Our take on Savitch’s algorithm is broken up into several parts. First, using a more “pythonic” approach to populating an n x n matrix, we take input as n and create an n x n matrix from this. The matrix is then initialized, and populated as an adjacency matrix with two built in diagonal paths. The algorithm will search from the top left to the bottom right. As such, Savitch’s algorithm is called with the matrix M as the first input, 0 as the second input, (n \* n) – 1 or the max distance as the third input, and the size of the matrix as the last input.

Savitch begins by appending the first predicate as the current first vertex (u), the current vertex to search for (v) and k to the stack. It then checks for the two base cases of either having the beginning and final vertices being the same or the edge connecting to the last vertex being true, or being 1 as it is in an adjacency matrix. For the recursive call, we search every midpoint w in the path, and call Savitch’s algorithm on both halves of the path, with k-1 vertices as the floor for the first recursive call, and k-1 vertices as the ceiling for the second recursive call. This returns true or false, and if true, it proceeds.