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 90 minutes to earn a maximum of 90 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) 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
1: Information	2	2
2: Exact Edges	1	16
3: Color Cost	1	18
4: Orkscapade	1	18
5: Count Cycles	1	18
6: Bellham's Fjord	1	18
Total		90

35

Name: PGD

School Email: _____

Problem 1. [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 2. [16 points] **Exact Edges**

Given a weighted, directed graph $G = (V, E, w)$ with positive and negative edge weights, and given a particular vertex $v \in V$, describe an $O(k|E|)$-time algorithm to return the minimum weight of any cycle containing vertex v that also has exactly k edges, or return that no such cycle exists. Recall that a cycle may repeat vertices/edges.

Use Bellman-Ford.

First, construct a graph for Bellman-Ford with k-duplicates. Then using DFS, find path from v_0 to v_k while adding up weights, if there isn't, return that no such cycle exists.

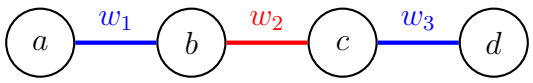
Among path weights, return minimum one.

Running Time: $k \cdot O(|E|) = O(k|E|)$

Problem 3. [18 points] **Color Cost**

A **3-color labeling** of a graph maps each edge to either red, green, or blue. The **color cost** of a 3-color labeled path is its path weight plus a positive integer w_c every time the path changes color.

For example, in the graph below (having four vertices $\{a, b, c, d\}$, a blue edge (a, b) with weight w_1, a red edge (b, c) with weight w_2, and another blue edge (c, d) with weight w_3) the path (a, b, c, d) has color cost $w_1 + w_2 + w_3 + 2w_c$, because the path changes color twice.



Given a 3-color labeling $c : E \to \{\text{red}, \text{green}, \text{blue}\}$ of a connected, weighted, undirected graph $G = (V, E, w)$ containing only **positive edge weights**, and given two vertices $s, t \in V$, describe an **efficient** algorithm to return a path from s to t having minimum color cost.
(By "efficient", we mean that faster correct algorithms will receive more points than slower ones.)

Let's define a graph $G' = (V', E', w')$.

For every vertex $v \in V$, make two additional vertices which are duplicate of v. And then label each vertex v_r, v_g, v_b, each of which represents red, green, blue respectively. $\forall e \in E$, e is incident to a vertex which has the same color. And v_r, v_g, and v_b which were derived from the same vertex $v \in V$ are connected by edges whose weights equal to w_c.

For a path π going through same vertex $u \in V$, if π traverses edges with same color, then it does not traverses an edge with w_c weight. If π traverses edges with different color, then π traverses an edge with w_c weight.

Then we can use Dijkstra on G' starting at s to find $\delta(s, t)$.

$|V'| = 3|V| = O(|V|)$, $|E'| = O(|E|)$. Thus the running time is $O(|V| \log(|V|) + |E|)$.

Problem 4. [18 points] **Orkscapade**

Ranger Raargorn needs to deliver a message from her home town of Tina's Mirth to the town of Riverdell, but the towns of Midgard have been overrun by an army of k Orks. Raargorn has a map of the n towns and $3n$ roads in Midgard, where each road connects a pair of towns in both directions. Scouts have determined the number of Orks $r_i \geq 1$ stationed in each town i (there is **at least one Ork** stationed in each town). Describe an $O(k)$-time algorithm to find a path from Tina's Mirth to Riverdell on which Raargorn will encounter the fewest total Orks in towns along the way. **Partial credit** will be awarded for slower correct algorithms, e.g., $O(k \log k)$ or $O(nk)$.

Build a graph $G = (V, E)$ such that $v \in V$ represents a town and $e \in E$ represents a road connecting between two towns. G is an undirected graph.

For every $v \in V$, there exists at least 1 ork in v, so $k = \Omega(|V|) = \Omega(n)$.

First of all, use BFS to determine level sets, then construct every edge to be a directed edge, such that for every edge $(u, v) \in E$, level of u is i and level of v is $(i+1)$. Then G become a DAG. Then label a weight for every edge (u, v) such that $w(u, v)$ = the number of orks in v. Then we can run DAG relaxation on G.

Running Time: $O(|V| + |E|) = O(n) = O(k)$ since $k = \Omega(n)$.

15

Problem 5. [18 points] **Count Cycles**

A **cycle-sparse** graph is any weighted directed simple graph $G = (V, E, w)$ for which every vertex $v \in V$ is reachable from at most one simple[1] negative-weight cycle in G. Given a cycle-sparse graph, describe an $O(|V|^3)$-time algorithm to return the number of negative-weight cycles in G.

First, make a vertex x and add edges (x, v) with 0 weight for every vertex $v \in V$. Then we Bellman-Ford on x to find SSSPs. Build G' from G by removing vertices not having negative infinite distance from x and their incident edges. Then the number of connected components = the number of negative weight cycles, since a connected compone in G' should contain only one negative cycle by assumption.

Running time : $O(|V||E|)$ · (Bellman-Ford)

$+ O(|V| + |E|)$ (Full-DFS)

$= O(|V||E|) = O(|V|^3)$ time.

[1] Recall a cycle is simple if visits any vertex at most once.

Problem 6. [18 points] **Bellham's Fjord**

Gralexandra Bellham wants to drive her electric car in Norway from location s in Oslo to a scenic Fjord at location t. She has a map of the n locations in Norway and the $O(n)$ one-way roads directly connecting pairs of them.

- Each location x is marked with its (positive or negative) integer **height** $h(x)$ above sea-level.
- Her car has **regenerative braking** allowing it to generate energy while going downhill. Each road from location x to y is marked with the integer energy $J(x, y)$ that the electric car will either spend (positive) or generate (negative) while driving along it.
- By the laws of physics, $J(x, y)$ is always strictly greater than the difference in **potential energy** between locations x and y, i.e., $J(x, y) > m \cdot g \cdot (h(y) - h(x))$, where m and g are the mass of the car and the acceleration due to gravity respectively.

Her car battery has very large **energy capacity** $b > 2nk$ where k is the maximum $|J(x, y)|$ of any road. Assuming she departs s at half capacity, $\lfloor b/2 \rfloor$, describe an $O(n \log n)$-time algorithm to determine the maximum amount of energy Bellham can have in her battery upon reaching t.
Partial credit will be awarded for slower correct algorithms, e.g., $O(n^2)$.

Let $G=(V,E)$ denote a graph that $v \in V$ denotes a location, $e \in E$ denotes a road between two locations. For any vertex $u, v \in V$, there exists either (u,v) or (v,u). Then use Bellman-Ford to get $\delta(s,t)$. If $\delta(s,t) = -\infty$, then b, otherwise $-\delta(s,t)$.

Continued on S1

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.

Timeout, the answer above is the answer I wrote in time. The answer below is irrelevant to the quiz. I just do additional work.

$J(x,y) > m \cdot g \cdot (h(y) - h(x))$,

every cycle in G is positive weight cycle.

Pf. Consider a cycle $C = (v_1, \ldots, v_k)$.

$$J(C) = \sum_{i=1}^{k-1} J(v_i, v_{i+1})$$

$$> \sum_{i=1}^{k-1} \left[m \cdot g \cdot (h(v_{i+1}) - h(v_i)) \right]$$

$$= mg \sum_{i=1}^{k-1} \left[h(v_{i+1}) - h(v_i) \right]$$

$$= mg\,(h(v_k) - h(v_1)) = 0. \quad \square$$

It's no use to traverse any cycle because it only consumes energy. So remove every cycles, searching cycles by Full-DFS. Then we use DAG Relaxation to find $\delta(s,t)$.

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.

6.006 Introduction to Algorithms

Spring 2020