1. **Programming Language**: Python
2. **Assumptions**:
   * The input matrix is well-formed with no irregularities such as mismatched row sizes or non-integer values.
   * Stock prices are all non-negative integers.
   * The input matrix will not be empty.
3. **Definitions**:
   * **matrix**: A list of lists in Python, where each sublist represents a stock and contains the prices of that stock for consecutive days.
   * **max\_profit\_info**: A tuple that holds the best transaction details including the stock index (1-indexed), the buy day (1-indexed), the sell day (1-indexed), and the maximum profit found.
   * **stock\_index**: An integer representing the index of the current stock in the iteration, starting from 0 for the first stock.
   * **prices**: A list of integers representing the prices of a stock over a series of days.
   * **max\_profit**: An integer representing the maximum profit found for a particular stock, initialized to 0 for each new stock in the iteration.
   * **buy\_day** and **sell\_day**: Integers representing the days on which buying and selling would result in the **max\_profit**, respectively. Initialized to 0 and updated within the nested loop when a profitable pair is found.
4. **Pseudocode:**

# Function to calculate the maximum profit by comparing all possible buy-sell pairs

Function find\_max\_profit\_brute\_force takes a matrix of stock prices:

# Initialize a tuple to store the best transaction details

Initialize max\_profit\_info to (0, 0, 0, 0) // This holds the stock index, buy day, sell day, and max profit

# Iterate over each stock using its index and price list

For each stock\_index and prices list in the matrix:

# Start with no profit as we haven't compared any prices yet

Initialize max\_profit for this stock to 0

# Default buy day is set to zero, to be updated when a profitable buy day is found

Initialize buy\_day to 0

# Default sell day is set to zero, to be updated when a profitable sell day is found

Initialize sell\_day to 0

# Nested loop to compare every possible buy-sell pair of days

For each day i in the range of prices:

# Start from the next day (i+1) since you cannot sell on the same day you buy

For each day j from i+1 to the end of prices:

# Calculate profit if you were to buy on day i and sell on day j

Calculate current\_profit as the difference between prices[j] (sell price) and prices[i] (buy price)

# Check if the calculated profit is greater than the previously recorded max profit

If current\_profit is greater than max\_profit:

# Update max\_profit with the new maximum

Set max\_profit to current\_profit

# Record the day you should buy to achieve this profit

Set buy\_day to i + 1 // Add 1 to convert from 0-indexed to 1-indexed format

# Record the day you should sell to achieve this profit

Set sell\_day to j + 1 // Add 1 for the same reason

# After evaluating all buy-sell pairs for this stock,

# check if the best profit from this stock beats the best profit from previous stocks

If max\_profit for the current stock is greater than the max\_profit stored in max\_profit\_info:

# Update max\_profit\_info with new best transaction details

Update max\_profit\_info with (stock\_index + 1, buy\_day, sell\_day, max\_profit)

# After checking all stocks, return the details of the transaction that yields the maximum profit

Return max\_profit\_info