

Answer the questions in the boxes provided on the question sheets. If you run out of room for an answer, add a page to the end of the document.

Name: _____

Wisc id: _____

Dynamic Programming

Do **NOT** write pseudocode when describing your dynamic programs. Rather give the Bellman Equation, describe the matrix, its axis and how to derive the desired solution from it.

1. Kleinberg, Jon. *Algorithm Design* (p.313 q.2).

Suppose you are managing a consulting team and each week you have to choose one of two jobs for your team to undertake. The two jobs available to you each week are a low-stress job and a high-stress job.

For week i , if you choose the low-stress job, you get paid ℓ_i dollars and, if you choose the high-stress job, you get paid h_i dollars. The difference with a high-stress job is that you can only schedule a high-stress job in week i if you have no job scheduled in week $i - 1$.

Given a sequence of n weeks, determine the schedule of maximum profit. The input is two sequences: $L := \langle \ell_1, \ell_2, \dots, \ell_n \rangle$ and $H := \langle h_1, h_2, \dots, h_n \rangle$ containing the (positive) value of the low and high jobs for each week. For Week 1, assume that you are able to schedule a high-stress job.

- (a) Show that the following algorithm does not correctly solve this problem.

Algorithm: JOBSEQUENCE

Input : The low (L) and high (H) stress jobs.

Output: The jobs to schedule for the n weeks

for *Each week* i **do**

if $h_{i+1} > \ell_i + \ell_{i+1}$ **then**

 Output "Week i: no job"

 Output "Week i+1: high-stress job"

 Continue with week $i+2$

else

 Output "Week i: low-stress job"

 Continue with week $i+1$

end

end

- (b) Give an efficient algorithm that takes in the sequences L and H and outputs the greatest possible profit.

- (c) Prove that your algorithm in part (c) is correct.

2. Kleinberg, Jon. *Algorithm Design* (p.315 q.4).

Suppose you're running a small consulting company. You have clients in New York and clients in San Francisco. Each month you can be physically located in either New York or San Francisco, and the overall operating costs depend on the demands of your clients in a given month.

Given a sequence of n months, determine the work schedule that minimizes the operating costs, knowing that moving between locations from month i to month $i + 1$ incurs a fixed moving cost of M . The input consists of two sequences N and S consisting of the operating costs when based in New York and San Francisco, respectively. For month 1, you can start in either city without a moving cost.

- (a) Give an example of an instance where it is optimal to move at least 3 times. Explain where and why the optimal must move.

- (b) Show that the following algorithm does not correctly solve this problem.

Algorithm: WORKLOCSEQ

Input : The NY (N) and SF (S) operating costs.

Output: The locations to work the n months

```
for Each month  $i$  do
    if  $N_i < S_i$  then
        | Output "Month  $i$ : NY"
    else
        | Output "Month  $i$ : SF"
    end
end
end
```

- (c) Give an efficient algorithm that takes in the sequences N and S and outputs the value of the optimal solution.

- (d) Prove that your algorithm in part (c) is correct.

3. Kleinberg, Jon. *Algorithm Design* (p.333, q.26).

Consider the following inventory problem. You are running a company that sells trucks and predictions tell you the quantity of sales to expect over the next n months. Let d_i denote the number of sales you expect in month i . We'll assume that all sales happen at the beginning of the month, and trucks that are not sold are stored until the beginning of the next month. You can store at most s trucks, and it costs c to store a single truck for a month. You receive shipments of trucks by placing orders for them, and there is a fixed ordering fee k each time you place an order (regardless of the number of trucks you order). You start out with no trucks. The problem is to design an algorithm that decides how to place orders so that you satisfy all the demands $\{d_i\}$, and minimize the costs. In summary:

- There are two parts to the cost: (1) storage cost of c for every truck on hand; and (2) ordering fees of k for every order placed.
 - In each month, you need enough trucks to satisfy the demand d_i , but the number left over after satisfying the demand for the month should not exceed the inventory limit s .
- (a) Give a recursive algorithm that takes in s , c , k , and the sequence $\{d_i\}$, and outputs the minimum cost. (The algorithm does not need to be efficient.)

- (b) Give an algorithm in time that is polynomial in n and s for the same problem.

- (c) Prove that your algorithm in part (b) is correct.

4. Alice and Bob are playing another coin game. This time, there are three stacks of n coins: A , B , C . Starting with Alice, each player takes turns taking a coin from the top of a stack – they may choose any nonempty stack, but they must only take the top coin in that stack. The coins have different values. From bottom to top, the coins in stack A have values a_1, \dots, a_n . Similarly, the coins in stack B have values b_1, \dots, b_n , and the coins in stack C have values c_1, \dots, c_n . Both players try to play optimally in order to maximize the total value of their coins.
- (a) Give an algorithm that takes the sequences a_1, \dots, a_n , b_1, \dots, b_n , c_1, \dots, c_n , and outputs the maximum total value of coins that Alice can take. The runtime should be polynomial in n .

- (b) Prove the correctness of your algorithm in part (a).

5. Kleinberg, Jon. *Algorithm Design* (p. 327, q. 16).

In a hierarchical organization, each person (except the ranking officer) reports to a unique superior officer. The reporting hierarchy can be described by a tree T , rooted at the ranking officer, in which each other node v has a parent node u equal to his or her superior officer. Conversely, we will call v a direct subordinate of u .

Consider the following method of spreading news through the organization.

- The ranking officer first calls each of her direct subordinates, one at a time.
- As soon as each subordinate gets the phone call, he or she must notify each of his or her direct subordinates, one at a time.
- The process continues this way until everyone has been notified.

Note that each person in this process can only call *direct* subordinates on the phone.

We can picture this process as being divided into rounds. In one round, each person who has already heard the news can call one of his or her direct subordinates on the phone. The number of rounds it takes for everyone to be notified depends on the sequence in which each person calls their direct subordinates.

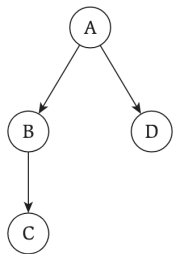


Figure 1: A hierarchy with four people. The fastest broadcast scheme is for A to call B in the first round. In the second round, A calls D and B calls C. If A were to call D first, then C could not learn the news until the third round.

Give an efficient algorithm that determines the minimum number of rounds needed for everyone to be notified, and outputs a sequence of phone calls that achieves this minimum number of rounds by answering the following:


- (a) Give a recursive algorithm. (The algorithm does not need to be efficient)

- (b) Give an efficient dynamic programming algorithm.

- (c) Prove that the algorithm in part (b) is correct.

6. Consider the following problem: you are provided with a two dimensional matrix M (dimensions, say, $m \times n$). Each entry of the matrix is either a **1** or a **0**. You are tasked with finding the total number of square sub-matrices of M with all **1**s. Give an $O(mn)$ algorithm to arrive at this total count by answering the following:

(a) Give a recursive algorithm. (The algorithm does not need to be efficient)



(b) Give an efficient dynamic programming algorithm.



- (c) Prove that the algorithm in part (b) is correct.

- (d) Furthermore, how would you count the total number of square sub-matrices of M with all 0s?

7. Kleinberg, Jon. *Algorithm Design* (p. 329, q. 19).

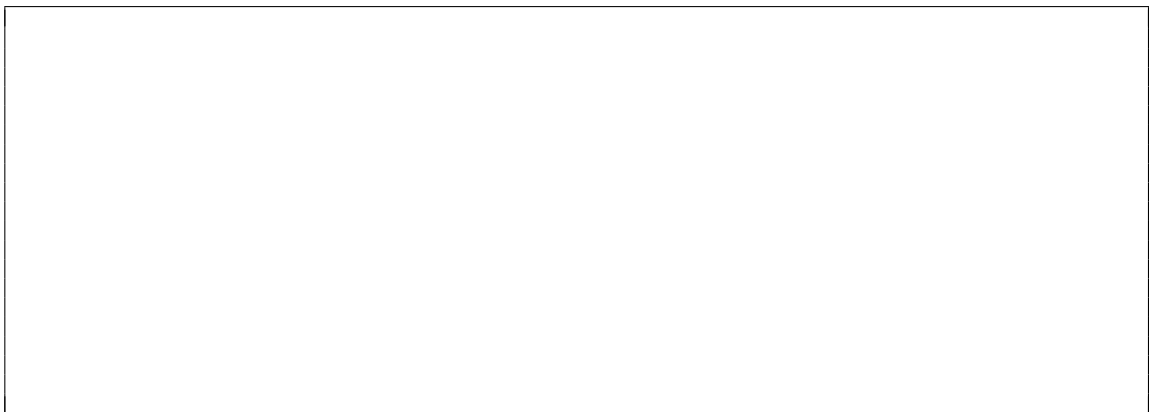
String x' is a *repetition* of x if it is a prefix of x^k (k copies of x concatenated together) for some integer k . So $x' = 10110110110$ is a repetition of $x = 101$. We say that a string s is an *interleaving* of x and y if its symbols can be partitioned into two (not necessarily contiguous) subsequences x' and y' , so that x' is a repetition of x and y' is a repetition of y . For example, if $x = 101$ and $y = 00$, then $s = 100010010$ is an interleaving of x and y , since characters 1, 2, 5, 8, 9 form 10110—a repetition of x —and the remaining characters 3, 4, 6, 7 form 0000—a repetition of y .

Give an efficient algorithm that takes strings s , x , and y and decides if s is an interleaving of x and y by answering the following:

- (a) Give a recursive algorithm. (The algorithm does not need to be efficient)



- (b) Give an efficient dynamic programming algorithm.



(c) Prove that the algorithm in part (b) is correct.

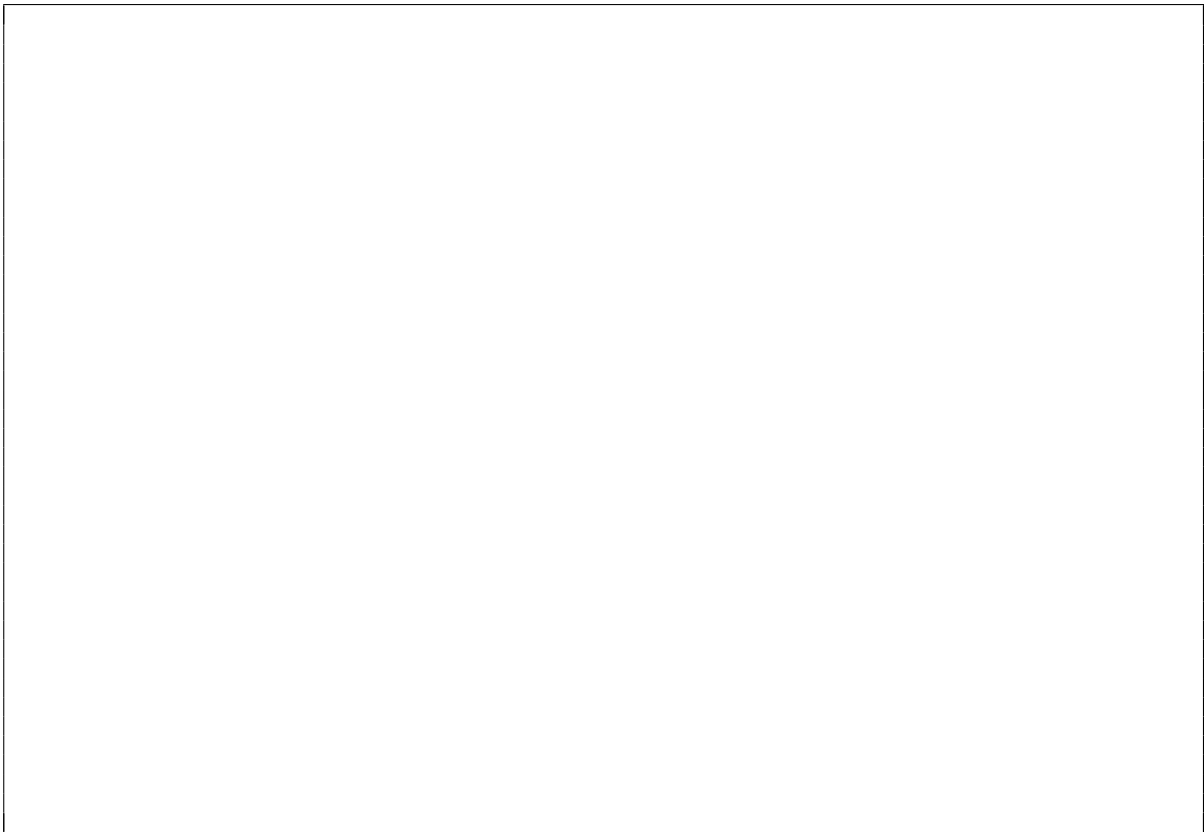


8. Kleinberg, Jon. *Algorithm Design* (p. 330, q. 22).

To assess how “well-connected” two nodes in a directed graph are, one can not only look at the length of the shortest path between them, but can also count the number of shortest paths.

This turns out to be a problem that can be solved efficiently, subject to some restrictions on the edge costs. Suppose we are given a directed graph $G = (V, E)$, with costs on the edges; the costs may be positive or negative, but every cycle in the graph has strictly positive cost. We are also given two nodes $v, w \in V$.

Give an efficient algorithm that computes the number of shortest $v - w$ paths in G . (The algorithm should not list all the paths; just the number suffices.)



9. The following is an instance of the Knapsack Problem. Before implementing the algorithm, run through the algorithm by hand on this instance. To answer this question, generate the table, indicate the maximum value, and recreate the subset of items.

item	weight	value
1	4	5
2	3	3
3	1	12
4	2	4

Capacity: 6

10. Coding Question 1: WIS

Implement the optimal algorithm for Weighted Interval Scheduling (for a definition of the problem, see the slides on Canvas) in either C, C++, C#, Java, Python, or Rust. Be efficient and implement it in $O(n^2)$ time, where n is the number of jobs. We saw this problem previously in HW3 Q2a, where we saw that there was no optimal greedy heuristic.

The input will start with a positive integer, giving the number of instances that follow. For each instance, there will be a positive integer, giving the number of jobs. For each job, there will be a trio of positive integers i , j and k , where $i < j$, and i is the start time, j is the end time, and k is the weight.

A sample input is the following:

```
2
1
1 4 5
3
1 2 1
3 4 2
2 6 4
```

The sample input has two instances. The first instance has one job to schedule with a start time of 1, an end time of 4, and a weight of 5. The second instance has 3 jobs.

The objective of the problem is to determine a schedule of non-overlapping intervals with maximum weight and to return this maximum weight. For each instance, your program should output the total weight of the intervals scheduled on a separate line. Each output line should be terminated by exactly one newline. The correct output to the sample input would be:

```
5
5
```

or, written with more explicit whitespace,

```
"5\n5\n"
```

Notes:

- Endpoints are exclusive, so it is okay to include a job ending at time t and a job starting at time t in the same schedule.
- In the third set of tests, some outputs will cause overflow on 32-bit signed integers.

11. Coding Question 2: Knapsack

Implement the algorithm for the Knapsack Problem in either C, C++, C#, Java, or Python. Be efficient and implement it in $O(nW)$ time, where n is the number of items and W is the capacity.

The input will start with an positive integer, giving the number of instances that follow. For each instance, there will two positive integers, representing the number of items and the capacity, followed by a list describing the items. For each item, there will be two nonnegative integers, representing the weight and value, respectively.

A sample input is the following:

```
2
1 3
4 100
3 4
1 2
3 3
2 4
```

The sample input has two instances. The first instance has one item and a capacity of 3. The item has weight 4 and value 100. The second instance has three items and a capacity of 4.

For each instance, your program should output the maximum possible value. The correct output to the sample input would be:

```
0
6
```