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 every page of this quiz booklet.
- You have 120 minutes to earn 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.
- This quiz is closed book. You may use **two** $8\frac{1}{2}'' \times 11''$ or A4 crib sheets (both sides). No calculators or programmable devices are permitted. No cell phones or other communications devices are permitted.
- Write your solutions in the space provided. If you need more space, write on the back of the sheet containing the problem. Pages may be separated for grading.
- Do not waste time and paper rederiving facts that we have studied. It is sufficient to cite known results.
- When writing an algorithm, a **clear** description in English will suffice. Pseudo-code is not required.
- When asked for an algorithm, your algorithm should have the time complexity specified in the problem with a correct analysis. If you cannot find such an algorithm, you will generally receive partial credit for a slower algorithm **if you analyze your algorithm correctly**.
- 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. Be neat.
- Good luck!

Problem	Parts	Points	Grade	Grader
1	1	1		
2	14	28		
3	5	26		
4	5	25		
5	1	15		
6	3	25		
Total		120		

Name: _____

Wed/Fri Recitation:	Ying	Kevin	Sarah	Yafim	Victor
	10, 11 AM	**11 AM**	**12, 1 PM**	**12 PM**	**2, 3 PM**

Problem 1. [1 points] Write your name on top of each page.

Problem 2. True/False [28 points] (14 parts)

Circle (T)rue or (F)alse. You don't need to justify your choice.

(a) (T) F [2 points] Computing $\lfloor \sqrt{a} \rfloor$ for an n-bit positive integer a can be done in $O(\lg n)$ iterations of Newton's method.

(b) T (F) [2 points] Suppose we want to solve a polynomial equation $f(x) = 0$. While our choice of initial approximation x_0 will affect how quickly Newton's method converges, it will always converge eventually.

(c) T (F) [2 points] Karatsuba's integer multiplication algorithm always runs faster than the grade-school integer multiplication algorithm.

(d) T (F) [2 points] If we convert an n-digit base-256 number into base 2, the resulting number of digits is $\Theta(n^2)$.

(e) T (F) [2 points] In a weighted undirected graph $G = (V, E, w)$, breadth-first search from a vertex s finds single-source shortest paths from s (via parent pointers) in $O(V + E)$ time.

(f) (T) F [2 points] In a weighted undirected **tree** $G = (V, E, w)$, breadth-first search from a vertex s finds single-source shortest paths from s (via parent pointers) in $O(V + E)$ time.

(g) (T) F [2 points] In a weighted undirected **tree** $G = (V, E, w)$, **depth**-first search from a vertex s finds single-source shortest paths from s (via parent pointers) in $O(V + E)$ time.

(h) T (F) [2 points] If a graph represents tasks and their interdependencies (i.e., an edge (u, v) indicates that u must happen before v happens), then the breadth-first search order of vertices is a valid order in which to tackle the tasks.

(i) T (F) [2 points] Dijkstra's shortest-path algorithm may relax an edge more than once in a graph with a cycle.

(j) T (F) [2 points] Given a weighted directed graph $G = (V, E, w)$ and a source $s \in V$, if G has a negative-weight cycle somewhere, then the Bellman-Ford algorithm will necessarily compute an incorrect result for some $\delta(s, v)$.

(k) (T) F [2 points] In a weighted directed graph $G = (V, E, w)$ containing no zero- or positive-weight cycles, Bellman-Ford can find a *longest* (maximum-weight) path from vertex s to vertex t.

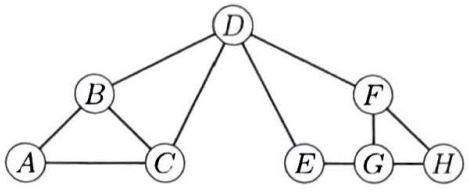
(l) T (F) [2 points] In a weighted directed graph $G = (V, E, w)$ containing a negative-weight cycle, running the Bellman-Ford algorithm from s will find a shortest acyclic path from s to a given destination vertex t.

(m) T (F) [2 points] The bidirectional Dijkstra algorithm runs asymptotically faster than the Dijkstra algorithm.

(n) (T)**F** [2 points] Given a weighted directed graph $G = (V, E, w)$ and a shortest path p from s to t, if we doubled the weight of every edge to produce $G' = (V, E, w')$, then p is also a shortest path in G'.

Problem 3. MazeCraft [26 points] (5 parts)

You are playing SnowStorm's new video game, *MazeCraft*. Realizing that you can convert a maze into a graph with vertices representing cells and edges representing passages, you want to use your newly learned graph-search algorithms to navigate the maze. Consider the following converted graph.



For the following questions, assume that the graph is represented using adjacency lists, and that **all adjacency lists are sorted**, i.e., the vertices in an adjacency list are always sorted alphabetically.

(a) [4 points] Suppose that you want to find a path from A to H. If you use breadth-first search, write down the resulting path as a sequence of vertices.

A B D f H

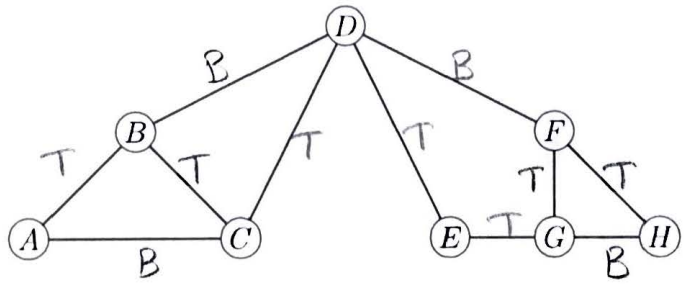
(b) [4 points] If you use depth-first search to find a path from A to H, write down the resulting path as a sequence of vertices.

A B C D E G f H

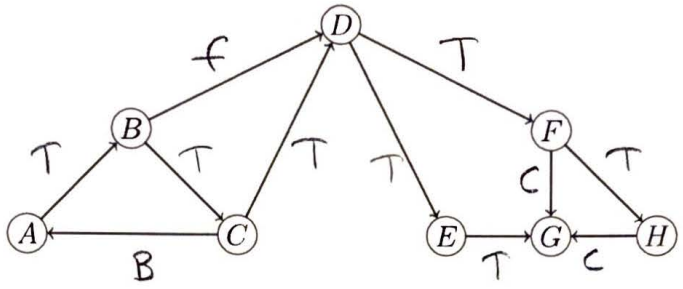
(c) [6 points] To determine whether the maze has cycles or multiple paths to the same destination, you decide to use the edge classification of depth-first search. Run depth-first search on the graph reproduced below, starting from vertex A, and label every edge with T if it's a tree edge, B if it's a back edge, F if it's a forward edge, and C if it's a cross edge.

As a reminder, recall that an edge (u, v) is

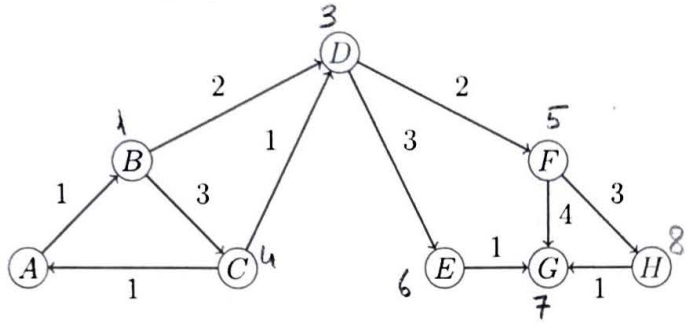
- a ***tree edge*** (T) if v was first discovered by exploring edge (u, v) (tree edges form the depth-first forest);
- a ***back edge*** (B) if v is u's ancestor in a depth-first tree;
- a ***forward edge*** (F) if v is u's descendant in a depth-first tree; and
- a ***cross edge*** (C) if none of the above apply.



(d) [6 points] Now suppose that the passages in the maze are directional. Rerun depth-first search in the directed graph below and label the edges accordingly.



(e) [6 points] Suppose each passage in the maze causes a different amount of ***damage*** to you in game. You change the graph to use weights to represent the damage caused by each edge. You then use Dijkstra's algorithm to find the path from A to H with the lowest possible damage. Write down the order in which vertices get removed from the priority queue when running Dijkstra's algorithm.



A B D C f E G H

Problem 4. Malfunctioning Calculator [25 points] (5 parts)

Former 6.006 student Mallory Funke is at a math competition, but her calculator isn't working. It seems to work fine for whole numbers, but the numbers after the decimal point always seem to be the sequence 07734 repeated over and over again, making those digits useless. For one of the problems, she has been asked to compute $\lfloor A^{3/4} \rfloor$ for a few different integer values of A. Mal knows that Newton's Method can be used to compute the square root or the cube root of an integer A. So as a first step in computing $\lfloor A^{3/4} \rfloor$, Mal wants to use Newton's Method to compute $\lfloor A^{1/4} \rfloor$. She then plans to use that information to compute $\lfloor A^{3/4} \rfloor$.

(a) [5 points] Mal decides to use the function $f(x) = x^4 - A$, because it has a root at $x = A^{1/4}$. Use Newton's Method on $f(x)$ to generate a formula for computing increasingly accurate estimates of $\lfloor A^{1/4} \rfloor$. In other words, give a formula for the more accurate estimate x_{i+1} in terms of a less accurate estimate x_i. The formula you construct must use **only** addition, subtraction, multiplication, and division. (You do not need to simplify the formula.)

Newton Raphson $\Rightarrow$ $x_{i+1} = x_i - \dfrac{f(x_i)}{f'(x_i)}$

$f'(x) = 4x^3$

$x_{i+1} = x_i - \dfrac{x_i^4 - A}{4x_i^3}$

$$x_{i+1} = x_i - \frac{x_i \cdot x_i \cdot x_i \cdot x_i - A}{4 \cdot x_i \cdot x_i \cdot x_i}$$

(b) [5 points] Mal decides to use the technique from part **(a)** to compute the value $B = \lfloor A^{1/4} \rfloor$. She then plans to compute $\lfloor A^{3/4} \rfloor$ by calculating the value $C = B^3 = B \cdot B \cdot B$. Provide an explanation of why this technique does not work.

Hint: Define α to be the fractional part of $A^{1/4}$, so that $B = A^{1/4} - \alpha$. What happens when you compute $C = B^3$?

$B^3 = (A^{\frac{1}{4}} - \alpha)^3 = A^{\frac{3}{4}} \underbrace{- 3A^{\frac{2}{4}}\alpha + 3A^{\frac{1}{4}}\alpha^2 - \alpha^3}_{\beta}$

If A is big enough, $\beta \geq 1$

According to this, $C = A^{3/4} \neq A^{3/4} - \beta$

(c) [5 points] Mal clearly needs a way to check her answer for $\lfloor A^{3/4} \rfloor$, using only integers. Given a pair of positive integers A and C, explain how to check whether $C \leq A^{3/4}$ using $O(1)$ additions, subtractions, multiplications, and comparisons.

$(C \leq A^{3/4})^4 = C^4 \leq A^3$

$C \cdot C \cdot C \cdot C \leq A \cdot A \cdot A$

(d) [5 points] Explain how to check whether $C = \lfloor A^{3/4} \rfloor$, again using only $O(1)$ additions, subtractions, multiplications, and comparisons.

Hint: Recall how the floor function is defined.

floor definition $\rightarrow C \leq A^{3/4}$ and $C + 1 > A^{3/4}$

we can solve it as in c section.

(e) [5 points] Give a brief description of an algorithm that takes as input a d-digit positive integer A and computes $\lfloor A^{3/4} \rfloor$. The only arithmetic operations you can use are integer addition, integer subtraction, integer multiplication, integer division, and integer comparison. Your algorithm should use $\Theta(\lg d)$ arithmetic operations in total, but partial credit will be awarded for using $\Theta(d)$ arithmetic operations. For this question, you may assume that Newton's Method has a quadratic rate of convergence for whatever function you devise.

ComputeNext(x_i, A)

let result be $x_i - (x_i \cdot x_i \cdot x_i \cdot x_i - A) / (x_i \cdot x_i \cdot x_i \cdot 4)$

return result.

Exp(a, b): uses multiplication

Compute(A):

Initilize x for A

do till $Exp(x, 4) \leq Exp(A, 3) < Exp(x+1, 4)$

$x = ComputeNext(x, A)$

return x

Compute function uses Newton Raphson method and has time complexity of $O(\log d)$

Problem 5. The Tourist [15 points]

Your new startup, *Bird Tours*, brings people around Boston in a new aviation car that can both drive and fly. You've constructed a weighted directed graph $G = (V, E, w)$ representing the best time to drive or fly between various city sites (represented by vertices in V). You've also written a history of Boston, which would be best described by visiting sites $v_0, v_1, \ldots, v_k$ in that order.

Your goal is to find the shortest path in G that visits $v_0, v_1, \ldots, v_k$ in order, possibly visiting other vertices in between. (The path must have $v_0, v_1, \ldots, v_k$ as a ***subsequence***; the path is allowed to visit a vertex more than once. For example, $v_0, v_2, v_1, v_2, \ldots, v_k$ is legal.) To do the computation, you've found an online service, Paths 'Я Us, that will compute the shortest path from a given source s to a given target t in a given weighted graph, for the bargain price of \$1. You see how to solve the problem by paying \$$k$, calling Paths 'Я Us with $(v_0, v_1, G), (v_1, v_2, G), \ldots, (v_{k-1}, v_k, G)$ and piecing together the paths. Describe how to solve the problem with only \$1 by calling Paths 'Я Us with (s, t, G') for a newly constructed graph $G' = (V', E', w')$, and converting the resulting path into a path in G.

for G':

Copy graph k times and label vertices and edges as in G^1 to G^k. Bind graphs as creating edges (v_1^1, v_1^2) to $(v_{k-1}^{k-1}, v_{k-1}^k)$ with 0 weights.

Then run 'Я Us v_1^1 to v_k^k. ('Я Us(v_1^1, v_k^k, G'))

What this does is, path has to visit and include v_{i+1}^{i+1} with shortest paths of every $(v_i^i\ v_i^{i+1})$ in order to find v_k^k.

for converting the result of 'Я Us (v_1^1, v_k^k, G'): Replace every $v' \in V'$ with $v \in V$ and remove dublicate vertexes (cycles we created to bind $(v_i^i\ v_i^{i+1})$) at path.

from → v_1^1 —— $v_2^1\ v_2^2$ —— $v_3^2\ v_3^3$ —— $v_{k-1}^{k-1}\ v_{k-1}^k$ —— v_k^k in G'

To → v_1 —— v_2 —— v_3 —— v_4 —— v_k in G

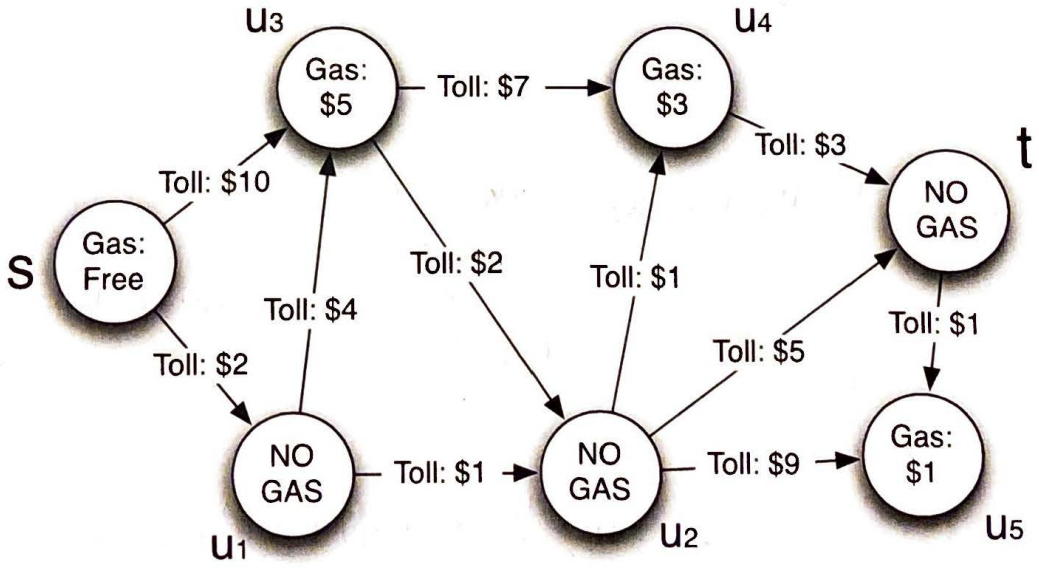
Problem 6. Fill 'Er Up! [25 points]

You are traveling by car from one city to another city. Unfortunately, you have a hole in your gas tank, and you have to refill your gas tank to travel across more than two roads. In addition, there is a toll booth on every road that charges you for using that road. Your goal is to find the least-expensive path from your start to your destination.

You represent the city network using a directed graph $G = (V, E, w)$ with weights w defined on both edges and vertices. The vertices V represent the cities and the edges E represent the roads. The weight $w(e)$ of an edge e represents the toll amount on that road. The weight $w(v)$ of a vertex v is the price of filling your gas tank in that city (which is a fixed price independent of how much gas you have left, or ∞ if there is no gas available to purchase). You are allowed (but not obligated) to end your journey with an empty tank, and you may assume that you always start your journey with a full tank.

Below is an example graph that we will use to answer part (a). One seemingly cheap path from s to t is (s, u_1, u_2, t) at a cost of \$8. Unfortunately, this path is not valid, because our leaky gas tank won't permit moving across three edges without refilling our gas tank.

One valid path is (s, u_3, u_2, t) at a cost of \$22. (This is a valid path: we begin with a full tank, travel across one edge to a gas station where we refill our tank, and then travel two edges to the destination, arriving with an empty gas tank. Notice that we are unable to continue the journey to u_5 if we wanted to, because even though there is a gas station there, we have to traverse a third edge to get to it.)



There are some extra copies of this graph at the end of the exam (for your convenience).

$S \to t$ { ~~$S, v_3, v_4, t \to 23$~~ ; $S, v_1, v_3, v_2, t \to 11$; ~~$S, v_3, v_2, t \to 22$~~

$S \to v_5$ { ~~$S, v_1, v_3, v_2, v_5 \to 22$~~ ; $S, v_1, v_3, v_2, v_4, t, v_5 \to 21$

(a) [5 points] The valid path given in the description above is not the *best* path. Find a least-expensive path from s to t in the graph above.

s, u_1, u_3, u_2, t

(b) [5 points] Find the least-expensive path from s to u_5 in the graph above.

$s, u_1, u_3, u_2, u_4, t, u_5$

(c) [15 points] Give an $O(V \log V + E)$ algorithm to find, in a given graph $G = (V, E, w)$, a least-expensive *valid* path from a city s to a city t, or report that no such path exists.

Transform G to G', for this

Visit every $v \in V$ and create V': v_e, v_h, v_f which stands for empty, half-full and full tank. Takes $O(V)$

Visit every $(u,v) \in E$ and create $(u_f, v_h), (u_h, v_e)$ with some weights. Also visit $v \in V$ and check if there is gas create $(v_h, v_f), (v_e, v_f)$ with weights of gas price at v. This takes $O(E)$ time.

Run Dijkstra, from s_f to t_e and t_h to find shortest path.

Transformation $O(V+E)$

\+ Dijkstra $O(V \log V + E)$

Algorithm takes $O(V \log V + E)$