Best time to buy and Sell Stocks infinite in C++

```cpp
#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;
}
```

Let's perform a **tabular dry run** of your code for the input:

prices = {11, 6, 7, 19, 4, 1, 6, 18, 4}

✅ Logic Summary:

- Buy at bd (buy day), sell at sd (sell day).
- Keep increasing sd as long as prices go up or stay the same.
- When price drops, add profit of the last segment (prices[sd] - prices[bd]) and reset bd = sd = i.

🗒 Dry Run Table:

i	prices[i]	Action Taken	bd	sd	Segment Profit	Total Profit
0	11	Initial buy	0	0		0
1	6	Drop → sell at 11, profit = 0	1	1	11 - 11 = 0	0
2	7	Rise → extend sell day	1	2		0
3	19	Rise → extend sell day	1	3		0
4	4	Drop → sell at 19, profit = 19 - 6 = 13	4	4	19 - 6 = 13	13
5	1	Drop → sell at 4, profit = 0	5	5	4 - 4 = 0	13
6	6	Rise → extend sell day	5	6		13
7	18	Rise → extend sell day	5	7		13
8	4	Drop → sell at 18, profit = 18 - 1 = 17	8	8	18 - 1 = 17	30
—	—	Final segment (bd == sd == 8) → 0 profit			4 - 4 = 0	30

✅ Final Output:

	Max profit: 30
	🧠 **Insight:**
	You earned profit from:
	- Buying at 6 → selling at 19 (Profit: 13)
	- Buying at 1 → selling at 18 (Profit: 17)
Output:- Max profit: 30	