Quiz 2

- Do not open this quiz booklet until you are 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.
- This quiz contains 5 problems, some with multiple parts. You have 120 minutes to earn 120 points.
- This quiz booklet contains 10 pages, including this one. Two extra sheets of scratch paper are attached. Please detach them before turning in your quiz at the end of the exam period.
- This quiz is closed book. You may use one $8\frac{1}{2}'' \times 11''$ or A4 crib sheet (both sides). No calculators or programmable 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. Do not put part of the answer to one problem on the back of the sheet for another problem, since the 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.
- Do not spend too much time on any one problem. Read them all through first, and attack them in the order that allows you to make the most progress.
- 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	7	35		
2	2	20		
3	2	20		
4	1	20		
5	1	25		
Total		120		

Name: _____

Circle your recitation time:
Hueihan Jhuang: **(10AM) (11AM)** Victor Costan **(2PM) (3PM)**

Problem 1. True or False [35 points] (7 parts)

Decide whether these statements are **True** or **False**. You must briefly justify all your answers to receive full credit.

(a) To sort n integers in the range from 1 to n^2, a good implementation of radix sort is asymptotically faster in the worst case than any comparison sort.

True **False**

Explain:

(b) Call an ordered pair (x_1, y_1) of numbers ***lexically less than*** an ordered pair (x_2, y_2) if either (i) $x_1 < x_2$ or (ii) $x_1 = x_2$ and $y_1 < y_2$. Then a set of ordered pairs can be sorted lexically by two passes of a sorting algorithm that only compares individual numbers.

True **False**

Explain:

(c) Any in-place sorting algorithm can be used as the auxiliary sort in radix sort.

True **False**

Explain:

(d) Every directed acyclic graph has only one topological ordering of its vertices.

True **False**

Explain:

(e) If the priority queue in Dijkstra's algorithm is implemented using a sorted linked list (where the list is kept sorted after each priority queue operation), then Dijkstra's algorithm would run in $O(E \lg V + V \lg V)$ time.

True **False**

Explain:

(f) In searching for a shortest path from vertex s to vertex t in a graph, two-way breadth-first search never visits more nodes than a normal one-way breadth-first search.

True **False**

Explain:

(g) Every sorting algorithm requires $\Omega(n \lg n)$ comparisons in the worst case to sort n elements.

True **False**

Explain:

Problem 2. Linear Dijkstra? [20 points] (2 parts)

(a) Professor Devamaine has just invented an exciting optimization for Dijkstra's algorithm that runs in $O(V + E)$ time for undirected graphs with edge weights of just 0 and 1.

Show the professor that the same time bound can be achieved simply by modifying the graph and then running breadth-first search as a black box.

(b) After hearing of his colleague's embarrassment, Professor Demaidas invents another modification to Dijkstra's algorithm that runs in $O(V + E)$ time for undirected graphs with edge weights of just 1 and 2.

Show the professor that the same time bound can again be achieved by modifying the graph and then running breadth-first search as a black box.

Problem 3. Bottleneck Path [20 points] (2 parts)

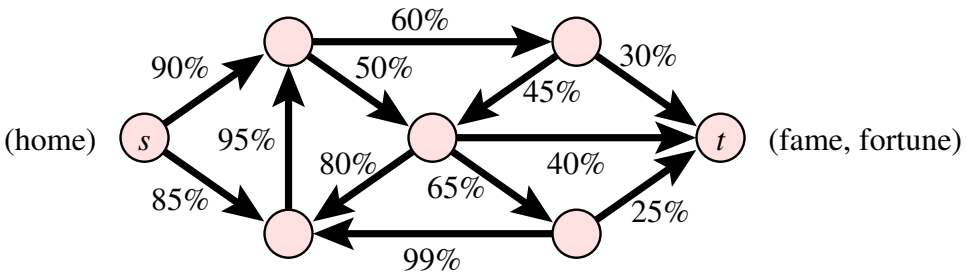
(a) In the ***bottleneck-path problem***, you are given a graph G with edge weights, two vertices s and t, and a particular weight W; your goal is to find a path from s to t in which every edge has at least weight W. Describe an efficient algorithm to solve this problem. Your algorithm should work even if the edge weights are negative and/or the particular weight W is negative.

(b) In the ***maximum-bottleneck-path problem***, you are given a graph G with edge weights, and two vertices s and t; your goal is to find a path from s to t whose minimum edge weight is maximized. Describe an efficient algorithm to solve this problem that uses an efficient algorithm from Part (a) as a subroutine. You may assume an efficient algorithm for Part (a) exists, and use it as a black box.

Problem 4. Reality [20 points]

You've just agreed to star in the new hit reality show, *Whose Geek Is It Anyway?* At the outset, you're given a map of an island of puzzles, which is a directed graph marked with a start vertex s and a goal vertex t. Traversing each edge e requires solving a puzzle, which you believe you can solve with probability $p(e)$. Describe how to modify the graph so that Dijkstra's algorithm will find a path from s to t that has the maximum probability of winning. (Assume your abilities to solve different puzzles are independent events.)

A sample input:



Problem 5. Negative-Weight Cycles [25 points]

If a directed graph $G = (V, E)$ contains a negative-weight cycle, shortest paths to some vertices will not exist, but there may still be shortest paths to other vertices. Assume that every vertex v is reachable from the source vertex s in G. Give an efficient algorithm that labels each vertex v with the shortest-path distance from s to v, or with $-\infty$ if no shortest path exists. (In other words, compute $\delta(s, v)$ for all v.) You can invoke all algorithms covered in lectures or recitations.

For reference, below is the pseudocode for Bellman Ford adapted from CLRS, which returns False if there are reachable negative weight cycles and True otherwise:

```
  def bellman_ford(V, E, w, s):
1     initialize_single_source(V, E, s)
2     for i in range(|V|-1):
3         for (u, v) in E:
4             relax(u, v, w)
5     for (u, v) in E:
6         if d[v] > d[u] + w(u, v):
7             return False
8     return True
```

SCRATCH PAPER

SCRATCH PAPER