Quiz 2

- Do not open this quiz booklet until directed to do so. Read all the instructions on this page.
- When the quiz begins, write your name on the top of every page of this quiz booklet. Circle your recitation at the bottom of this page.
- You have 120 minutes to earn a maximum of 120 points. Do not spend too much time on any one problem. Read them all first, and attack them in the order that allows you to make the most progress.
- **You are allowed two double-sided letter-sized sheets with your own notes**. No calculators, cell phones, or other programmable or communication devices are permitted.
- Write your solutions in the space provided. If you need more space, write on the scratch pages at the end of the exam, and refer to the scratch pages in the solution space provided. Pages will be scanned and separated for grading.
- Do not waste time and paper rederiving facts that we have studied. Simply cite them.
- When writing an algorithm, a **clear** description in English will suffice. Pseudo-code is not required. But be sure to prove the required bound on **running time** and explain **correctness**. Even if your running time is slower than the requested bound, you will likely receive partial credit if your algorithm and analysis are correct.
- **Pay close attention to the instructions for each problem**. Depending on the problem, partial credit may be awarded for incomplete answers.

Problem	Parts	Points	Grade	Grader
0: Name	0	2		
1: True or False, and Justify	12	48		
2: Two Weights, One Edge	1	10		
3: Red and Blue States	1	10		
4: free-fruit@mit.edu	4	15		
5: Algorithmic Adventure Time!	3	20		
6: Double Negative	1	15		
Total		120		

Name: _____

Circle your recitation:	R01 Skanda Koppula	R02 Ludwig Schmidt	R03 Parker Zhao	R04 Gregory Hui	R05 Yonadav Shavit	R06 Atalay Ileri	R07 Anton Anastasov
	R08 Akshay Ravikumar	R09 Aradhana Sinha	R10 Ray Hua Wu	R11 Daniel Zuo	R12 Themistoklis Gouleakis	R13 Adam Hesterberg	R14 Ali Vakilian

Problem 0. [2 points] **What is Your Name?** (2 parts)

(a) [1 point] Flip back to the cover page. Write your name and circle your recitation section.

(b) [1 point] Write your name on top of each page, including the scratch paper!

Problem 1. [48 points] **True or False, and Justify** (12 parts)

Circle T or F to indicate whether the statement is true or false, and explain your answer: if true, sketch a proof; and if false, give a counterexample. Your justification is worth more than your true or false designation.

(a) T F [4 points] Consider a hash table storing n items in m slots, with collisions resolved by chaining. Assume simple uniform hashing. Then SEARCH has expected running time $O(1)$.

(b) T F [4 points] Consider a hash table storing n keys in m slots, with resolutions resolved by chaining, and hash function

$$h(k) = ((k^2 + 3) \mod 8) \mod m.$$

Then DELETE has expected running time $\Theta(n/m)$.

(c) T F [4 points] Consider an empty ($n = 0$) hash table with $m = 4$ initial slots. Assume that the table maintains a load factor ($\alpha = n/m$) of at most 2 via table doubling (i.e., doubling the table size whenever the load factor is exceeded). Then, after inserting 28 keys into the table (and not deleting any), the hash table must have $m \geq 32$.

(d) T F [4 points] Suppose we dynamically resize a hash table by the following rules:

(i) If the table becomes full (you already have m items, and you try to insert one more), the table doubles in size: $m \to 2m$.

(ii) If the table becomes less than half full ($< \frac{m}{2}$ elements), the table halves in size: $m \to \frac{m}{2}$.

Then the amortized time complexity for resizing the table when adding or removing an element is $O(1)$.

(e) T F [4 points] In every directed acyclic graph, BFS and DFS visit the vertices in the same order.

(f) T F [4 points] Recall that a ***simple path*** is a path that visits no vertex more than once. In any (unweighted) undirected graph $G = (V, E)$, the number of simple paths between two vertices is at most $2^{|V|}$.

(g) T F [4 points] Consider the forest (vertex-disjoint set of trees) returned by a complete run of DFS on a **directed** graph. The number of trees in this forest will always be the same, regardless of the order in which you visit nodes in the outermost loop of DFS.

(h) T F [4 points] Consider the forest (vertex-disjoint set of trees) returned by a complete run of DFS on an **undirected** graph. The number of trees in this forest will always be the same, regardless of the order in which you visit nodes in the outermost loop of DFS.

(i) T F [4 points] When solving a single-source shortest-paths problem on a directed graph where all edges have weight 1 or 2, Dijkstra's algorithm will explore nodes in the same order that BFS explores them.

(j) T F [4 points] Consider a connected, undirected graph with more edges than vertices and only negative edge weights. Then, for any start node, Bellman–Ford necessarily detects a negative-weight cycle.

(k) T F [4 points] Recall that Johnson's algorithm reweights edges by creating a new node s, connecting s to every other vertex v with an edge (s, v) of weight 0, and running Bellman–Ford from s. The algorithm would still work if, instead, we picked an arbitrary existing node u, and added an edge (u, v) of weight 0 to every other vertex v for which edge (u, v) did not already exist.

(l) T F [4 points] Recall that the Floyd–Warshall algorithm involves three nested *for* loops, iterating through intermediate, start, and destination vertices, respectively. If we remove the middle loop and just use a fixed start vertex, then we obtain an $O(V^2)$-time single-source shortest-paths algorithm.

Problem 2. [10 points] **Two Weights, One Edge** (1 part)

Given an edge-weighted undirected graph $G = (V, E, w)$ where every edge e has an integer weight $w(e)$ equal to 0 or 1, design an algorithm to solve the single-source shortest-paths problem in $O(V + E)$ time.

Problem 3. [10 points] **Red and Blue States**[1] (1 part)

Let $G = (V, E)$ be an (unweighted) undirected graph. We want to color each vertex either red or blue in such a way that no vertices with the same color are connected by an edge. Design an $O(V + E)$-time algorithm that determines whether G has such a coloring.

[1] Too soon?

Problem 4. [15 points] **`free-fruit@mit.edu`** (4 parts)

Whole Shaw's Star Market Basket offers several different varieties of fruit, given by the set F. On day $i \in \{0, 1, \ldots, n-1\}$, their dumpster contains a subset $F_i \subseteq F$ of fruits (which you can predict via stock analysis). Unfortunately, your band of dumpster divers will let you eat **exactly one** fruit per day (and you cannot save fruit for later days, as it would go bad).

Each fruit $f \in F$ contains a number $c(f)$ of calories, and your goal is to maximize the number of calories you consume. The catch is that, in every window of three consecutive days, you must consume a total of between $c_{\min}$ and $c_{\max}$ calories (so that you do not die of starvation or overconsumption).

You and your freegan 6.006 TA decide to use dynamic programming to solve this problem—choosing exactly one fruit $f_i \in F_i$ for each day i to maximize total calorie consumption $\sum_i c(f_i)$ while satisfying the three-day-window min/max constraints.

(a) [3 points] Your TA defined the following subproblems: $T(i, d_2, d_1)$ is the maximum number of calories you can consume from the ith day through the final day, supposing that you ate fruit $d_2 \in F$ two days ago and you ate fruit $d_1 \in F$ one day ago (using special value None when we are in the first one or two days).

How many subproblems are there, as a function of n and $|F|$?

(b) [2 points] Supposing you have solutions to all subproblems $T(i, d_2, d_1)$, how can you solve the original problem?

(c) [7 points] Your TA wrote on a napkin the following recurrence for solving $T(i, d_2, d_1)$, but he failed to fill in some blanks (drawn here with boxes):

$$T(i, d_2, d_1) = \max\{\ \boxed{A} + T(\boxed{B}, \boxed{C}, \boxed{D}) \mid f \in F_i, \text{ where } \boxed{E} \text{ holds } \}$$

where $\max$ is defined to return $-\infty$ (meaning "impossible to survive") if the given set is empty.

Unfortunately, your TA is away on vacation, so you need to fill in the blanks yourself. Provide the expressions for $\boxed{A}$, $\boxed{B}$, $\boxed{C}$, and $\boxed{D}$ in terms of f and i. Also describe the condition $\boxed{E}$ to check your calorie consumption bounds.

(d) [3 points] What is the running time of the resulting dynamic program, as a function of n and $|F|$? Assume that, for any $f \in F$, $c(f)$ can be computed in constant time.

Problem 5. [20 points] **Algorithmic Adventure Time!** (3 parts)

Finn and Jake are trying to travel through the wonderful country of Dijkstan, which consists of ***cities*** $V = \{v_1, v_2, \ldots, v_{|V|}\}$ and one-way ***flights*** $E = \{e_1, e_2, \ldots, e_{|E|}\}$ connecting ordered pairs of cities. Each flight $e \in E$ has a positive integer ***time*** $t(e)$ and positive integer ***cost*** $c(e)$ required to traverse it (measured in minutes and dollars, respectively).

You may assume that there exists a path between every pair of cities, and that there is at most one flight between any ordered pair of cities.

Answer the following questions to help Finn and Jake plan our their schedules. In the following parts, partial credit will be awarded for correct solutions, but the amount of credit depends on the efficiency of your algorithm. **Clearly specify your running time.**

(a) [5 points] Finn wants to travel from city s to city t using the path p that minimizes $a \cdot t(p) + b \cdot c(p)$, where $t(p)$ and $c(p)$ are the total time and cost of the edges (flights) along path p, respectively, and a and b are positive real numbers. Design an $O(V \log V + E)$-time algorithm to compute the optimal such route.

(b) [10 points] Jake wants to travel from city s to city t using the minimum time, subject to not spending more than his positive integer budget of B dollars, Design an $O(BV \log(BV) + BE)$-time algorithm to compute the optimal such route.

(c) [5 points] Princess Bubblegum is considering removing some flights from Dijkstan. Call a flight $e \in E$ ***useless*** if e is not used by any optimal path, according to Finn's objective from part (a), between any pair (s, t) of cities. Design an $O(V^2 \log V + VE)$-time algorithm to find all useless flights.

Problem 6. [15 points] **Double Negative** (1 part)

Suppose you are given an edge-weighted directed graph $G = (V, E, w)$ which has exactly **two** negative-weight edges $e_1, e_2 \in E$. Design an $O(V \log V + E)$-time algorithm to detect whether G has a negative-weight cycle.

Partial credit will be awarded for answers that solve the problem for exactly one negative edge.

SCRATCH PAPER 1. DO NOT REMOVE FROM THE EXAM.

You can use this paper to write a longer solution if you run out of space, but be sure to refer to the scratch paper next to the question itself. Also be sure to write your name on the scratch paper!

SCRATCH PAPER 2. DO NOT REMOVE FROM THE EXAM.

You can use this paper to write a longer solution if you run out of space, but be sure to refer to the scratch paper next to the question itself. Also be sure to write your name on the scratch paper!