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 through 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 sheet (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. **This quiz is shorter than the first, so we expect you to take the time to write clear and thorough solutions.**
- Good luck!

Problem	Parts	Points	Grade	Grader
1	2	2		
2	4	38		
3	2	20		
4	1	20		
5	3	20		
6	1	20		
Total		120		

Name: ______________________

Friday Recitation:	Aleksander **11 AM**	Arnab **12 PM**	Alina **3 PM**	Matthew **4 PM**

Problem 1. What is Your Name? [2 points] (2 parts)

(a) [1 point] Flip back to the cover page. Write your name there.

(b) [1 point] Flip back to the cover page. Circle your recitation section.

Problem 2. Short Answer [38 points] (4 parts)

(a) [9 points] Give an example of a graph such that running Dijkstra on it would give incorrect distances.

(b) [9 points] Give an efficient algorithm to sort n dates (represented as month-day-year and all from the 20^{th} century), and analyze the running time.

(c) [10 points] Give an $O(V + E)$-time algorithm to remove all the cycles in a directed graph $G = (V, E)$. Removing a cycle means removing an edge of the cycle. If there are k cycles in G, the algorithm should only remove $O(k)$ edges.

(d) [10 points] Let $G = (V, E)$ be a weighted, directed graph with exactly one negative-weight edge and no negative-weight cycles. Give an algorithm to find the shortest distance from s to all other vertices in V that has the same running time as Dijkstra.

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

We are given a directed graph $G = (V, E)$, and, for each edge $(u, v) \in E$, we are given a probability $f(u, v)$ that the edge may fail. These probabilities are independent. The reliability $\pi(p)$ of a path $p = (u_1, u_2, \ldots u_k)$ is the probability that no edge fails in the path, i.e.
$\pi(p) = (1 - f(u_1, u_2)) \cdot (1 - f(u_2, u_3)) \ldots \cdot (1 - f(u_{k-1}, u_k))$. Given a graph G, the edge failure probabilities, and two vertices $s, t \in V$, we are interested in finding a path from s to t of maximum reliability.

(a) [10 points] Propose an efficient algorithm to solve this problem. Analyze its running time.

(b) [10 points] You tend to be risk-averse and in addition to finding a most reliable simple path from s to t, you also want to find a next-most reliable simple path, and output these two paths. Propose an algorithm to solve the problem, argue its correctness, and give its asymptotic running time.

Problem 4. Flight Plans [20 points]

When an airline is compiling flight plans to all destinations from an airport it serves, the flight plans are plotted through the air over other airports in case the plane needs to make an emergency landing. In other words, flights can be taken only along pre-defined edges between airports. Two airports are adjacent if there is an edge between them. The airline also likes to ensure that all the airports along a flight plan will be no more than three edges away from an airport that the airline regularly serves.

Given a graph with V vertices representing all the airports, the subset W of V which are served by the airline, the distance $w(u, v)$ for each pair of adjacent airports u, v, and a base airport s, give an algorithm which finds the shortest distance from s to all other airports, with the airports along the path never more than 3 edges from an airport in W.

Problem 5. Tree Searches [20 points] (3 parts)

In this problem we consider doing a depth first search of a perfect binary search tree B. In a perfect binary search tree a node p can have either 2 or 0 children (but not just one child) with the usual requirement that any node in the left subtree of p is less than p and node in the right subtree is greater than p. In addition, all nodes with no children (leaves) must be at the same level of the tree. To make B into a directed graph, we consider the nodes of B to be the vertices of the graph. For each node p, we draw a directed edge from p to its left child and from p to its right child. An example of a perfect binary search tree represented as a graph is shown in Figure 1.

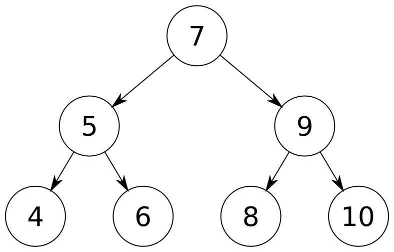

Figure 1: An example of a perfect binary search tree represented as a directed graph.

(a) [6 points] We structure our adjacency function such that at a node p, we first run DFS-VISIT on the left child of p and then on the right child. When we have finished expanding a node (i.e. just before we return from DFS-VISIT), we print the node. What is the first node printed? What is the last node printed? Give a short defense of your answer.

(b) [7 points] Does DFS print out the nodes of the tree in increasing or decreasing order? If yes, give a proof. If no, give a small counter example where the algorithm fails to print out the nodes in increasing or decreasing order and show the output of DFS on your example.

(c) [7 points] Recall that usually when doing depth first search, we use the *parent* structure to keep track of which vertices have been visited. During the search, if a vertex v is in *parent*, the search will not run DFS-VISIT(v) again. Aspen Tu declares that *parent* is unnecessary when doing a DFS of B. She says that whenever the algorithm checks if a vertex v is in *parent*, the answer is always false. Do you agree with Aspen? If you do, prove that she is correct. If you do not, give a small counter-example where a depth first search through B will see a vertex twice. Remember, B is a directed graph.

Problem 6. Computing Minimum Assembly Time [20 points]

As you might have heard, NASA is planning on deploying a new generation of space shuttles. Part of this project is creating a schedule according to which the prototype of the space shuttle will be assembled.

The assembly is broken down into atomic actions – called *jobs* – that have to be performed to build the prototype. Each job has a *processing time* and a (possibly empty) set of *required jobs* that need to be completed before this job can start – we will refer to this set as *precedence constraint*. Given such specification, we call an assembly schedule *valid* if it completes all the jobs and all the precedence constraints are satisfied.

Now, as the plan of the whole undertaking is being finalized, NASA has to compute the *minimum assembly time* of the prototype. This time is defined as the minimum, taken over all the valid assembly schedules, of the time that passes since the processing of the first scheduled job starts until the processing of the last job finishes. (Note that we allow jobs to be processed in parallel, as long as their precedence constraints are satisfied.)

As the prototype assembly is an immensely complex task, can you help NASA by designing an algorithm that computes the minimum assembly time efficiently? Prove the correctness of your algorithm and analyze its running time in terms of the number of jobs n and the total length of the required jobs lists m.

Formally, the assembly is presented as a list of n jobs $J_1, \ldots, J_n$, and each job J has a specified processing time, and the set of required jobs. We assume that there always is at least one valid assembly schedule corresponding to the given specification.

Example:

Job:	Processing time:	Required jobs:
J_1	1	$\{J_6, J_7\}$
J_2	6	$\emptyset$
J_3	4	$\{J_2, J_5\}$
J_4	2	$\{J_2, J_3\}$
J_5	3	$\emptyset$
J_6	5	$\emptyset$
J_7	7	$\emptyset$

Here, $n = 7$ and $m = 6$.

Solution: The minimum assembly time is 12.

(The corresponding schedule starts jobs J_2, J_5, J_6, J_7 at time 0, J_3 at time 6, J_1 at time 7, and J_4 at time 10.)

SCRATCH PAPER

SCRATCH PAPER