Note: Your TA probably will not cover all the problems. This is totally fine, the discussion worksheets are deliberately made long so they can serve as a resource you can use to practice, reinforce, and build upon concepts discussed in lecture, readings, and the homework.

1 Carnival

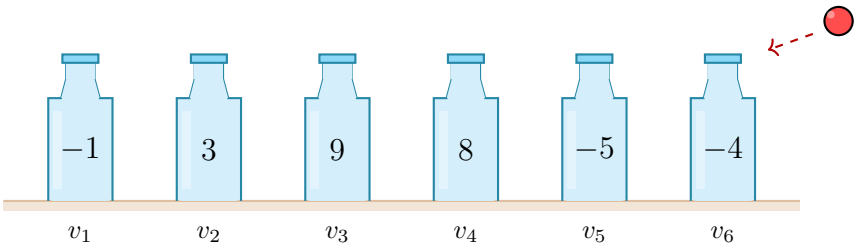
Playground: `https://gemini.google.com/share/d7225194933d`

You are throwing balls at a row of n bottles. Each bottle i has a value v_i written on it. On each throw, you have two options:

- Option 1 (Direct Hit): Hit bottle i directly. It falls over, and you earn v_i points.
- Option 2 (Gap Shot): Hit the gap between two adjacent bottles i and $i+1$. Both bottles fall over, and you earn $v_i \times v_{i+1}$ points.

Each bottle can participate in at most one throw. You may also choose to leave bottles standing. Your goal is to maximize your total score.

Example: Consider the following row of 6 bottles:



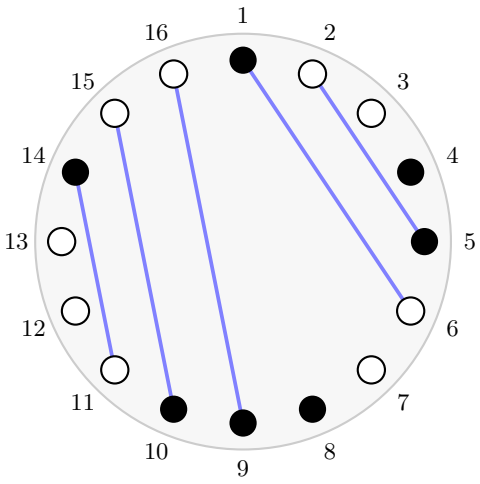
Come up with a dynamic programming algorithm to compute the maximum possible score. Define your subproblems, write the recurrence, specify the base cases, and analyze the runtime.

2 Circuit Design

A start-up is working on a new electronic circuit design for highly-parallel computing. Evenly spaced along the perimeter of a circular wafer sit n ports, each holding 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.

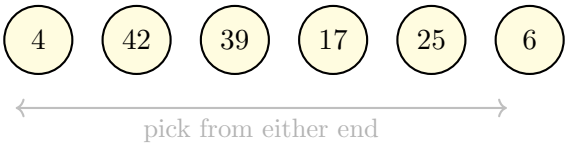
The input to your algorithm is a boolean array $A[1 \dots n]$ where $A[i]$ indicates whether port i is a power source or a computing unit. Describe an $O(n^3)$-time dynamic programming algorithm to match computing units to power sources by etching non-crossing wires between them onto the surface of the 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 $n = 16$ ports. Non-crossing wires connect computing units ($\circ$) to power sources ($\bullet$), powering 5 of the 9 computing units:



3 Alternating Coin Game

A row of n coins with values $V_1, V_2, \ldots, V_n$ is laid out on a table, where n is even. Two players take turns. In each turn, the current player selects either the first or the last coin from the row, removes it permanently, and receives the value of that coin. Player 1 goes first.

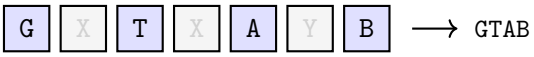


(a) Try playing the game on the row of coins shown above: [4, 42, 39, 17, 25, 6]. What is the best score any player can achieve? You can use: `https://gemini.google.com/share/1694f6ba260e` to play the game.

(b) Design a dynamic programming algorithm that computes the maximum total value Player 1 can guarantee, regardless of how Player 2 plays.

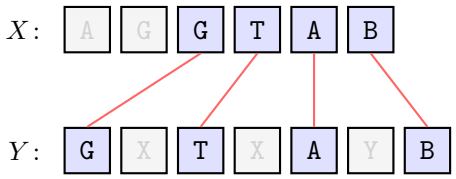
4 Longest Common Subsequence

A *subsequence* of a string is obtained by deleting zero or more characters without changing the order of the remaining characters. Crucially, the selected characters need not be contiguous. For example, `GTAB` is a subsequence of `GXTXAYB`:



Given two strings $X = x_1x_2 \cdots x_m$ and $Y = y_1y_2 \cdots y_n$, a *longest common subsequence* (LCS) is the longest string that is a subsequence of both X and Y.

Example: Consider $X =$ `AGGTAB` and $Y =$ `GXTXAYB`. The LCS is `GTAB` (length 4). Notice how the matched characters are spread across both strings, not necessarily next to each other:



Come up with a dynamic programming algorithm to compute the length of the longest common subsequence. Define your subproblems, write the recurrence, specify the base cases, and analyze the runtime.

