproblems from the Floyd-Warshall algorithm.

Solution: $O(n^2)$ time. For all $u \in V$, $C_{u,n}^{(n-1)}$ may use edge $(n-1, n)$. This changes $O(n)$ calculations. Subsequently, in calculating $C_{uv}^{(n)}$, all pairs $u, v \in V$ may use the result from $C_{u,n}^{(n-1)}$. This changes $O(n^2)$ calculations. Finally, the distance between $n-1$ and n may be different. Since no paths may use vertex $n-1$ or $n-1$ in the first $n-2$ iterations, the algorithm can skip calculating $C_{n-1,n}$ for these calculations. It takes $O(n)$ time to loop through the other vertices and find some intermediate value for the path from $n-1$ to n.

Problem 5. Edge connectivity [10 points] (1 parts)

The *edge connectivity* is the minimum number k of edges that must be removed to disconnect the graph into two or more components. Given an undirected, unweighted graph G with vertices V and edges E, design an algorithm which computes the edge connectivity. Provide the runtime analysis for your algorithm. You may give your runtime in terms of $T(n, m)$, the best algorithm for computing maximum flow values on a graph of n nodes and m edges.

Solution: This can be solved by making a flow network out of the graph, where all edges have edge capacity of 1. Choose one of the vertices V as a source node S, and iterate through the remaining vertices and set it as the sink node T. Running a maximum-flow algorithm with the given source and sink yields the minimum cut of the flow network which is equivalent to minimum number of edges that must be removed to disconnect the source from sink. Take the minimum of the max flow value over the sink node choices. Running the Ford-Fulkerson algorithm takes $O(|E| * k)$, which is bounded by $O(|E| * |V|)$. We run $|V - 1|$ iterations of the algorithm, so the entire algorithm has time complexity of $O(|E| * |V|^2)$.

Problem 6. Linear Program [10 points] (1 parts)

Consider the following linear program:

$$\begin{aligned} \max_{x,y} \quad & 5x - 3y \\ \text{subject to} \quad & x - y \leq 1 \\ & 2x + y \leq 2 \\ & x, y \geq 0 \end{aligned}$$

Show that $x = 1, y = 0$ is an optimal solution by writing down the dual problem and using duality (i.e., select a dual feasible solution to provide an upper bound on the optimal primal objective value).

Solution: The dual problem is

$$\begin{aligned} \min_{u,v} \quad & u + 2v \\ \text{subject to} \quad & u + 2v \geq 5 \\ & v - u \geq -3 \\ & u, v \geq 0 \end{aligned}$$

The solution $u = 1$, $v = 2$ gives an objective value of 5. By duality, this objective value 5 of the dual feasible solution is an upper bound on the objective value of any primal feasible solution. The objective value of the primal feasible solution $x = 1, y = 0$ is also 5. Therefore, $x = 1, y = 0$ is a primal optimal solution.

Problem 7. ILP and Total Unimodularity [10 points] (1 parts)

For any integer $n \times n$ matrix B and any integer $1 \times n$ vector c, prove that the following ILP (integer linear program) can be solved in polynomial time:

$$\begin{aligned} \max_{x} \quad & cx \\ \text{subject to} \quad & x_i - x_j \leq B_{ij} \qquad \forall i \in \{1, ..., n\}, j \in \{1, ..., n\} \\ & x_i \geq 0 \qquad \forall i \in \{1, ..., n\} \\ & x = [x_1 \quad x_2 \quad \cdots \quad x_n]^T \text{ is an integer-valued vector} \end{aligned}$$

where B_{ij} denotes the (i, j)-th entry in the matrix B.

Solution: The given linear program, without the integrality constraint, is of the form $Ax \leq B$, $x \geq 0$, and can be solved in polynomial time. The matrix A is of size $n^2 \times n$ where each row corresponds to one contraint. It's easy to show that A^T satisfies the sufficiency constraints for total unimodularity. The entries of A^T are 0, 1 and -1. The columns of A^T are the rows of A. Therefore, each column of A^T has exactly two non-zero entries: 1 and -1. We can partition the rows of A^T into $M_1 = \emptyset$ and $M_2 = M$, where M is the set of all rows of A^T. We have that for each column of A^T, the sum of entries in each partition equals to 0. Since A^T is TU, A also must be TU. This means that the optimal solution is integral and can be found in polynomial time.

Problem 8. Universal Hashing [20 points] (2 parts)

For a matrix A of size $m \times n$ with $\{0, 1\}$ entries, define the hash function

$$h_A(x) = Ax$$

where the input x is a binary column vector of length n and all operations are done $\mod 2$. In this problem you will show that the hash family H that consists of all such hash functions (for a fixed m and n) is universal.

(a) [10 points] Let x and x' be column vectors of length n where each entry is in $\{0, 1\}$ and $x \neq x'$. Let r be a random row vector of length n such that each entry is chosen from $\{0, 1\}$ uniformly and independently at random. Show that

$$Pr[rx = rx' \mod 2] = \frac{1}{2}$$

Solution:

For any two inputs $x \neq x'$, there is a difference at least in one entry. Without loss of generality, let there be a difference in the jth entry, i.e., let the index j be such that $x_j \neq x'_j$. Consider all possible $1 \times n$ rows with $\{0, 1\}$ entries. It is possible to pair them by difference in index j. For any such pair (r, r'), we have that

$$rx - rx' \neq r'x - r'x'.$$

Therefore, for a randomly drawn row vector r,

$$Pr[rx = rx'] = \frac{1}{2}.$$

(b) [10 points] Using the result from part (a), show that H is a universal hash family.

Solution:

Using the result from part (a), for a random matrix A, we have that

$$Pr[Ax = Ax'] = \left(\frac{1}{2}\right)^m$$

because $Ax = Ax'$ only when each bit of the output is the same. Each output bit is determined by a row in A and the output size is m. Therefore, we must have that H is a universal hash family.

Problem 9. Approximation [10 points] (1 parts)

Prove that there is no polynomial-time $\left(1 + \frac{1}{2n}\right)$-approximation algorithm for Vertex Cover (unless P = NP).

Solution: Solution: Assume that there exists an $(1 + \frac{1}{2n})$-approximation algorithm A for Vertex Cover. For a given graph $G = (V, E)$, let S be a minimum vertex cover for G. Then A can decide if G has a vertex cover of size at most

$$|S| \left(1 + \frac{1}{2n}\right) < |S| + 1$$

That implies that A can solve Vertex Cover in polynomial time. Contradiction.

Problem 10. Second Best is Leader [5 points] (1 parts)

Suppose we have a ring of n processes connected using bidirectional links. Assume that the processes have unique, strictly positive IDs. We wish to find the process with the *second* highest ID. Give an algorithm that can do this using $O(n^2)$ messages. You can quote algorithms from lecture.

Solution: [5 points]

1. In the first phase, find the maximum ID using the algorithm shown in lecture.
2. The process with maximum ID should pretend its ID is 0 and merely pass along the IDs it receives in the second phase (Step 3 below).
3. Find the process with the maximum ID in the remaining processes.

Since the algorithm given in lecture uses $O(n^2)$ messages, the above uses $O(n^2)$ messages.

Problem 11. Ben Bitdiddle Tries Cryptography [10 points] (2 parts) Ben Bitdiddle proposes the following public key scheme. Alice chooses a secret key x, and computes $g^x \ mod \ N$ as her public key. g and N are public parameters known to all. If Bob wants to encrypt a message m, he computes $g^{x \cdot m}$ and sends it to Alice.

(a) Ignoring questions of efficiency, is this scheme secure? State what computational problems need to be intractable for an adversary who knows g, N and $g^{x \cdot m}$ to not be able to discover m. The adversary does not know x.

Solution: [5 points] Very similar assumptions to Diffie-Hellman key exchange.

1. Discrete Log Problem is hard: Can't compute discrete log to find $x \cdot m$ from $g^{x \cdot m}$ and x from g^x.
2. Diffie-Hellman-like problem is hard: Seeing g^x and $g^{x \cdot m}$ the adversary shouldn't be able to discover m.

(b) Explain what Alice has to do to discover m. Indicate whether you think this scheme is efficient or not.

Solution: [5 points] Alice has x, and sees $g^{x \cdot m}$. She has to try to find m by raising g^x to some power so it equals $g^{x \cdot m}$ which she sees. This is difficult to do efficiently.

Problem 12. Duplicate customers [15 points] (2 parts)

Two servers 1 and 2 each have a list of customers, and they would like to figure out which customers are in the intersection of their lists. In the following, the names of the customers are represented as integers in $\{1, \ldots, D\}$. Each server has n distinct customers. Let S denote the set of customers in the intersection of the two lists.

(a) Server 1 sends a message to server 2, after which server 2 needs to produce the set S, based on the message and server 2's list. Show how the servers can do this in such a way that server 1 only sends at most $O(n \log |D|)$ bits to server 2.

Solution: [5 points]

Server 1 sends server 2 the names of his n customers. Each name needs only $\log |D|$ bits to specify.

(b) Server 1 sends a message to server 2, after which server 2 should be able to produce a set C based on the message and its own list, such that $S \subseteq C$, and with probability no less than $\frac{3}{4}$,

$$|C| < |S| + 0.1n.$$

Show how the servers can do this so that the number of bits that server 1 sends to server 2 is as efficient as you can achieve.

Hint: You may use Markov's inequality for a non-negative random variable X and $a > 0$:

$$Pr[X \geq a] \leq \frac{E[X]}{a}$$

Solution: [10 points]

Server 1 should send server 2 a Bloom filter of size $O(n)$ with a false positive rate of ϵ. Let X be the random variable that denotes the number false positives. Then,

$$E[X] \leq n\epsilon$$

We want $X < .1n$. Using the Markov's inequality:

$$Pr[X \geq .1n] \leq \frac{E[X]}{.1n} \leq \frac{\epsilon}{.1}$$

For $\epsilon \leq .025$, we have that $\frac{\epsilon}{.1} \leq \frac{1}{4}$. Then, with probability at least $\frac{3}{4}$, the server 2 can produce the desired C.

SCRATCH PAPER