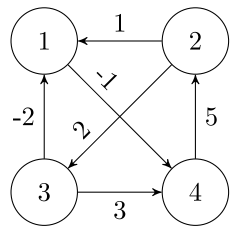
# Weekly Assignment 9:
# Dynamic Programming

1. Run the Floyd-Warshall algorithm on the following graph. Show $D^{(0)}$ and the matrix $D^{(k)}$ after each iteration.




2. How can we use the output of the Floyd-Warshall algorithm to detect the presence of a cycle with a negative weight? Justify your answer.

3. In dynamic programming, we put up a recursive equation for the solution to our problem, and implement this either bottom-up, or top-down using memoization. Briefly (using a few sentences max) explain what a top-down implementation is in this context, and the role of memoization.

4. Consider the following problem.

   The net income (positive or negative) of a company during the last $n$ years is represented as an array $\mathcal{I} = (i_1, i_2, \ldots, i_n)$ with $n \geq 1$. For a contiguous period $x \ldots y$ with $1 \leq x \leq y \leq n$, we define the accrued income as $AI = (i_x + i_{x+1} + \ldots + i_y)$. We are now interested in the best period of the history of the company, where the accrued income is maximal. For instance, the maximal accrued income (MAI) for $\mathcal{I} = (-5, 1, 3, -1, 10, -2, 0)$ is MAI $= 1 + 3 - 1 + 10 = 13$, with starting position 2 and ending position 5.

   1. Give a linear time algorithm which computes the MAI, based on dynamic programming. Specify the recursion equations on which your solution is based, explain why these equations are correct, describe a bottom-up implementation based on the recursion equations, and explain why the time complexity is linear.
   2. Extend your algorithm so that it also returns the starting and ending position of the MAI.

5. Let $M$ be a $m \times n$ matrix of natural numbers and let $C$ be a natural number. A $C$-path of $M$ is a sequence of the form $M[i_1, j_1], M[i_2, j_2], \ldots, M[i_l, j_l]$, with $l \geq 1$, such that:

   1. the sum of its elements is $C$;
   2. for any two consecutive elements $M[i_k, j_k]$ and $M[i_{k+1}, j_{k+1}]$, either $i_{k+1} = i_k + 1$ and $j_{k+1} = j_k$; or $i_{k+1} = i_k$ and $j_{k+1} = j_k + 1$.

In words, a $C$-path is a non-empty path of sum $C$ through the matrix: at each step it either goes one cell down, or one cell to the right.

We want to write a dynamic programming algorithm that computes the number of $C$-paths from $M[0,0]$ to $M[m,n]$. For example, for

$$M = \begin{array}{|c|c|c|} \hline 1 & 2 & 3 \\ \hline 4 & 6 & 5 \\ \hline 3 & 2 & 1 \\ \hline \end{array} \qquad C = 12$$

we have two such $C$-paths: $1, 2, 6, 2, 1$ and $1, 2, 3, 5, 1$.

(a) Let $P[i, j, k]$ be the number of $k$-paths from $M[0,0]$ to $M[i,j]$. Give recurrence equations that can be used to compute $P[i, j, k]$, and explain why these recurrences hold.

(b) Give a bottom-up dynamic programming algorithm that returns the number of $C$-paths from $M[0,0]$ to $M[m,n]$. Analyze the time complexity of your algorithm.

(c) Explain why in practice a top-down, recursive implementation (using memoization) might be faster.