Andres Torrado

Divide and Conquer Approach

The divide and conquer approach for maximizing stock trading profit involves a meticulous strategy:

Base Case Handling: Begin by defining base cases for recursion. When the sequence contains only one or two elements, compute the maximum profit directly.
Sequence Division: Divide the sequence of stock prices into two halves.
Recursive Exploration: Recursively explore the left and right halves to find the maximum profit.
Crossing Subproblem: Compute the maximum profit achievable by buying in the left half and selling in the right half (crossing subproblem).
Result Combination: Return the maximum of the profits obtained from steps 3 and 4.

Pseudo Code

```
maxProfit(prices, start, end):
   if start >= end:
      return 0
   if end - start == 1:
      return max(0, prices[end] - prices[start])
   mid = (start + end) / 2
   leftProfit = maxProfit(prices, start, mid)
   rightProfit = maxProfit(prices, mid + 1, end)
   crossProfit = maxCrossingProfit(prices, start, mid, end)
   return max(leftProfit, rightProfit, crossProfit)

maxCrossingProfit(prices, start, mid, end):
   leftMin = prices[mid]
   rightMax = prices[mid + 1]
   for i from mid downto start:
      leftMin = min(leftMin, prices[i])
   for i from mid + 1 to end:
      rightMax = max(rightMax, prices[i])
   return max(0, rightMax - leftMin)
```
Time Complexity Analysis

The time complexity of the divide and conquer approach is $O(n \log n)$, where n represents the length of the sequence. This complexity is derived from the recurrence equation $T(n) = 2T(n/2) + O(n)$.

Linear Time Divide and Conquer

Design

To achieve linear time complexity O(n) using divide and conquer, a modified approach is proposed:

Sequence Division: Divide the sequence into two halves.
Recursive Exploration: Recursively find the maximum profit in the left and right halves.
Profit Computation: Compute the maximum profit that can be obtained by buying in the left half and selling in the right half.
Result Combination: Return the maximum profit among the left, right, and cross profits.

Pseudo Code

```
// Function maxProfit as described above

function maxProfitLinear(prices, start, end):
    if end - start <= 2:
        return maxProfitBaseCase(prices, start, end)
    mid = (start + end) / 2
    leftResult = maxProfit(prices, start, mid)
    rightResult = maxProfit(prices, mid + 1, end)
    crossProfit = prices[rightResult.sellIndex] - prices[leftResult.buyIndex]
    if leftResult.maxProfit >= rightResult.maxProfit and leftResult.maxProfit >= crossProfit:
        return leftResult
    else if rightResult.maxProfit >= crossProfit:
        return rightResult
    else:
        // Combine left and right results
        return {
            buyIndex: leftResult.buyIndex,
            sellIndex: rightResult.sellIndex,
            maxProfit: crossProfit,
            // Update other indices as needed
        }

function maxProfitBaseCase(prices, start, end):
    if end - start == 1:
        // Handle base case with two elements
    else if end - start == 2:
        // Handle base case with three elements
```

Time Complexity Analysis

The linear time complexity O(n) is achieved by the modified divide and conquer approach. This efficiency results from a single pass through the sequence of prices.

Decrease and Conquer

Design

The decrease and conquer approach offers an alternative solution with linear time complexity O(n):

Price Tracking: Iterate through the sequence of prices, keeping track of the minimum price seen so far and the maximum profit achievable.
Profit Calculation: Compute the maximum profit that can be obtained by selling at the current price.
Result Return: Return the maximum profit calculated.

Pseudo Code

```
function maxProfitDecrease(prices):
  minPrice = prices[0]
  maxProfit = 0
  for i from 1 to prices.length - 1:
    minPrice = min(minPrice, prices[i])
    maxProfit = max(maxProfit, prices[i] - minPrice)
  return maxProfit
```

Time Complexity Analysis

The time complexity of O(n) is attained through a single pass through the sequence of prices, resulting in an efficient solution.

Conclusion

The divide and conquer strategy offers a robust framework for addressing the stock trading profit maximization problem efficiently. While the basic approach achieves O(n log n) time complexity, modifications enable linear time solutions. The decrease and conquer approach provides yet another avenue for achieving linear time complexity. These strategies offer versatile solutions tailored to the demands of stock trading optimization.