Final Exam

- Do not open this exam booklet until you are directed to do so. Read all the instructions first.
- The exam contains 9 problems, with multiple parts. You have 180 minutes to earn 180 points.
- This exam booklet contains 18 pages, including this one.
- This exam is closed book. You may use three double-sided letter ($8\frac{1}{2}'' \times 11''$) or A4 crib sheets. No calculators or programmable devices are permitted. Cell phones must be put away.
- 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 exam, 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!

Q	Title	Points	Parts	Grade	Q	Title	Points	Parts	Grade
1	True or False	56	14		6	Be the Computer	14	3	
2	Überstructure	10	1		7	Startups are Hard	20	3	
3	Meancorp	15	2		8	Load Balancing	15	2	
4	Forgetful Forrest	15	3		9	Distributed Coloring	20	3	
5	Piano Recital	15	3		Total		180		

Name: _____

Problem 1. True or False. [56 points] (14 parts)

Circle **T** or **F** for each of the following statements to indicate whether the statement is true or false and briefly explain why.

(a) T F [4 points] Suppose algorithm $\mathcal{A}$ has two steps, and $\mathcal{A}$ succeeds if both the steps succeed. If the two steps succeed with probability p_1 and p_2 respectively, then $\mathcal{A}$ succeeds with probability $p_1 p_2$.

(b) T F [4 points] If the divide-and-conquer convex hull algorithm (from Lecture 2) used a $\Theta(n^2)$ strategy to discover the maximum and minimum tangents, the overall algorithm would run in $\Theta(n^2 \log n)$ time.

(c) T F [4 points] In order to get an expected $\Theta(n \log n)$ runtime for "paranoid" quicksort (from Lecture 3), we require the recursive divide step to split the array into two subarrays each of at least $\frac{1}{4}$ the size of the original array.

(d) T F [4 points] A binary min-heap with n elements supports INSERT in $O(\log n)$ amortized time and DELETE-MIN in 0 amortized time.

(e) T F [4 points] The hash family $H = \{h_1, h_2\}$ is universal, where $h_1, h_2 : \{1, 2, 3\} \to \{0, 1\}$ are defined by the following table:

	1	2	3
h_1	0	1	0
h_2	1	0	1

(For example, $h_1(3) = 0$.)

(f) T F [4 points] Recall the $O(n^3 \lg n)$ matrix-multiplication algorithm to compute shortest paths, where we replaced the matrix-multiplication operator pair $(*, +)$ with $(+, \min)$. If we instead replace the operator pair with $(+, *)$, then we compute the product of the weights of all paths between each pair of vertices.

(g) T F [4 points] Negating all the edge weights in a weighted undirected graph G and then finding the minimum spanning tree gives us the *maximum*-weight spanning tree of the original graph G.

(h) T F [4 points] In a graph with unique edge weights, the spanning tree of second-lowest weight is unique.

(i) T F [4 points] In the recursion of the Floyd–Warshall algorithm:

$$d_{uv}^{(k)} = \min\{d_{uv}^{(k-1)}, d_{uk}^{(k-1)} + d_{kv}^{(k-1)}\},$$

$d_{uv}^{(k)}$ represents the length of the shortest path from vertex u to vertex v that contains at most k edges.

(j) T F [4 points] Consider a network of processes based on an arbitrary undirected graph $G = (V, E)$ with a distinguished vertex $v_0 \in V$. The process at each vertex $v \in V$ starts with a positive integer x_v. The goal is for the process at v_0 to compute the maximum $\max_{v \in V} x_v$. There is an asynchronous distributed algorithm that solves this problem using $O(diam^2 d)$ time and $O(E + diam \cdot n)$ messages.

(k) T F [4 points] Suppose a file server stores a hash of every file in addition to the file contents. When you download a file from the server, you also download the hash and confirm that it matches the file. This system securely verifies that the downloaded file has not been modified by an adversary, provided the hash function has collision resistance.

(l) T F [4 points] Suppose Alice, Bob, and Charlie secretly generate a, b and c, respectively, and publish $g^a \bmod p$, $g^b \bmod p$, and $g^c \bmod p$, where p is a prime. Then, Alice, Bob, and Charles can each compute $g^{abc} \bmod p$ as a shared secret known only to the three of them.

(m) T F [4 points] The number of memory transfers used by the best cache-oblivious algorithm is always at least the number of memory transfers used by the best external-memory algorithm for the same problem.

(n) T F [4 points] If there is a time-optimal divide-and-conquer algorithm for a problem, then that algorithm is also optimal with respect to memory transfers in the cache-oblivious model.

Problem 2. Überstructure [10 points] (1 part)

Design a data structure that maintains a dynamic set S of n elements subject to the following operations and time bounds:

Operation	**Effect**	**Time Bound**	
1. INSERT(x, S)	Insert x into S.	$O(\log n)$	expected amortized
2. DELETE(x, S)	Delete x from S.	$O(\log n)$	expected amortized
3. SUCCESSOR(x, S)	Find the smallest element in S larger than x.	$O(\log n)$	worst-case
4. FIND-MIN(S)	Return the smallest element in S.	$O(1)$	worst-case
5. SEARCH(x, S)	Return TRUE if element x is in S.	$O(1)$	expected

Describe how the operations are implemented on your data structure and justify their runtime.

Problem 3. Meancorp [15 points] (2 parts)

You are in charge of the salary database for Meancorp, which stores all employee salaries in a 2-3 tree ordered by salary. Meancorp compiles regular reports to the Department of Fairness about the salary for low-income employees in the firm. You are asked to implement a new database operation AVERAGE(x) which returns the average salary of all employees whose salary is at most x.

(a) [10 points] What extra information needs to be stored at each node? Describe how to answer an AVERAGE(x) query in $O(\lg n)$ time using this extra information.

(b) [5 points] Describe how to modify INSERT to maintain this information. Briefly justify that the worst-case running time for INSERT remains $O(\lg n)$.

Problem 4. Forgetful Forrest [15 points] (3 parts)

Prof. Forrest Gump is very forgetful, so he uses automatic calendar reminders for his appointments. For each reminder he receives for an event, he has a 50% chance of actually remembering the event (decided by an independent coin flip).

(a) [5 points] Suppose we send Forrest k reminders for each of n events. What is the expected number of appointments Forrest will remember? Give your answer in terms of k and n.

(b) [5 points] Suppose we send Forrest k reminders for a *single* event. How should we set k with respect to n so that Forrest will remember the event with high probability, i.e., $1 - 1/n^{\alpha}$?

(c) [5 points] Suppose we send Forrest k reminders for each of n events. How should we set k with respect to n so that Forrest will remember *all* the events with high probability, i.e., $1 - 1/n^{\alpha}$?

Problem 5. Piano Recital [15 points] (3 parts)

Prof. Chopin has a piano recital coming up, and in preparation, he wants to learn as many pieces as possible. There are m possible pieces he could learn. Each piece i takes p_i hours to learn.

Prof. Chopin has a total of T hours that he can study by himself (before getting bored). In addition, he has n piano teachers. Each teacher j will spend up to t_j hours teaching. The teachers are very strict, so they will teach Prof. Chopin only a single piece, and only if no other teacher is teaching him that piece.

Thus, to learn piece i, Prof. Chopin can either (1) learn it by himself by spending p_i of his T self-learning budget; or (2) he can choose a unique teacher j (not chosen for any other piece), learn together for $\min\{p_i, t_j\}$ hours, and if any hours remain ($p_i > t_j$), learn the rest using $p_i - t_j$ hours of his T self-learning budget. (Learning part of a piece is useless.)

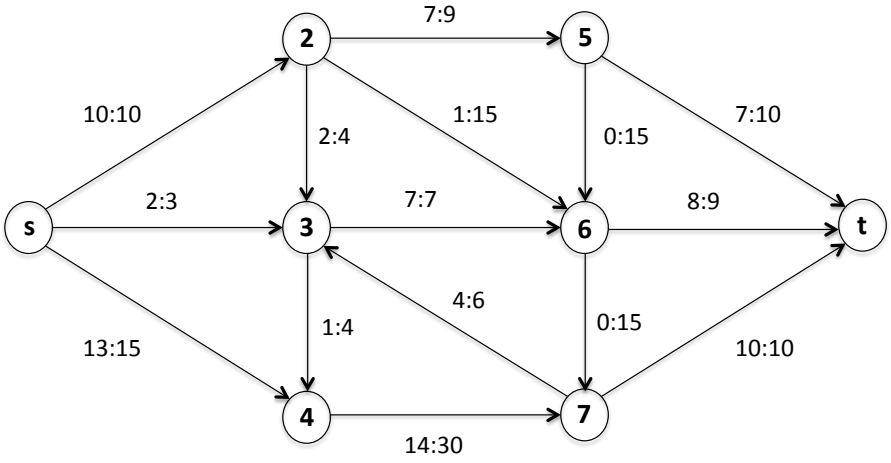
(a) [6 points] Assume that Prof. Chopin decides to learn exactly k pieces. Prove that he needs to consider only the k lowest p_is and the k highest t_js.

(b) [5 points] Assuming part (a), give an efficient greedy algorithm to determine whether Prof. Chopin can learn exactly k pieces. Argue its correctness.

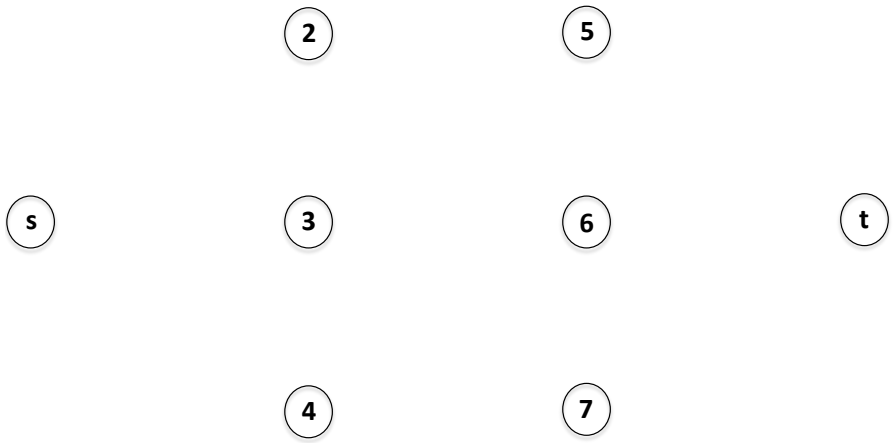
(c) [4 points] Using part (b) as a black box, give an efficient algorithm that finds the maximum number of pieces Prof. Chopin can learn. Analyze its running time.

Problem 6. Be the Computer [14 points] (3 parts)

Consider the following flow network and initial flow f. We will perform one iteration of the Edmonds–Karp algorithm.



(a) [5 points] Draw the residual graph G_f of G with respect to f.



(b) [4 points] List the vertices in the shortest augmenting path, that is, the augmenting path with the fewest possible edges.

(c) [5 points] Perform the augmentation. What is the value of the resulting flow?

Problem 7. Startups are Hard [20 points] (3 parts)

For your new startup company, *Uber for Algorithms*, you are trying to assign projects to employees. You have a set P of n projects and a set E of m employees. Each employee e can only work on one project, and each project $p \in P$ has a subset $E_p \subseteq E$ of employees that must be assigned to p to complete p. The decision problem we want to solve is whether we can assign the employees to projects such that we can complete (at least) k projects.

(a) [5 points] Give a straightforward algorithm that checks whether any subset of k projects can be completed to solve the decisional problem. Analyze its time complexity in terms of m, n, and k.

(b) [5 points] Is your algorithm in part (a) fixed-parameter tractable? Briefly explain.

(c) [10 points] Show that the problem is NP-hard via a reduction from 3D matching.

Recall the 3D matching problem: You are given three sets X, Y, Z, each of size m; a set $T \subseteq X \times Y \times Z$ of triples; and an integer k. The goal is to determine whether there is a subset $S \subseteq T$ of (at least) k disjoint triples.

Problem 8. Load Balancing [15 points] (2 parts)

Suppose you need to complete n jobs, and the time it takes to complete job i is t_i. You are given m identical machines $M_1, M_2, \ldots, M_m$ to run the jobs on. Each machine can run only one job at a time, and each job must be completely run on a single machine. If you assign a set $J_j \subseteq \{1, 2, \ldots, n\}$ of jobs to machine M_j, then it will need $T_j = \sum_{i \in J_j} t_i$ time. Your goal is to partition the n jobs among the m machines to minimize $\max_i T_i$.

(a) [5 points] Describe a greedy approximation algorithm for this problem.

(b) [10 points] Show that your algorithm from part (a) is a 2-approximation algorithm.
Hint: Determine an ideal bound on the optimal solution OPT. Then consider the machine M_ℓ with the longest T_ℓ, and the last job i^* that was added to it.

Problem 9. Distributed Coloring [20 points] (3 parts)

Consider an undirected graph $G = (V, E)$ in which every vertex has degree at most Δ. Define a new graph $G' = (V', E')$, the ***Cartesian product*** of G with a clique of size $\Delta + 1$. Specifically, V' is the set of pairs (v, i) for all vertices $v \in V$ and integers i with $0 \leq i \leq \Delta$, and E' consists of two types of edges:

1. For each edge $\{u, v\} \in E$, there is an edge between (u, i) and (v, i) in E', for all $0 \leq i \leq \Delta$. (Thus, each index i forms a copy of G.)
2. For each vertex $v \in V$, there is an edge between (v, i) and (v, j) in E', for all $i \neq j$ with $0 \leq i, j \leq \Delta$. (Thus each v forms a $(\Delta + 1)$-clique.)

Here is an example of this transformation with $\Delta = 3$:

Figure 1: Graph G.

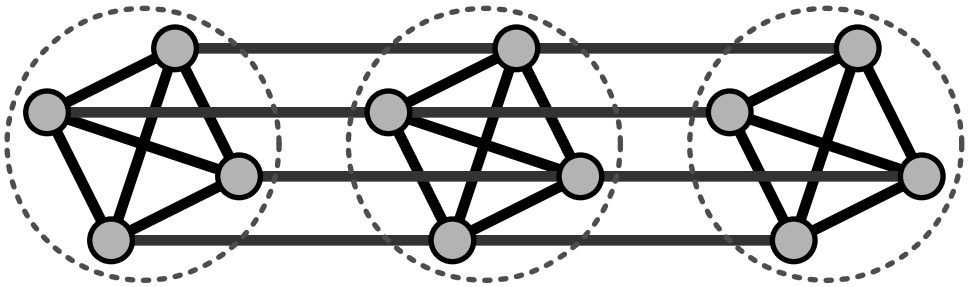


Figure 2: The Cartesian product G' of G and a clique of size 4.

(a) [8 points] Let S be any *maximal* independent set of G' (i.e., adding any other vertex to S would violate independence). Prove that, for each vertex $v \in V$, S contains exactly one of the $\Delta + 1$ vertices in V' of the form (v, i). *Hint:* Use the Pigeonhole Principle.

(b) [8 points] Now consider a synchronous network of processes based on the graph G, where every vertex knows an upper bound Δ on the degree. Give a distributed algorithm to find a vertex $(\Delta + 1)$-coloring of G, i.e., a mapping from vertices in V to colors in $\{0, 1, \ldots, \Delta\}$ such that adjacent vertices have distinct colors. The process associated with each vertex should output its color. Argue correctness.

Hint: Combine part (a) with Luby's algorithm.

(c) [4 points] Analyze the expected time and communication costs for solving the coloring problem in this way, including the cost of Luby's algorithm.

SCRATCH PAPER

SCRATCH PAPER

MIT OpenCourseWare
http://ocw.mit.edu

6.046J / 18.410J Design and Analysis of Algorithms
Spring 2015

For information about citing these materials or our Terms of Use, visit: http://ocw.mit.edu/terms.