Homework 3
Due Friday, February 14 at 5:00 PM

“I always thought, it would be neat to make the Olympic team.”
-Michael Phelps

Handing In
To hand in a homework, go to the directory where your work is saved and run `cs0160_handin hwX` where X is the number of the homework. Make sure that your written work is saved as a .pdf file, and any Python problems are completed in the same directory or a subdirectory. You can re-handin any work by running the handin script again. We’ll only grade your most recent submission. To install stencil Python files for a homework, run `cs0160_install hwX`. You will lose points if you do not hand in your written work as a .pdf file.

1 Written Problems

Problem 3.1

Induction

1. Consider the statement $P(n) : (1 + a)^n \geq 1 + a(n - 1)$, given some $a \geq 0$. Use induction to prove $P(n)$ for all $n \geq 1$. While it may be a helpful reference, do not use the results of the proof in the [induction handout](on the Docs page) in this proof—the problems are similar but not the same.

2. Consider the statement $P(n) : (1 + a)^n \geq 1 + a(n + 1)$. Prove that $P(n - 1) \implies P(n)$ for $a > 0, n > 0$. Why doesn’t this prove that $P(n)$ holds for every $n > 0$?

Problem 3.2

Greedy vs. Dynamic Programs

![Figure 1: The input graph.](image)
Let $G$ be a graph (a collection of nodes and directed arcs) with $n$ nodes, labeled $1, 2, \ldots, n$, and arcs as drawn in Figure 1 for $n = 8$. There are two types of arcs:

1. For $1 \leq i \leq n - 1$, there is an arc $(i, i + 1)$.
2. For $1 \leq i \leq n - 2$, there is an arc $(i, i + 2)$.

Let $cost(i, j)$ be a function that takes as input two nodes connected by an edge in $G$ and returns the cost of the edge between them. See Figure 2 for an example of arcs labelled with their costs. Consider the problem of finding a low-cost path from node 1 to node $n$ (the cost of a path is the sum of the costs of the arcs on the path).

Give a description of your approach in sentences – you do NOT have to write pseudocode for this problem. (Questions to think about: Where will your algorithm start? What will it do at each node?) Your solution only needs to return the minimum cost, not the path itself.

1. Describe a greedy algorithm for solving the problem. A greedy algorithm is an algorithm that makes the best choice locally at each stage with the hope of finding a global optimum solution. Does your algorithm find the optimum for Figure 2? Will it always find the optimum? Prove or give a counterexample. (A counterexample would be a single instance of a graph in the specified form, and a set of weights for that graph (i.e., a picture like Figure 2 above), with the property that the path produced by the greedy algorithm is not optimal).

2. Describe a dynamic programming approach (like the one we used for solving seam-carving) to find an optimal solution. An algorithm that uses dynamic programming breaks the larger problem into smaller subproblems then uses the solutions to those subproblems to build up to solving the overall problem. See the lecture slides for more information.
2 Python Problems

Problem 3.3

Minimum Path

Design an algorithm using dynamic programming that returns the minimum number of steps it takes to reach the end of the array. Each entry in the array represents the maximum number of steps that can be taken from that position. For example, if the first element of the array is a 3, then up to 3 steps can be taken. If the first element of the array is a 0, then no steps can be taken and the race cannot be finished.

Example:

Let’s say you have the array [3, 1, 2, 0, 8]. The minimum path would be to take 2 steps from the start and then 2 steps to the end. Therefore, your minimum number of steps would be 2.

Notes:

- All you need to return is the minimum number of steps. Don’t worry about returning the actual path.
- The maximum number of steps available at each location is given by an element in the array, which means a step can be made up of less rungs. For example, in the array above, the first movement can be 3, 2, or 1 steps. Here, the best choice would be to take 2 steps, rather than the maximum of 3.
- You will always start at the first element and the top is reached once you reach the last element. It is not necessary to step past the last element.
- If the input is an empty array or None, then raise an InvalidInputException.
- If the input array is of length 1, then return 0 since you begin already at the last index.

Examples:

minsteps([3]) -> 0
minsteps([0]) -> 0
minsteps([1]) -> 0

- If the end cannot be reached you should return None.

Examples:

minsteps([0, 4, 5]) -> None
minsteps([1, 0, 0, 5]) -> None

Problem 3.4

The Jewelry Set Conundrum

Suppose Serena starts off with 180 dollars but wants to make more money by buying and selling a set of jewelry. She knows how much the value of the
jewelry set will fluctuate for \( n \) days, and she has written it down in an array of \( n \) numbers \( r[0] \ldots r[n - 1] \), where \( r[i] \) represents the change in value for day \( i \).

You are investing on behalf of Serena, and you can choose the days on which you buy and sell jewelry sets. You can only buy and sell them once, so you must hold onto them for a number of consecutive days. Your goal is to maximize your profits. Use Python to implement a linear time algorithm for finding the maximum amount of money you could have. Assume that jewelry sets don’t lose value on their own.

**Example:**

Let \( r = [-1, 2, 7, -8, 13, -2] \). To achieve maximum profits, you should buy on the second day \((i = 1)\) and sell on the fifth day \((i = 4)\). Therefore, when you sell out he will have \( 180 + 2 + 7 - 8 + 13 = 194 \) dollars. Your function should return 194.

**Notes:**

- Money is gained (or lost) on both the day you buy and sell the jewelry set. For example, if you buy on the 12th day and sell on the 14th and if \( r[11] = 10, r[12] = -3, r[13] = 4 \) then you will end up having \( 180 + 10 - 3 + 4 = 191 \) dollars.

- Your algorithm is not required to return the buying and selling days. You only need to return the maximum money that can be achieved by your investments.

- Remember that one option is to not buy at all, leaving you at 180 dollars. Another option is for you to buy and sell on the same day (your worth would be 180 + the dollar change for that day).

- If the input array is empty, you should return 180.

- if the input array is None, you should raise an `InvalidInputException`.