Dynamic Programming and Greedy Algorithms

1. Congratulations! You have successfully conquered Camelot, transforming the former battle-scarred kingdom with an anarcho-syndicalist commune, where citizens take turns to act as a sort of executive-officer-for-the-week, but with all the decisions of that officer ratified at a special bi-weekly meeting, by a simple majority in the case of purely internal affairs, but by a two-thirds majority, in the case of more major.

As a final symbolic act, you order the Round Table (surprisingly, an actual circular table) to be split into pizza-like wedges and distributed to the citizens of Camelot as trophies. Each citizen has submitted a request for an angular wedge of the table, specified by two angles—for example, Sir Robin the Brave might request the wedge from $23.17^\circ$ to $142.091^\circ$, and Sir Galahad the Pure might request the wedge from $358\frac{2}{3}^\circ$ to $7^\circ$. Each citizen will be happy if and only if they receive precisely the wedge that they requested. Unfortunately, some of these ranges overlap, so satisfying all the citizens' requests is simply impossible. Welcome to politics.

Describe and analyze an algorithm to find the maximum number of requests that can be satisfied.

[Hint: Circles are round. Your algorithm's output should not change if we add $17^\circ$ to every input angle, or in other words, if we rotate the table by $17^\circ$. No, this problem is not equivalent to a problem you saw in class.]

2. You and your friends decide to take a road trip, but before you leave, you need to figure out exactly how much money to bring for gasoline. Suppose you compile a list of all gas stations along your planned route, containing the following information:

- A sorted array $\text{Dist}[0..n]$, where $\text{Dist}[0] = 0$ and $\text{Dist}[i]$ is the number of miles from the beginning of your route to the $i$th gas station. Your route ends at the $n$th gas station.
- A second array $\text{Price}[1..n]$, where $\text{Price}[i]$ is the price of one gallon of gasoline at the $i$th gas station. (Unlike in real life, these prices do not change over time.)

You start the trip with a full tank of gas. Whenever you buy gas, you must completely fill your tank. Your car holds exactly 10 gallons of gas and travels exactly 25 miles per gallon; thus, starting with a full tank, you can travel exactly 250 miles before your car dies. Finally, $\text{Dist}[i+1] < \text{Dist}[i] + 250$ for every index $i$, so the trip is possible.

Describe and analyze an algorithm to determine the minimum amount of money you must spend on gasoline to guarantee that you can drive the entire route.

3. Recall that a palindrome is any string that is exactly the same as its reversal, like I, DAD, HANNAH, AIBOHPHOBIA (fear of palindromes), or the empty string.

Recall that a string $x$ is a subsequence of a string $y$ if we can obtain $x$ by deleting zero or more letters from $y$. A string $x$ is a supersequence of a string $y$ if we can obtain $x$ by inserting zero or more letters into $y$, or equivalently, if $y$ is a subsequence of $x$. For example, the string DAMPRAG is a subsequence of the string DYNAMICPROGRAMMING; symmetrically, DYNAMICPROGRAMMING is a supersequence of DAMPRAG.

We would only ask one of the following questions on a real exam.
(a) Describe and analyze an algorithm to find the length of the shortest supersequence of a given string that is also a palindrome.

For example, the 11-letter string \textbf{EHECADACEHE} is the shortest palindrome supersequence of HEADACHE, so given HEADACHE as input, your algorithm should return 11.

(b) Describe and analyze an algorithm to find the length of the longest subsequence of a given string that is also a palindrome.

For example, the 11-letter string \textbf{MHYMRORMYHM} is longest palindrome subsequence of MAHODYNAMICPROGRAMZLETMESHOWYOUTHENV, so given that string as input, your algorithm should return 11.

4. Suppose we are given an \( n \)-digit integer \( X \). Repeatedly remove one digit from either end of \( X \) (your choice) until no digits are left. The square-depth of \( X \) is the maximum number of perfect squares that you can see during this process. For example, the number 32492 has square-depth 3, by the following sequence of removals:

\[
32492 \rightarrow 3249 \rightarrow 324 \rightarrow 324 \rightarrow 324 \rightarrow 4.
\]

Describe and analyze an algorithm to compute the square-depth of a given integer \( X \), represented as an array \( X[1..n] \) of \( n \) decimal digits. Assume you have access to a subroutine \texttt{IsSquare} that determines whether a given \( k \)-digit number (represented by an array of digits) is a perfect square in \( O(k^2) \) time.

Minimum Spanning Trees and Shortest Paths

5. When there is more than one shortest path from one node \( s \) to another node \( t \), it is often convenient to choose a shortest path with the fewest edges; call this the best path from \( s \) to \( t \). Suppose we are given a directed graph \( G \) with positive edge weights and a source vertex \( s \) in \( G \). Describe and analyze an algorithm to compute best paths in \( G \) from \( s \) to every other vertex.

6. Let \( G = (V,E) \) be a connected directed graph with non-negative edge weights, let \( s \) and \( t \) be vertices of \( G \), and let \( H \) be a subgraph of \( G \) obtained by deleting some edges. Suppose we want to reinsert exactly one edge from \( G \) back into \( H \), so that the shortest path from \( s \) to \( t \) in the resulting graph is as short as possible. Describe and analyze an algorithm to choose the best edge to reinsert. For full credit, your algorithm should run in \( O(E \log V) \) time.

7. Let \( G \) be an undirected graph with weighted edges.

(a) Describe and analyze an algorithm to compute the maximum weight spanning tree of \( G \).

(b) A feedback edge set of \( G \) is a subset \( F \) of the edges such that every cycle in \( G \) contains at least one edge in \( F \). In other words, removing every edge in \( F \) makes \( G \) acyclic. Describe and analyze a fast algorithm to compute the minimum weight feedback edge set of \( G \).

\textbf{[Hint: Don’t reinvent the wheel!]}]

8. Consider a path between two vertices \( s \) and \( t \) in an undirected graph \( G \) with weighted edges. The bottleneck length of this path is the maximum weight of any edge in the path. The bottleneck
distance between s and t is the minimum bottleneck length of any path from s to t. (If s and t are in different components of G, the bottleneck distance between s and t is ∞.)

Describe an algorithm to compute the bottleneck distance between every pair of vertices in an arbitrary undirected weighted graph. Assume that no two edges have the same weight.

9. In this problem we will discover how you, too, can be employed by Wall Street and cause a major economic collapse! The arbitrage business is a money-making scheme that takes advantage of differences in currency exchange. In particular, suppose that 1 US dollar buys 120 Dogecoin; 1 Dogecoin buys 0.01 euros; and 1 euro buys 1.2 US dollars. Then, a trader starting with $1 can convert his money from dollars to Dogecoin, then from Dogecoin to euros, and finally from euros back to dollars, ending with $1.44! The cycle of currencies $ → D → € → $ is called an arbitrage cycle. Of course, finding and exploiting arbitrage cycles before the prices are corrected requires extremely fast algorithms.

Suppose n different currencies are traded in your currency market. You are given the matrix Rate[1..n, 1..n] of exchange rates between every pair of currencies; for each i and j, one unit of currency i can be traded directly for Rate[i, j] units of currency j. (Do not assume that Rate[i, j] · Rate[j, i] = 1.)

We would only ask one of the following questions on a real exam.

(a) Describe an algorithm that returns a two-dimensional array MaxRate[1..n, 1..n], where MaxRate[i, j] is the maximum amount of currency j that you can obtain by any sequence of trades, starting with one unit of currency i, assuming there are no arbitrage cycles.

(b) Describe an algorithm to determine whether the given matrix of currency exchange rates creates an arbitrage cycle.