Competitive Programming

Greedy Algorithm

B.Bhuvaneswaran, AP (SG) / CSE

9791519152

bhuvaneswaran@rajalakshmi.edu.in

RAJALAKSHMI ENGINEERING COLLEGE

An AUTONOMOUS Institution

Affiliated to ANNA UNIVERSITY, Chennai

Greedy Algorithm

- A greedy algorithm builds up a solution by choosing the option that looks the best at every step.

Example

- Say you're a cashier and need to give someone 67 cents (US) using as few coins as possible. How would you do it?
- Whenever picking which coin to use, you'd take the highest-value coin you could.
- A quarter, another quarter, then a dime, a nickel, and Finally two pennies. That's a *greedy* algorithm, because you're always *greedily* choosing the coin that covers the biggest portion of the remaining amount.

Other Problems

- Trying to fit as many overlapping meetings as possible in a conference room? At each step, schedule the meeting that *ends* earliest.
- Looking for a minimum spanning tree in a graph? At each step, greedily pick the cheapest edge that reaches a new vertex.

Note

- Sometimes a greedy algorithm doesn't give you an optimal solution.
- When filling a duffel bag with cakes of different weights and values choosing the cake with the highest value per pound doesn't always produce the best haul.
- To find the cheapest route visiting a set of cities, choosing to visit the cheapest city you haven't been to yet doesn't produce the cheapest overall itinerary.

Coin Change Problem

- Rs. 36
- Indian Currency
  - 100, 50, 20, 10, 5, 2, 1
- Minimize the number of notes / coins

Fictional Currency?

- Rs. 36
- Fictional Currency
  - 100, 50, 20, 18, 10, 5, 2, 1
- Minimize the number of notes / coins
- Greedy
  - 20 + 10 + 5 + 1 → 4 items
- Another Solution?
- Optimal substructure.

Coin Denominations

- Write a program to take value V and we want to make change for V Rs, and we have infinite supply of each of the denominations in Indian currency, i.e., we have infinite supply of {1, 2, 5, 10, 20, 50, 100, 500, 1000} valued coins/notes, what is the minimum number of coins and/or notes needed to make the change.
- Input Format:
  - Take an integer from stdin.
- Output Format:
  - Print the integer which is change of the number.

Example Input and Output

- Example Input:
  - 64
- Output:
  - 4
- Explanation:
  - We need a 50 Rs note and a 10 Rs note and two 2 rupee coins.
- Example Input:
  - 49
- Output:
  - 5
- Explanation:
  - We need a two 20 Rs notes and a 5 Rs coins and two 2 rupee coins.

Best Time to Buy and Sell Stock

- You are given an array prices where prices[i] is the price of a given stock on the i^{th} day.
- You want to maximize your profit by choosing a single day to buy one stock and choosing a different day in the future to sell that stock.
- Return the maximum profit you can achieve from this transaction. If you cannot achieve any profit, return 0.

Example Input and Output 1

- Input:
  - 6
  - 7 1 5 3 6 4
- Output:
  - 5
- Explanation:
  - Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6 - 1 = 5.
  - Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell.

Example Input and Output 2

- Input:
  - 5
  - 7 6 4 3 1
- Output:
- 0
- Explanation:
  - In this case, no transactions are done and the max profit = 0.

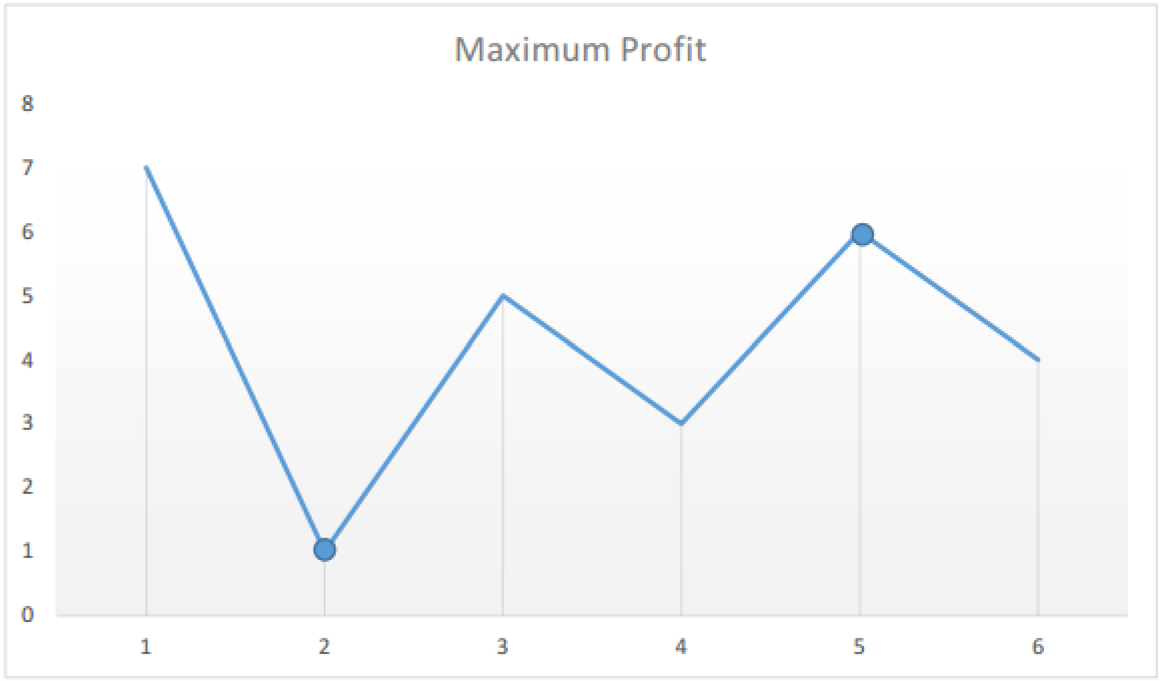
Constraints

- $1 <= prices.length <= 10^5$
- $0 <= prices[i] <= 10^4$

Brute Force

```
int maxProfit(int prices[], int n)
{
        int i, j, profit, maxprofit = 0;
        for (i = 0; i < n - 1; i++)
        {
                for (j = i + 1; j < n; j++)
                {
                        profit = prices[j] - prices[i];
                        if (profit > maxprofit)
                                maxprofit = profit;
                }
        }
        return maxprofit;
}
```

One Pass



Hint

- The points of interest are the peaks and valleys in the given graph.
- We need to find the largest peak following the smallest valley.
- We can maintain two variables - minprice and maxprofit corresponding to the smallest valley and maximum profit (maximum difference between selling price and minprice) obtained so far respectively.

One Pass

```
int maxProfit(int prices[], int n)
{
        int i, minprice, maxprofit;
        minprice = INT_MAX;
        maxprofit = 0;
        for (i = 0; i < n; i++)
        {
                if (prices[i] < minprice)
                        minprice = prices[i];
                else if (prices[i] - minprice > maxprofit)
                        maxprofit = prices[i] - minprice;
        }
        return maxprofit;
}
```

Best Time to Buy and Sell Stock V 2.0

- You are given an integer array prices where prices[i] is the price of a given stock on the i^{th} day.
- On each day, you may decide to buy and/or sell the stock. You can only hold at most one share of the stock at any time. However, you can buy it then immediately sell it on the same day.
- Find and return the maximum profit you can achieve.

Example Input and Output 1

- Input:
  - 6
  - 7 1 5 3 6 4
- Output:
- 7
- Explanation:
  - Buy on day 2 (price = 1) and sell on day 3 (price = 5), profit = 5-1 = 4.
  - Then buy on day 4 (price = 3) and sell on day 5 (price = 6), profit = 6-3 = 3.
  - Total profit is 4 + 3 = 7.

Example Input and Output 2

- Input:
  - 5
  - 1 2 3 4 5
- Output:
- 4
- Explanation:
  - Buy on day 1 (price = 1) and sell on day 5 (price = 5), profit = 5-1 = 4.
  - Total profit is 4.

Example Input and Output 3

- Input:
  - 5
  - 7 6 4 3 1
- Output:

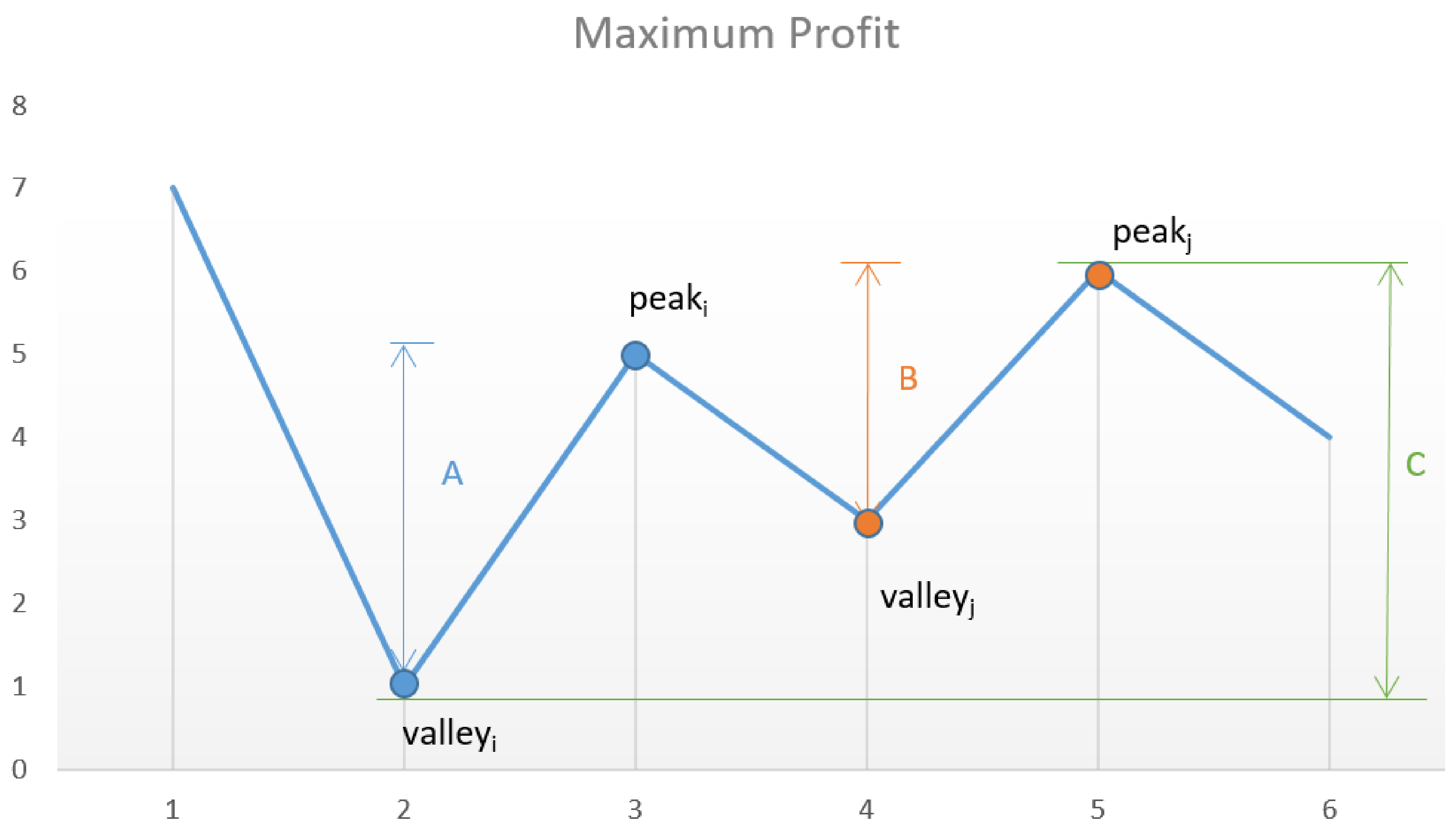
0

- Explanation:
  - There is no way to make a positive profit, so we never buy the stock to achieve the maximum profit of 0.

Constraints

- $1 <= prices.length <= 3 * 10^4$
- $0 <= prices[i] <= 10^4$

Peak Valley Approach



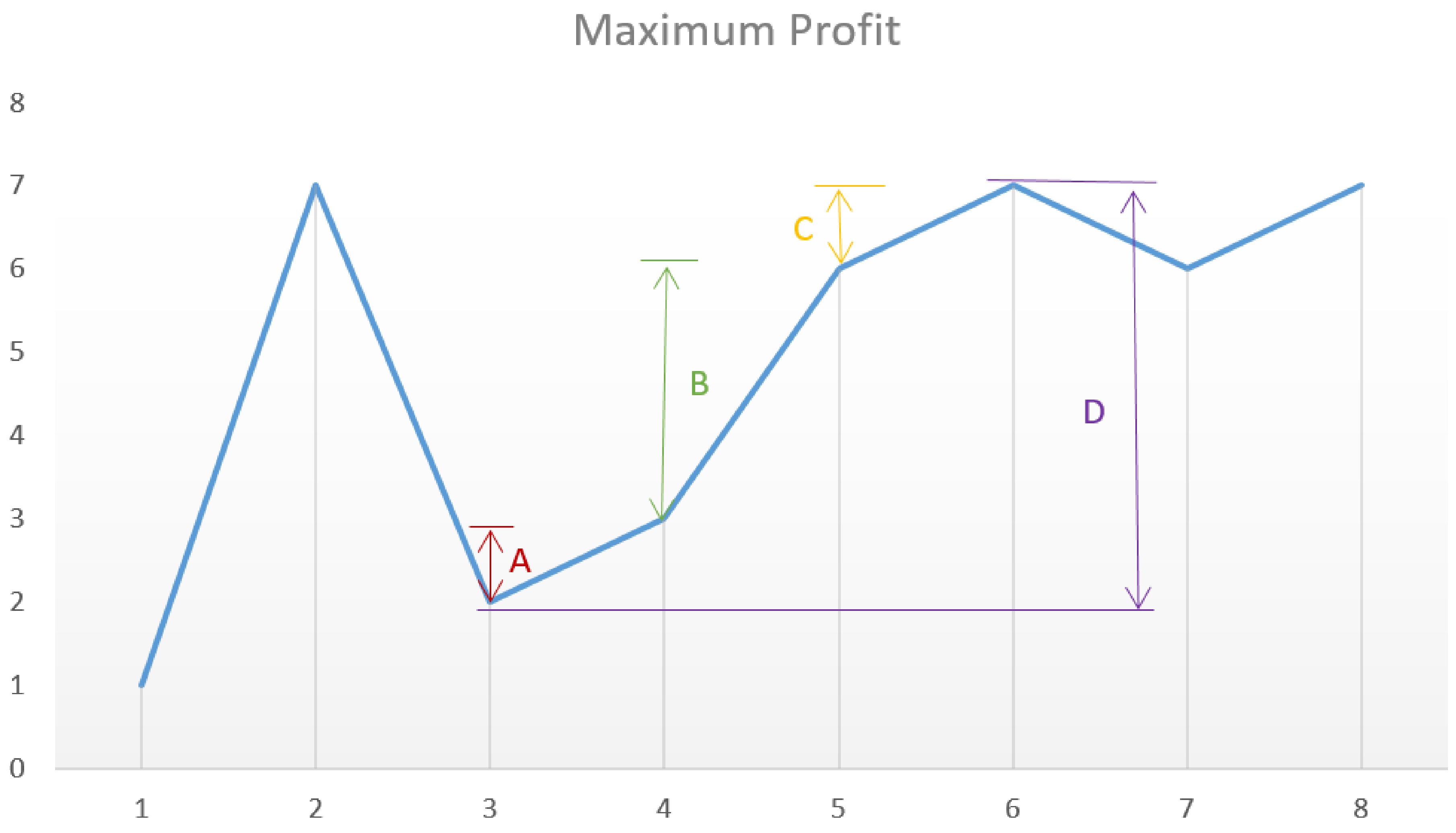
Hint

- The key point is we need to consider every peak immediately following a valley to maximize the profit.
- In case we skip one of the peaks (trying to obtain more profit), we will end up losing the profit over one of the transactions leading to an overall lesser profit.
- For example, in the above case, if we skip $peak_i$ and $valley_j$ trying to obtain more profit by considering points with more difference in heights, the net profit obtained will always be lesser than the one obtained by including them, since *C* will always be lesser than *A*+*B*.

Complexity Analysis

- Time complexity:
  - *O*(*n*) Single pass.
- Space complexity:
  - *O(1)* Constant space required.

Simple One Pass



Hint

- In this case, instead of looking for every peak following a valley, we can simply go on crawling over the slope and keep on adding the profit obtained from every consecutive transaction.
- In the end, we will be using the peaks and valleys effectively, but we need not track the costs corresponding to the peaks and valleys along with the maximum profit, but we can directly keep on adding the difference between the consecutive numbers of the array if the second number is larger than the first one, and at the total sum we obtain will be the maximum profit. This approach will simplify the solution.
- From the above graph, we can observe that the sum *A*+*B*+*C* is equal to the difference *D* corresponding to the difference between the heights of the consecutive peak and valley.

Fractional Knapsack

- Max weight
  - 28
- Items
  - Item 1: Rs.60 & 10 Kg.
  - Item 2: Rs.40 & 5 Kg.
  - Item 3: Rs.70 & 14 Kg.
  - Item 4: Rs.80 & 20 Kg.

Approaches

- Most valuable
- Least weight
- Value by weight

Coin Change

- Write a program to implement coin change making problem i.e. finding the minimum number of coins of certain denominations that add up to given amount of money.
- The only available coins are of values 1, 2, 3, 4
- Input Format:
  - Integer input from stdin.
- Output Format:
  - Print the minimum number of coins required to meet the given target.

Example Input and Output

- Example Input:
  - 16
- Output:
  - 4
- Explanation:
  - We need only 4 coins of value 4 each
- Example Input:
  - 25
- Output:
  - 7
- Explanation:
  - We need 6 coins of 4 value, and 1 coin of 1 value.

Solution

```
#include <stdio.h>
int main()
{
        int target, nums;
        scanf("%d", &target);
        nums = target / 4;
        if (target % 4 != 0)
                nums = nums + 1;
        printf("%d", nums);
        return 0;
}
```

Queries?

Thank You…!