Problem Set 10

All parts are due on May 11, 2018 at 11PM. Please write your solutions in the LaTeX and Python templates provided. Aim for concise solutions; convoluted and obtuse descriptions might receive low marks, even when they are correct. Solutions should be submitted on the course website, and any code should be submitted for automated checking on `py.mit.edu/6.006`.

Please solve the following problems using dynamic programming. Be sure to define a set of subproblems, relate the subproblems recursively, argue the relation is acyclic, provide base cases, construct a solution from the subprob- lems, and analyze running time. Correct but inefficient polynomial time dynamic programs will be awarded significant partial credit.

Problem 10-1. [15 points] **Mountain Finding**

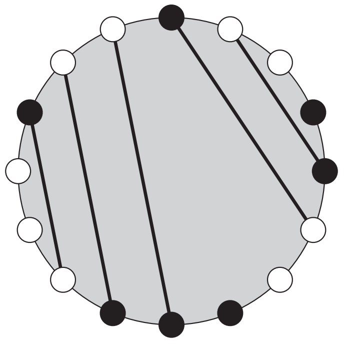
A sequence of integers is considered a **mountain** if it strictly monotonically increases to a maximum and then strictly monotonically decreases. For example, $A = (2, 3, 5, 4, 1)$ is a mountain, but $B = (3, 6, 8, 7, 9, 6, 4)$ is not. However B does contain a long subsequence $B' = (3, 6, 7, 9, 6, 4)$ which is a mountain. Given a sequence of n **unique** integers, describe an $O(n^2)$ time dynamic programming algorithm that finds its longest (not necessarily contiguous) subsequence that is also a mountain.

Problem 10-2. [15 points] **Professor Devkusha**

The nefarious Professor Devkusha enjoys playing games with his students. He calls you to his office, and places a row of n coins with positive values $c_1, c_2, \dots c_n$, onto a table in front of you, and proceeds to describe the rules of an asymmetric turn-based game. Professor Devkusha will first point to the space between any two coins, breaking the row into two **sections**. Then, you in turn may choose to remove any one coin on the table to obtain for yourself, but all other coins in the same section as the chosen coin will be removed from the table and discarded. These turns alternate until either zero or one coin remain on the table. Professor Devkusha will let you pass his class, only if you can obtain coins whose values total to at least k by the end of the game. Assuming that Professor Devkusha always plays optimally, describe an $O(n^3)$ time dynamic programming algorithm to determine whether you can pass his class.

Problem 10-3. [15 points] **Wafer Power**

A start-up is working on a new electronic circuit design for highly-parallel computing. Evenly-spaced along the perimeter of a circular wafer sits n ports for either a power source or a computing unit. Each computing unit needs energy from a power source, transferred between ports via a wire etched into the top surface of the wafer. However, if a computing unit is connected to a power source that is too close, the power can overload and destroy the circuit. Further, no two etched wires may cross each other. The circuit designer needs an automated way to evaluate the effectiveness of different designs, and has asked you for help. Given an arrangement of power sources and computing units plugged into the n ports, describe an $O(n^3)$ time dynamic programming algorithm to match computing units to power sources by etching non-crossing wires between them onto the surface of the wafer, in order to maximize the number of powered computing units, where wires may not connect two adjacent ports along the perimeter. Below is an example wafer, with non-crossing wires connecting computing units (white) to power sources (black).



Problem 10-4. [15 points] **Oh Charlie, Where Art Thou?**

A wealthy family, Alice, Bob, and their young son Charlie set off on a sailing journey around the world in their yacht. However, during their voyage the family encounters a massive storm, which throws Charlie from the boat, lost to the sea. Twenty years later, Alice and Bob are approached by a young man claiming to be Charlie, having survived for many years on a deserted island. Alice and Bob are excited at the prospect of reconnecting with their long lost son but are also skeptical of this man's claim. They order a DNA matching test from the genetic testing company 46AndThee. Given equal length n DNA sequences from Alice, Bob, and Charlie, the testing center uses the following criteria to test biological relations: Charlie is Alice and Bob's son if and only if Charlie's DNA can be partitioned into two (not necessarily contiguous) subsequences of equal length, such that one is a subsequence of Alice's DNA, and the other is a subsequence of Bob's DNA. For example, suppose Alice's DNA is `AATT` and Bob's DNA is `CCGG`. If Charlie's DNA were to be `CATG`, he would be matched as a son, as Charlie's DNA would contain disjoint subsequences CG and AT which are subsequences of Alice and Bob's DNA respectively. However, Charlie would be found to be an imposter if his DNA were `AGTC`. Describe an $O(n^4)$ time dynamic programming algorithm to determine whether Charlie is indeed the couple's son.

Problem 10-5. [40 points] **Winning Elections**

Jane 'The Paper' Dwonson is running for president of the Federated Districts of Algorithmia. Each district has a population and a designated electoral count. If greater than 50% of the population of a district votes for a candidate, that candidate receives the electoral count associated with that district (a winner-take-all system). Given a list of n districts in Algorithmia and their corresponding populations and electoral counts, Jane would like to determine the minimum number of votes she must win, in order to receive exactly c electoral counts, for a given value of c. For example, here is a list of districts, with associated populations and electoral counts:

District	Population	Electoral Count
Balaska	1056	30
Nine	1256	30
Bexastama	77	10
Twelve	1223	20
Dashington W.C.	454	20

To receive exactly $c = 40$ electoral counts, Jane must win at a minimum 568 votes, accomplished by winning districts Balaska and Bexastama, with 529 and 39 votes respectively.

(a) [10 points] Describe a dynamic programming algorithm to determine the minimum number of votes Jane needs to win, in order to receive exactly c electoral counts.

(b) [5 points] Briefly argue why a recursive top-down implementation of a dynamic programming algorithm might be preferable to a bottom-up implementation.

(c) [25 points] Write the Python function `min_votes(districts, c)` that implements your algorithm from part (a) using a **top-down** dynamic programming implementation. The input `districts` is a list of length n containing the information for districts, which are tuples consisting of the population and electoral count for the district. Your function should return the minimum number of people who must vote for Jane in order to obtain exactly c electoral counts, or output `None` if it is not possible to obtain the exact number. Submit your code online at `py.mit.edu/6.006`.