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, S5) 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 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.
- **Pay close attention to the instructions for each problem**. Depending on the problem, partial credit may be awarded for incomplete answers.

Problem	Parts	Points
0: Information	2	2
1: Relax!	3	12
2: Negative-Weight Cycles	1	8
3: Directed Graph Radius	1	8
4: Future MIT	1	15
5: Gone to the Dogs	1	15
6: Troll Tolls	1	15
7: Muscle Mono-Toning	1	15
8: Super Lario World	1	15
9: Dealer's Choice	1	15
Total		120

Name: ____________________

School Email: ____________________

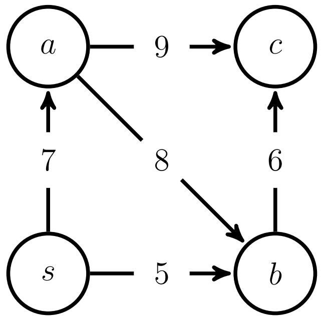
Problem 0. [2 points] **Information** (2 parts)

(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. [12 points] **Relax!**

Consider the graph $G = (V, E)$, where $V = \{a, b, c, s\}$, $E = \{(a, b), (a, c), (b, c), (s, a), (s, b)\}$, with weights as shown below. Each of the single-source shortest paths algorithms below repeatedly calls the `relax(..., u, v)` function below on various edges (u, v).



```python
def relax(Adj, w, d, parent, u, v):
    # relax edge (u, v)
    if d[v] > d[u] + w(u, v):
        d[v] = d[u] + w(u, v)
        parent[v] = u
```

For each algorithm below, give the sequence of edges (u, v) on which `relax(..., u, v)` gets called, for some valid execution of the algorithm from source vertex s. Note that an algorithm might call `relax(..., u, v)` on the same edge (u, v) more than once, and a call to `relax(..., u, v)` **might not** decrease the value of `d[v]`.

(a) [4 points] DAG Relaxation

(b) [4 points] Dijkstra

(c) [4 points] Bellman-Ford

Problem 2. [8 points] **Negative-Weight Cycles**

Given a weighted directed graph, where **every edge** in the graph has **negative weight**, describe an efficient[1] algorithm to determine whether the graph contains a negative-weight cycle.

[1]By "efficient", we mean that faster correct algorithms will receive more points than slower ones.

Problem 3. [8 points] **Directed Graph Radius**

Given a weighted graph $G = (V, E, w)$, recall that the **radius** $r(a)$ of a vertex a is the largest weight of any minimum-weight path from a to any other vertex, i.e. $r(a) = \max\{\delta(a, b) \mid b \in V\}$; and the **graph radius** $R(G)$ of G is the smallest radius of any vertex, i.e. $R(G) = \min\{r(a) \mid a \in V\}$. Describe an $O(|V|^3)$-time algorithm to compute the graph radius of any **directed** weighted graph having **possibly negative** edge weights, but no negative-weight cycle.

Problem 4. [15 points] **Future MIT**

In the future, MIT has taken over all of Cambridge. The campus now consists of b buildings and t underground indoor tunnels, with each tunnel connecting a pair of buildings. In addition to the underground tunnel network, every pair of buildings is connected via an outside route. Describe an efficient[1] algorithm to determine the minimum number of times you must go outside in order to visit the inside of every building at least once, starting inside building 6006.

[1]By "efficient", we mean that faster correct algorithms will receive more points than slower ones.

Problem 5. [15 points] **Gone to the Dogs**

Most residents in Barkindog Town love dogs. In fact, by law, every stretch of road between intersections must be home to at least one dog (though many roads contain more than one). Resident Catnip Everdeen hates dogs. She wants to walk along roads from her home to the pet store (both located at road intersections) while minimizing canine interaction. Catnip has a map of all roads in Barkindog Town, marked with how many dogs live along each stretch of road. Given that k dogs live in Barkingdog Town, describe an $O(k)$-time algorithm to find a route from Catnip's house to the pet store that minimizes the number of dogs passed along the way.

Problem 6. [15 points] **Troll Tolls**

Balbo Biggins is in a mountaintop metropolis consisting of m mountain villages connected by b two-way rope bridges, where each bridge is patrolled by a troll. In order to cross a bridge, Balbo must pay a **toll**: some **positive integer number** of gold pieces, where each bridge troll may demand a different toll. Whenever Balbo arrives at a mountain village via a bridge, a village elder welcomes him with a gift of a single gold piece (even if he has been there before). Balbo has a map listing each mountain village and its corresponding height, and each bridge and its corresponding toll. Balbo wants to reach the unique highest village from his starting location, and starts with a fixed amount of gold, large enough to reach any mountain village. Describe an efficient[1] algorithm to determine the maximum amount of gold Balbo can have when entering the highest village.

[1]By "efficient", we mean that faster correct algorithms will receive more points than slower ones.

Problem 7. [15 points] **Muscle Mono-Toning**

Bain Usolt wants to do a running workout on his morning commute to work. He has measured the length of all r stretches of road in the city and the **unique** elevation of each of the city's n intersections, including the locations of his home and office. His running route follows a revolutionary muscle-toning technique called **Mono-Toning**: starting at home, he wants to only increase his elevation at each successive road intersection until he stops for a water break at some intersection. After the break, he wants to only decrease his elevation at each successive intersection all the way to his office. Describe an efficient[1] algorithm to help Bain find a Mono-Toning route from home to work that **maximizes** the length of his run.

[1]By "efficient", we mean that faster correct algorithms will receive more points than slower ones.

Problem 8. [15 points] **Super Lario World**

In a classic video game, Lario Muigi must traverse g game levels. At the end of each level are at most p pipes; Lario can choose to enter any one pipe which will transport him deterministically to the start of some other level. One pipe is **winning**: if Lario enters that pipe, he will win the game! Each level contains a positive number of **coins** that Lario can collect. In addition, some levels contain either a mushroom or a boss, but not both. Eating a **mushroom** will make Lario big (Lario may be either big or small). In a **boss** level, Lario must fight the boss which will decrease his size from big to small and cause Lario to lose 100 coins (Lario is allowed a negative number of coins); Lario will **lose the game** if he tries to fight a boss while small. You know which pipes connect to which levels, and the locations of all coins, mushrooms, and bosses. Re-entering a level will regenerate everything in that level. Assuming Lario begins the game with no coins and small size, describe an efficient[1] algorithm to determine the maximum number of coins (possibly negative) that Lario can have when he enters the winning pipe.

[1]By "efficient", we mean that faster correct algorithms will receive more points than slower ones.

Problem 9. [15 points] **Dealer's Choice**

Annie Doeshun plays a card game against a single opponent using a deck of $2n$ cards, where each card has a positive integer value. At the beginning of the game, Annie deals n cards to each player. After the deal, Annie's **advantage** will be the total value of cards dealt to her, minus the total value of cards dealt to her opponent. Annie has some flexibility when dealing cards: she first deals either **one** or **two** cards to her opponent from the top of the deck, immediately followed by dealing **the same number** of cards to herself. She repeats this procedure (dealing both players either one or two cards, opponent first) until all cards are evenly dealt. Given the values of the $2n$ cards in their deck order at the start of the deal, describe an $O(n)$-time **dynamic programming** algorithm to determine the largest advantage Annie can have at the end of the deal.

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 write "Continued on S1" on the problem statement's page.

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 write "Continued on S2" on the problem statement's page.

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 write "Continued on S3" on the problem statement's page.

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 write "Continued on S4" on the problem statement's page.

SCRATCH PAPER 5. 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 write “Continued on S5” on the problem statement’s page.