|  |  |
| --- | --- |
| **Best time to buy and Sell Stocks infinite in C++** | |
| #include <iostream>  #include <vector>  using namespace std;  class BestTimeToBuyAndSellStocksInfiniteTransactions {  public:  int maxProfit(vector<int>& prices) {  if (prices.empty()) return 0;  int bd = 0; // Buy day  int sd = 0; // Sell day  int profit = 0;  for (int i = 1; i < prices.size(); ++i) {  if (prices[i] >= prices[i - 1]) {  sd++;  } else {  profit += prices[sd] - prices[bd];  bd = sd = i;  }  }  profit += prices[sd] - prices[bd];  return profit;  }  };  int main() {  BestTimeToBuyAndSellStocksInfiniteTransactions solution;  // Test case  vector<int> prices = {11, 6, 7, 19, 4, 1, 6, 18, 4};  int maxProfit = solution.maxProfit(prices);  cout << "Max profit: " << maxProfit << endl; // Output: 30  return 0;  } | **Dry Run of the Code:**  **Input:**  prices = {11, 6, 7, 19, 4, 1, 6, 18, 4};  **Step 1: Initialize Variables**  bd = 0, sd = 0, profit = 0;  **Step 2: Iterate Through Prices**  **Iteration 1 (i = 1, price = 6):**   * prices[i] = 6, prices[i-1] = 11. * Since 6 < 11, we make a profit from the previous transaction: profit += prices[sd] - prices[bd] = prices[0] - prices[0] = 0. * Now, bd = sd = 1, as we reset the buy and sell days.   **Iteration 2 (i = 2, price = 7):**   * prices[i] = 7, prices[i-1] = 6. * Since 7 > 6, we continue to increase the sd. No profit is made yet.   **Iteration 3 (i = 3, price = 19):**   * prices[i] = 19, prices[i-1] = 7. * Since 19 > 7, continue to increase sd.   **Iteration 4 (i = 4, price = 4):**   * prices[i] = 4, prices[i-1] = 19. * Since 4 < 19, we sell at the previous sell day (19) and buy at 4. Profit is added:   profit += prices[sd] - prices[bd] = 19 - 7 = 12;   * Now, we reset the buy and sell days: bd = sd = 4.   **Iteration 5 (i = 5, price = 1):**   * prices[i] = 1, prices[i-1] = 4. * Since 1 < 4, we sell at 4, buy at 1, and update the buy day:   profit += prices[sd] - prices[bd] = 4 - 1 = 3;   * Reset the buy and sell days: bd = sd = 5.   **Iteration 6 (i = 6, price = 6):**   * prices[i] = 6, prices[i-1] = 1. * Since 6 > 1, continue increasing the sd.   **Iteration 7 (i = 7, price = 18):**   * prices[i] = 18, prices[i-1] = 6. * Since 18 > 6, continue increasing the sd.   **Iteration 8 (i = 8, price = 4):**   * prices[i] = 4, prices[i-1] = 18. * Since 4 < 18, we sell at 18, buy at 4, and update the profit:   profit += prices[sd] - prices[bd] = 18 - 6 = 12;  **Step 3: Add Final Profit for the Last Transaction**   * After the loop finishes, we still need to add the profit from the last buy-sell cycle:   profit += prices[sd] - prices[bd] = 4 - 4 = 0; // No profit is added.  **Final Profit Calculation:**  Total profit = 12 (from the second buy-sell cycle) + 3 (from the third buy-sell cycle) + 12 (from the fourth buy-sell cycle) = **30** |
| Output:- Max profit: 30 | |