CS 336 Assignment 5

February 28th 2025

Problem 1.

A group of m people goes to an apple orchard to pick apples, which has n apple trees, located on a straight line in positions 1, 2, ..., n. The ith tree has a_i apples. A truck will drop off each person at any location they want (potentially different location for each person), and after that they will travel rightwards by foot, picking all apples from any tree they encounter along the way. Each person can travel at most distance L by foot. That is, if the truck drops off a person at location i, the person can gather all apples from trees $i, i+1, \ldots, \min(n, i+L)$. Of course, after a person gathers apples from a tree, the tree has no apples remaining, so there's no additional benefit from more than one person visiting the same apple tree.

What is the maximum number of apples this group can collect?

Example: Assume you have $a = [3, 0, 9, 7, 4]$, $m = 2$ and $L = 1$. An optimal solution is to drop off the first person at location 3 so that they gather $9 + 7$ apples, and then drop off the second person at location 5 so that they gather 4 apples.

Solve the problem using dynamic programming. Please do the following:

- Formulate the subproblem. Please state it as precisely as possible.
- Design a dynamic programming algorithm for solving this problem:
  - State the base case.
  - State the recurrence relation.
  - Explain why the recurrence relation is correct (from your explanation, one should understand how to get your the recurrence relation).
  - Please provide the **pseudocode**. Please use the bottom-up approach.
  - Explain:
    * What is the running time of your algorithm (all arithmetic operations take constant time). **The running time must be as small as possible.**
    * How to recover the maximum reward.

Problem 2.

There is a new series in your streaming platform, Panopto. The series contains n episodes in total. Episodes need to be watched in order; that is, you cannot watch episode j before episode i if $i < j$. Since you're busy, you decide to skip some subset of episodes (potentially empty). Your goal is to minimize the total amount of energy needed for this series, computed as follows:

- You figure out that if you skip episode i, you would have to spend p_i energy at the end of the year to figure out the missed content.
- In addition, each episode has excitement value e_i. You don't want to dramatically change your emotions as well. So, for any consecutive episode i and j you watch, you need to spend $|e_i - e_j|$ energy to adjust your mood as well.

For example, if there are 5 episodes:

- If you decide to watch episodes 1, 3, and 4, you need to spend $p_2+p_5+|e_1-e_3|+|e_3-e_4|$ units of energy.
- If you only decide to watch episode 3, you need to spend $p_1 + p_2 + p_4 + p_5$ units of energy.
- If you decide to watch none of the episodes, you need to spend $p_1 + p_2 + p_3 + p_4 + p_5$ units of energy.

Implement the following function, which returns the list of episodes you decided to watch in the sorted order (the episodes are 1-indexed). For example, if you decide to watch first, third, and fourth episodes, your function must return a vector with items $1, 3, 4$, in exactly this order. The input arrays are e and p respectively. It is guaranteed that for all test cases, the optimal answer is unique.

```
vector<int> Episodes(const vector<int>& excitement, const vector<int>& penalty)
```

Time limit The instructions are similar to the previous programming assignments. Your program should pass each tests in no more than 1 second. You can assume that $1 \leq n \leq 10^4$ and all numbers are between 1 and 10^9.