For both the memoized and dynamic programming versions, describe a scenario in which these would exhibit the worst-case time complexity and explain why this is the worst-case. Note that we are not referring to removing all seams but one. Rather, we are asking *what kind of input* (image) would cause the worst-case behavior. You do not need to do a full-blown big-O analysis.

The **memoized** implementation exhibits worst-case time complexity when the input image has **square** dimensions. This is because a square image maximizes the number of potential seams that can pass through it. Despite using memoization to reduce overlapping subproblems, the implementation follows a brute-force approach, requiring it to evaluate all possible seams in some capacity. Consequently, as the number of possible seams increases, the time complexity grows accordingly.

The **dynamic** approach will also exhibit worse-case time complexity on square images. The function calls the `best-seam-to` function once for each pixel in the image regardless of the dimensions of the image. However, the higher the average length of `valid-pixels` per function call (since the list is `fold`-ed over) the more computation that will have to be executed in total.