Note for Fall 2025: This is based on an old exam from a previous quarter; it may have covered slightly different things and had different HW problems, which would make these problems differently easy or hard than they are this quarter. But they are all good practice problems!

Instructions

- Answer all of the questions as well as you can. You have three hours.
- The exam is **non-collaborative**; you must complete it on your own.
- This exam is **closed-book**, except for **up to three double-sided sheets of paper** that you have prepared ahead of time. You can have anything you want written on these sheets of paper.

General Advice

- If you get stuck on a question or a part, move on and come back to it later. The questions on this exam have a wide range of difficulty, and you can do well on the exam even if you don't get a few questions.
- Pay attention to the point values. Don't spend too much time on questions that are not worth a lot of points.
- There are **105** + **3** (bonus) total points on this practice exam. There are **five problems** across **17 pages**.

Name and SUNet ID (please print clearly):

____

1. **(20 pt.) [Multiple Choice!]** For each of the parts below, clearly shade in **all** of the answers that are correct:

●	○	⊘ Ø ☺
Filled in means: "This is a true statement"	Empty means: "This is **not** true statement"	Anything else may be marked as incorrect!

Unless stated otherwise, we are always referring to worst-case analysis guarantees. Do not make any assumptions that are not stated in the problem.

Grading note: Each of the "main" answers is worth **one point**. The "None of the above" option on its own is not worth any points, it's just there so that you can register that you intentionally didn't select anything. If you select "None of the above" and any other answer, we will ignore your "None of the above". Ambiguously filled in answers will be marked as incorrect.

[**We are expecting:** *For each part, clearly filled-in answers. No justification is required or will be considered when grading. Ambiguously filled-in answers will be marked as incorrect.*]

(a) **(4 pt.)** Which of the following quantities are $O(n^2)$? Fill in all that apply.

- ○ $g(n) = 2^{\log_4 n}$
- ○ $T(n)$, where $T(n) = 2T(n-1) + 1$ for $n \geq 1$, and $T(0) = 1$.
- ○ The worst-case running time of QuickSort
- ○ The maximum number of items that can be inserted into a Red-Black tree before its height might (in the worst-case) exceed $100 \log n$
- ○ None of the above.

(b) **(4 pt.)** Recall the 0/1 Knapsack problem from lecture, where the objective is to maximize the value of the knapsack, given that we have a single copy of each of n items. Which of the following are true? Fill in all that apply.

- ○ The optimal solution can always be obtained using a greedy algorithm that greedily takes the items with the best value-to-weight ratio.
- ○ Let $K[k]$ denote the maximum value you can fit into a knapsack of capacity k. Then there is an $O(n)$-time DP solution to 0/1 Knapsack that uses the subproblems $K[k]$.
- ○ Any correct DP solution to this problem *must* be implemented in a top-down way.
- ○ Any correct DP solution to this problem *must* be implemented in a bottom-up way.
- ○ None of the above.

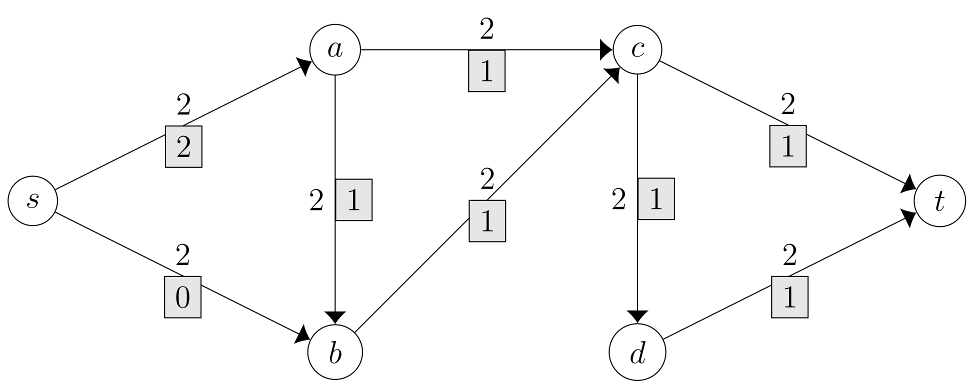
(c) **(4 pt.)** Let $G = (V, E)$ be a weighted directed graph. The shortest path from a node $s \in V$ to a node $t \in V$ will remain unchanged if:

- ◯ Each edge weight $w(v, u)$ is replaced by $C \cdot w(v, u)$ for a constant $C > 0$.
- ◯ Each edge weight $w(v, u)$ is replace by $w(v, u) + C$ for a constant $C > 0$.
- ◯ Each edge weight $w(v, u)$ is replace by $w(v, u) - C$ for a constant $C > 0$.
- ◯ Each edge weight $w(v, u)$ is replace by $w(v, u)/C$ for a constant $C > 0$.
- ◯ None of the above

(d) **(4 pt.)** Which of the following algorithms runs in $O(n^2)$ time? (Note: A fully-connected graph is a graph in which there is an edge between every pair of nodes.)

- ◯ Bellman-Ford on a fully-connected directed graph with n nodes.
- ◯ Kosaraju's algorithm on a fully-connected graph with n nodes.
- ◯ Floyd-Warshall on a connected graph with n nodes, so that each node has degree exactly 3.
- ◯ Dijkstra's algorithm (implemented with a Red-Black tree) on a directed acyclic graph with n nodes and positive edge weights, so that each node has up to 7 outgoing edges.
- ◯ None of the above.

(e) **(4 pt.)** Consider the graph G given below, where the labels without boxes are capacities and the labels with boxes are flows in a flow from s to t. Which of the following are true?



- ◯ $s \to a \to c \to t$ is an augmenting path from s to t.
- ◯ $s \to b \to a \to c \to t$ is an augmenting path from s to t.
- ◯ The flow shown here is a maximum flow.
- ◯ The minimum cut in this graph has value 4.
- ◯ None of the above.

2. **(20 pt.) [True or False?]** For each of the parts below, answer either True or False, and give a few sentences of explanation.

   [**We are expecting:** *Your answer (True or False) clearly stated, as well as a few sentences of justification. Ambiguous answers (e.g., where both True and False appear on the page) will be marked as incorrect.*]

   (a) **(4 pt.)** True or False (and explain): It is possible to detect whether or not a graph is bipartite in time $O(n + m)$.

   (b) **(4 pt.)** True or False (and explain): The SCCs returned by Kosaraju's algorithm can be different based on which node the algorithm starts at.

   (c) **(4 pt.)** Suppose that you are working on a "Maps" app for the Stanford campus. You are tasked with developing an algorithm to find the fastest route from any point A on campus to any other point B. To model the problem, you make a graph that represents all of the locations on campus, and an estimate of the time it takes to get between any points that are directly connected by a road. To get the estimates, you ride your bike between all the pairs of vertices on a sunny Saturday afternoon and time yourself. After you've generated this weighted graph, you run Dijkstra's algorithm to find shortest paths.

   True or false (and explain): the scenario above involves *idealization*, as we defined it in the Embedded EthiCS lectures.

More parts on next page

(d) **(4 pt.)** Consider the following algorithm, which purports to find a minimum spanning tree in an unweighted, undirected graph G:

- Maintain a set $\mathcal{C}$ of "components", initialized to $\mathcal{C} = \{\{v\} : v \in V\}$.
- Maintain a set of edges S, which we initialize to the empty set $\emptyset$.
- Until $\mathcal{C}$ consists of just one big component:
  - Choose an arbitrary component $C \in \mathcal{C}$.
  - Let $e = \{u, v\}$ be a minimum-weight edge in E so that $u \in C$, $v \notin C$. Let C' be the component in $\mathcal{C}$ that contains v.
  - Add e to S, and merge C and C' in $\mathcal{C}$.
- Return S.

Note that this algorithm is similar to Kruskal's algorithm, except that instead of picking a minimum weight edge crossing *any* two components to add to S, we pick a minimum weight edge coming out of an *arbitrary* component.

True or False (and explain): this algorithm correctly returns a minimum spanning tree.

another part on next page!

(e) **(4 pt.)** Suppose that $a_1, \ldots, a_k$ are positive integers. For a positive integer x and $j \leq k$, let $D(x, j)$ denote the number of ways to write x as the sum of numbers in $\{a_1, \ldots, a_j\}$, where each number is used at most once and order doesn't matter.

> **For example**, if $a_1 = 1, a_2 = 2, a_3 = 3$ and $a_4 = 4$, then there are two ways to make $x = 5$ out of $\{a_1, a_2, a_3, a_4\}$, namely $5 = 2 + 3 = 1 + 4$. Thus, $D(5, 4) = 2$.

You want to use the subproblems $D(x, j)$ to design a dynamic programming algorithm to compute $D(x, k)$, the number of ways to write x as a sum of the numbers $a_1, \ldots, a_k$.

True or False (and explain): The following relationship is correct.

$$D(x, j) = D(x - a_j, j - 1) + D(x, j - 1)$$

If your answer is True, explain why; if it is False, explain what's wrong and how to fix it. (Don't worry about base cases).

3. **(20 + 3 BONUS pt.) [How to do it?]** For each of the following, describe how to accomplish it. You may use any result/algorithm from class as a black box, but you must state clearly how you are using it (e.g., what are the inputs). If you modify a result/algorithm from class, clearly state how you would modify it.

   We have done the first one for you to give you an idea of what we are expecting.

   [**We are expecting:** *For each, a clear paragraph explaining how you would do it, following the guidelines above. You may use pseudocode if it helps you be clear, but it is not required. An explanation about why your approach is correct may be considered for partial credit, but is not required.*]

   (-) **(0 pt.)** **(Example)** Given a directed, weighted graph G with n vertices and m edges, possibly with negative edge weights, detect whether or not there is a negative cycle in G in time $O(n^3)$.

   Answer: Run the Floyd-Warshall algorithm for n steps, and suppose that $D^{(n)}$ is our final array. For each vertex v, check to see if $D^{(n)}[v, v] < 0$. If you find such a v, output "negative cycle!" If there is no such v, output "No negative cycle!"

   Expanation for partial credit, just in case: This works because we showed in class that $D^{(n)}[v, v]$ is the length of the shortest path between v and v, using the vertices from $1, \ldots, n$. So $D^{(n)}[v, v]$ is negative if and only if there is a negative cycle containing v.

   (a) **(5 pt.)** Suppose that there are n cities, some of which are connected by roads. Each road (say from city A to city B) takes one hour to traverse, and costs $w(A, B)$ dollars in tolls. Find the cost of the cheapest path from city S to city T that takes at most four hours to traverse, or return `None` if no such path exists.

More parts on next page!

(b) **(5 pt.)** Suppose that users on a social media platform are represented by a directed graph $G = (V, E)$ with n nodes and m edges. Users are the nodes. If there is an edge $(a, b) \in E$ from a to b, it means that user b follows user a on the social media platform. Say that b is *downstream* of a if there is a directed path from a to b in G is *downstream* of a if there is a directed path from a to b in G.

You create a new account and haven't followed anyone yet. Given G, give an algorithm that will find the smallest set of users you can follow to make sure that you are downstream of all users on the platform.

Your algorithm should run in $O(m + n)$ time.

(c) **(5 pt.)** Let $G = (V, E)$ be a directed weighted graph, possibly with negative weights, and vertex set $V = [v_1, v_2, \cdots, v_n]$. The *all-pairs shortest path matrix* (APSPM) of G is an $n \times n$ matrix A where $A[i, j]$ is the shortest path distance in G between v_i and v_j.

Given G, its APSPM A, and a new edge $e = (v_a, v_b) \notin E$ with weight w, find the APSPM of G after adding e to E. Assume that there no negative cycles before or after e is added. Your algorithm should run in $O(n^2)$ time.

More parts on next page!

(d) **(5 pt.)** Suppose there are N students and N classes. Each student i has a set S_i of classes that they are interested in taking. Suppose that each student can take at most 4 classes, and each class can seat at most 30 students. Given the lists S_i, in time at most $O(N^5)$, find a way to match students to classes, so that:

- No student is in more than 4 classes and no class has more than 30 students.
- No student is in a class they aren't interested in.
- The assignment has as many student-class matches as possible.

(e) **(3 BONUS pt.)** Let $G = (V, E)$ be a directed unweighted graph. We say that G is "kind-of-connected" if for every $u, v \in G$, *either* there is a path from u to v, *or* there is a path from v to u (or possibly both). Given an algorithm that determines whether or not G is kind-of-connected, in time $O(n + m)$.

4. **(25 pt.) [Greedy Algorithms!]**

There are n final exams today at Stanford; exam i is scheduled to begin at time a_i and end at time b_i. Two exams which overlap cannot be administered in the same classroom; two exams i and j are defined to be *overlapping* if $[a_i, b_i] \cap [a_j, b_j] \neq \emptyset$ (including if $b_i = a_j$, so one starts exactly at the time that the other ends). Consider the following problem.

Input: Arrays A and B of length n so that $A[i] = a_i$ and $B[i] = b_i$.

Output: The smallest number of classrooms necessary to schedule all of the exams, and an optimal assignment of exams to classrooms.

For example: Suppose there are three exams, with start and finish times as given below:

i	1	2	3
a_i	12pm	4pm	2pm
b_i	3pm	6pm	5pm

Then the exams can be scheduled in two rooms; Exam 1 and Exam 2 can be scheduled in Room 1 and Exam 3 can be scheduled in Room 2.

(a) **(10 pt.)** Design a **greedy algorithm** that solves the following problem. Your algorithm should run in time $O(n \log(n) + nk)$, where k is the minimum number of classrooms needed.

[**We are expecting:** *Pseudocode* ***AND*** *a short English description, as well as a short justification of the running time. You do not need to prove that your algorithm is correct (yet).*]

More space on next page!

More space for your greedy algorithm!

Another part on next page!

(b) **(15 pt.)** Prove formally that your greedy algorithm from part (a) is correct.

[**We are expecting:** *A formal proof. If you do a proof by induction, be sure to clearly state your inductive hypothesis, base case, inductive step, and conclusion.*]

5. **(20 pt.) [Dynamic Programming!]** Suppose you have n coins with distinct integer values $c_0, c_1, \ldots, c_{n-1}$, so that $c_i > 0$ for all i. In this problem you will design a **dynamic programming** algorithm which takes as input the values $c_0, \ldots, c_{n-1}$, and an integer $k \geq 0$, and outputs the number of ways to divide the coins into two piles so that both piles have total value at least k. Your algorithm should run in time $O(nk^2)$.

> **For example**, if $n = 4$ and $k = 4$, and the four coins have values $c_0 = 1, c_1 = 2, c_3 = 4, c_4 = 5$, then the algorithm should output 8, since there are eight ways to split up the coins in to two piles, where each pile has total value at least 4: $\{1, 2, 4\}$ and $\{5\}$; $\{1, 2, 5\}$ and $\{4\}$; $\{1, 4\}$ and $\{2, 5\}$; $\{1, 5\}$ and $\{2, 4\}$; and then the same things again but swapping the piles.

(a) **(10 pt.)** What are the sub-problems you will use? What is the recursive relationship between the sub-problems, and what base cases do they satisfy?

[**We are expecting:** *A clear definition of your sub-problems, the recursive relationship that they satisfy, and a complete set of base cases. You should also give a clear explanation of why your recursive relationship and base cases hold.*]

Another part on next page.

(b) **(5 pt.)** Write pseudocode for your algorithm.

[**We are expecting:** *Just pseudocode. You do not need to explain what it is doing or why it works, but it should use the sub-problems you defined in the previous part.*]

(c) **(5 pt.)** Now suppose that the coin values and the value k are not necessarily integers (but are still positive). Would your algorithm from part (b) still return the correct answer? If not, how would you fix it so that it would return the correct answer? Would the running time still be $O(nk^2)$?

[**We are expecting:** *The following things:*

- *Whether your algorithm from (b) would still work and an explanation.*
- *Pseudocode* ***OR*** *a high-level description of how to fix it if not.*
- *Whether or not the running time would still be $O(nk^2)$, and why or why not.*

]

This is the end of the exam!

This is the end of the exam! You can use this page for extra work on any problem. **Keep this page attached** to the exam packet (whether or not you use it), and if you want extra work on this page to be graded, clearly label which question your extra work is for.

This page is for extra work on any problem. **Keep this page attached** to the exam packet (whether or not you use it), and if you want extra work on this page to be graded, clearly label which question your extra work is for.

This page is for extra work on any problem. **Keep this page attached** to the exam packet (whether or not you use it), and if you want extra work on this page to be graded, clearly label which question your extra work is for.

This page is for extra work on any problem. **Keep this page attached** to the exam packet (whether or not you use it), and if you want extra work on this page to be graded, clearly label which question your extra work is for.