IV003 Set 2, problem 2 25 points **due to 16. 4. 2025**

Name: Andrii Bezrukov **UČO:** 570943

0007 *list* 1 *učo* 570943 *body*

0123456789

Suppose you own a mobile shop that you can move from place to place. You sell in Brno or Znojmo, depending on where there is more demand. Suppose you have accurate demand projections for both cities for the next n days. However, you can't simply move your store depending on the greater demand because it takes one day and costs c to move your store from one city to another. For example, if you are in Brno on day 4 and want to move to Znojmo, then on day 5, you will have no earnings, you will pay a cost of c and you will have earnings from Znojmo only starting from day 6. Design an efficient *dynamic programming* algorithm that takes as input the cost of moving c, the earnings per day in the two cities, say $b_1, b_2, \ldots, b_n$ and $z_1, z_2, \ldots, z_n$, and outputs a schedule for the n days maximizing the total earnings. Assume that you can start with any of the two cities on day 1, without any additional cost.

1 Main Idea

The key idea is to define a dynamic programming state which captures the maximum earnings up to day i when you operate in either Brno or Znojmo.

Subproblems

Let us define:

$$\text{dp}[i][\text{Brno}] \quad \text{and} \quad \text{dp}[i][\text{Znojmo}]$$

as the maximum total earnings achievable up to day i if you are operating (i.e., selling) on day i in Brno or Znojmo, respectively.

Decisions:

- **Case 1: Staying in the same city:** If you continue in the same city, you simply add the earnings of day i to the total from day $i-1$. For instance, if you are in Brno, then

$$\text{dp}[i][\text{Brno}] = \text{dp}[i-1][\text{Brno}] + b_i.$$

- **Case 2: Switching cities:** In order to switch from one city to the other, you incur a cost c and you sacrifice one day of earnings. For example, if you want to be in Brno on day i, you must switch from Znojmo. That means on day $i-1$ you move, earning nothing and incurring cost c, and you base your decision on the total up to day $i-2$ in Znojmo:

$$\text{dp}[i][\text{Brno}] = \text{dp}[i-2][\text{Znojmo}] - c + b_i.$$

  A similar recurrence holds for switching to Znojmo.

Base Cases:

- On day 1, you can start in either city without any moving cost:

$$\text{dp}[1][\text{Brno}] = b_1, \quad \text{dp}[1][\text{Znojmo}] = z_1.$$

- On day 2, since a move requires a full day (i.e., the moving day), there is no opportunity to move and earn on day 2. Thus, you can only "stay":

$$\text{dp}[2][\text{Brno}] = \text{dp}[1][\text{Brno}] + b_2, \quad \text{dp}[2][\text{Znojmo}] = \text{dp}[1][\text{Znojmo}] + z_2.$$

Name: Andrii Bezrukov **UČO:** 570943

0007 *list* 2 *učo* 570943 *body*

0 1 2 3 4 5 6 7 8 9

2 Pseudo-Code

Below is the pseudo-code written with the `algorithm2e` package.

Input: Number of days n, moving cost c, arrays $b[1 \dots n]$, $z[1 \dots n]$ for Brno and Znojmo earnings.

Output: Maximum total earnings and schedule.

```
Initialize:
dp[1][Brno] ← $b_1$
dp[1][Znojmo] ← $z_1$
if $n \geq 2$ then
    dp[2][Brno] ← dp[1][Brno] + $b_2$
    dp[2][Znojmo] ← dp[1][Znojmo] + $z_2$

for $i \leftarrow 3$ to $n$ do
    // Option 1: Stay in the same city
    $option1 \leftarrow$ dp[$i-1$][Brno] + $b_i$
    // Option 2: Switch from Znojmo (move on day $i-1$)
    $option2 \leftarrow$ dp[$i-2$][Znojmo] − $c + b_i$
    dp[$i$][Brno] ← max{$option1$, $option2$}

    $option1 \leftarrow$ dp[$i-1$][Znojmo] + $z_i$
    $option2 \leftarrow$ dp[$i-2$][Brno] − $c + z_i$
    dp[$i$][Znojmo] ← max{$option1$, $option2$}

Answer: $maxEarnings \leftarrow$ max{dp[$n$][Brno], dp[$n$][Znojmo]}
return ($maxEarnings$, schedule)        // Schedule can be reconstructed by backtracking choices
```

Algorithm 1: Maximum Earnings for the Mobile Shop Location Problem

3 Proof of Correctness

- **Optimal Substructure:** The decision on day i is based solely on the optimal decisions made on previous days. Whether you choose to stay or to switch is decided by taking the maximum between the cost of staying and the cost of switching (which includes the moving penalty and the skipped day).
- **Base Cases:** On day 1, you simply take the earnings for the chosen city. On day 2, since a move would require the entire day for moving (and thus you would not earn on day 2), you must add the day 2 earnings to the day 1 earnings.
- **Inductive Step:** Assuming that the subproblems for all days less than i are solved optimally, the recurrence guarantees that for day i, by choosing the best between staying and switching, the solution remains optimal. Switching is modeled correctly by referring back to day $i-2$ and subtracting the moving cost.

Name: Andrii Bezrukov **UČO:** 570943

0007 *list* 3 *učo* 570943 *body*

0123456789

4 Running Time.

$O(n)$

5 Justification.

- **Time Complexity:** The algorithm loops from day 3 to day n and performs a constant number of operations per day. Hence, the running time is $O(n)$.
- **Space Complexity:** We maintain a table of size $2 \times n$ (for the two cities and n days), resulting in $O(n)$ space. Additional space for backtracking the schedule is also $O(n)$.