9. Given two strategies A and B, as well as the corresponding P&L of these strategies on each day. If one is going to be shut down, how to decide which one to shut?

In the actual trading, the decision of shutting down a strategy may depend on various different metrics, like alpha, beta, sharpe ratio, etc. Here we formulate the problem as, given two time series of performance metrics, testing whether one series is significantly better (not necessarily larger) than the other. The definition of “better” depends on the actual performance metrics we choose. Here we use P&L as an example. Since we have paired P&L time series (i.e., the ith elements from two time series of P&L are from the same day), we should take advantage of the pairing information when evaluating two strategies. We can process in two directions:

First, since we have paired P&L, we take the daily distance between the (scaled if needed) P&L of the two strategies. Here distance can be naïve difference, L1 norm, L2 norm, Lp norm or other distance measures. We can then use parametric or nonparametric test to see whether the distance time series is a stationary time series

Second, we can fit a common model that would reasonably describe each series separately. This might be an ARIMA model or a multiply-trended Regression Model with possible Level Shifts or a composite model integrating both memory (ARIMA) and dummy variables. This common model could be estimated globally and separately for each of the two series and then one could construct an F test to test the hypothesis of a common set of parameters.

Ref:

[1] Harvey, C. R., & Liu, Y. (2014). Evaluating trading strategies. Available at SSRN 2474755.

[2] Hamilton, J. Time series analysis.

14. Consider a chess board with each cell has a value on it, you can only walk right or down, find the path from up-left to down-right which has largest value.

This is a typical dynamic programming problem. Ref: <https://www.topcoder.com/community/data-science/data-science-tutorials/dynamic-programming-from-novice-to-advanced/>

First of all we have to find a state. The first thing that must be observed is that there are at most 2 ways we can come to a cell – from the left (if it’s not situated on the first column) and from the top (if it’s not situated on the most upper row). Thus to find the best solution for that cell, we have to have already found the best solutions for all of the cells from which we can arrive to the current cell.

From above, a recurrent relation can be easily obtained:  
**S[i][j]=A[i][j] + max(S[i-1][j], if i>0 ; S[i][j-1], if j>0)**(where **i** represents the row and **j** the column of the table , its left-upper corner having coordinates {0,0} ; and **A[i][j]** being the number of apples situated in cell**i,j**).

**S[i][j]**must be calculated by going first from left to right in each row and process the rows from top to bottom, or by going first from top to bottom in each column and process the columns from left to right.

Pseudocode:

For i = 0 to N - 1

For j = 0 to M - 1

S[i][j] = A[i][j] +

max(S[i][j-1], if j>0 ; S[i-1][j], if i>0 ; 0)

Output S[n-1][m-1]