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.
- You have 120 minutes to earn a maximum of 120 points. Do not spend too much time on any one problem. Skim 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 sheet with your own notes**. No calculators, cell phones, or other programmable or communication devices are permitted.
- Write your solutions in the space provided. Pages will be scanned and separated for grading. If you need more space, write "Continued on S1" (or S2, S3, S4) and continue your solution on the referenced scratch page at the end of the exam.
- Do not waste time and paper rederiving facts that we have studied in lecture, recitation, or problem sets. Simply cite them.
- When writing an algorithm, a **clear** description in English will suffice. Pseudo-code is not required. Be sure to **briefly** argue that your **algorithm is correct**, and analyze the **asymptotic running time of your algorithm**. Even if your algorithm does not meet a requested bound, you **may** receive partial credit for inefficient solutions that are correct.

Problem	Parts	Points
0: Information	2	2
1: Connect 4ward	2	8
2: Counting Blobs	1	10
3: HAM DAG	1	10
4: Long Paths	1	15
5: Unicycles	1	15
6: Doh!-nut	2	30
7: Selfish Coins	1	15
8: Alternating Sums	1	15
Total		120

Name: ____________________

School Email: ____________________

Problem 0. [2 points] **Information** (2 parts)

(a) [1 point] Write your name and email address on the cover page.

(b) [1 point] Write your name at the top of each page.

Problem 1. [8 points] **Connect 4ward** (2 parts)

The **super-cool** graph of order n is a graph on vertices $v_1, v_2, \ldots, v_n$ with a directed edge from vertex v_i to vertex v_j whenever $i < j$.

(a) [2 points] Draw the super-cool graph of order 4.

(b) [6 points] Given a super-cool graph of order n, run depth-first search from vertex v_1, always searching neighbors with lower index first (i.e., if vertex v_i and v_j are both neighbors with $i < j$, explore v_i before v_j). In the boxes below, write the number of tree, back, forward, and cross edges in the resultant DFS tree **in terms of** n.

Tree	Back	Forward	Cross

Problem 2. [10 points] **Counting Blobs** (1 part)

An **image** is a 2D grid of black and white square pixels where each white pixel is contained in a **blob**. Two white pixels are in the same blob if they share an edge of the grid. Black pixels are not contained in blobs. Given an $n \times m$ array representing an image, describe an $O(nm)$ algorithm to count the number of blobs in the image.

Problem 3. [10 points] **HAM DAG** (1 part)

A **Hamiltonian path** is a path in a directed graph that visits every vertex exactly once. Describe a linear time algorithm to determine whether a directed **acyclic** graph $G = (V, E)$ contains a Hamiltonian path. (**Hint:** It might help to draw a DAG which contains a Hamiltonian path.)

Problem 4. [15 points] **Long Paths** (1 part)

A directed graph is **strongly-connected** if there exists a path from every vertex to every other vertex. Given a weighted strongly-connected directed graph $G = (V, E)$ containing both positive and negative edge weights, describe an $O(|V||E|)$ time algorithm to determine whether there exists a (not necessarily simple) path from vertex s to vertex t that has path weight greater than w.

Problem 5. [15 points] **Unicycles** (1 part)

Given a **connected** weighted undirected graph $G = (V, E)$ having only positive weight edges containing **exactly one cycle**, describe an $O(|V|)$ time algorithm to determine the minimum weight path from vertex s to vertex t.

Problem 6. [30 points] **Doh!-nut** (2 parts)

Momer has just finished work at the FingSprield power plant (at location p), and needs to drive to his home (at location h). Momer knows that if his driving route comes within a one mile drive of a donut shop, he will stop and eat donuts, and his wife, Harge, will be angry. Momer knows the layout of FingSprield, which can be modeled as a set of n locations, with two-way roads of known length connecting some pairs of locations (you may assume that no location is incident to more than five roads), as well as the locations of the d donut shops and g grocery stores in the city. Note that for this problem, the two parts can be solved independently.

(a) [15 points] Describe an $O(n \log n)$ algorithm to find the shortest driving route from the power plant back home that avoids driving **within a one mile drive** of a donut shop (or determine that no such route exists). Please specify any graphs you may construct.

(b) [15 points] Just as Momer is getting ready to leave, Harge calls to tell Momer to pick up some groceries on his way home. A grocery store will satisfy Momer's cravings, so he no longer needs to avoid passing close to donut shops on his drive from the power plant back home. Describe an $O(n \log n)$ algorithm to find the shortest driving route from the power plant that passes at least one grocery store location on the way home (or determine that no such route exists). Please specify any graphs you may construct.

Problem 7. [15 points] **Selfish Coins** (1 part)

You are given a row of n coins, where each coin c_i has a different value v_i. You would like to select coins from the row to maximize the value of chosen coins. You may choose any subsequence of coins from left to right, except that if you select any two adjacent coins, you may not select either of the next two coins on their right (adjacent pairs of coins are **selfish**). For example, if the values of coins in the row are $(6, 7, 1, 8, 2)$, then choosing coins $\{7, 8, 2\}$ would be a subset achieving the maximum value sum 17 (if you had instead selected coins 6 and 7, the selfish condition would forbid you from also picking coins 1 or 8).

Given a row of coin values v_i for $i \in [1, n]$, describe a dynamic program that determines the maximum value of any subset of coins subject to the selfish constraint. Be sure to define a set of subproblems, relate the subproblems recursively, argue the relation is acyclic, provide base cases, construct a solution from the subproblems, and analyze running time. Correct but inefficient polynomial time dynamic programs will be awarded significant partial credit.

Problem 8. [15 points] **Alternating Sums** (1 part)

Given an array A containing n integers, describe a dynamic programming algorithm to find an increasing subsequence of array indices $B = (b_0, \ldots, b_{m-1})$, with $0 \leq b_0 < b_1 < \ldots < b_{m-1} < n$, that maximizes the alternating subsequence sum:

$$\sum_{i=0}^{m-1}(-1)^i A[b_i] = A[b_0] - A[b_1] + A[b_2] - A[b_3] + \ldots$$

For example, if $A = (5, 2, 7, 1, 3, 8)$, then $B = (0, 1, 2, 3, 5)$ would achieve the maximum alternating sum $5 - 2 + 7 - 1 + 8 = 17$. Be sure to define a set of subproblems, relate the subproblems recursively, argue the relation is acyclic, provide base cases, construct a solution from the subproblems, and analyze running time. Correct but inefficient polynomial time dynamic programs will be awarded significant partial credit.

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.

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.

SCRATCH PAPER 3. 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.

SCRATCH PAPER 4. 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.

SCRATCH PAPER 4. 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.