2 Sales itinerary (9 marks)

A salesperson lives on the coast. They travel to various cities along the coast to work. Sometimes they stay in a city for a day or a few days to work, and other times they simply pass through on their way to another city.

Since Covid-19, each city has instituted a policy of having travelers quarantine, but only if they want to stay in the city. If they are just passing through, they can continue on their way without quarantining. Also, each city asks visitors to quarantine for a different amount of time.

The salesperson has an idea of how much money they can make by working in each city, for each day. They need to decide which cities to travel to, and which cities to work in, in order to make the most money.

Each day, the salesperson can either work for the day in their current city (assuming they have finished quarantine), or they can travel to either adjacent city. Traveling always takes 1 day, and since the cities are along a coast, each city has two adjacent cities, except for two cities on the ends of the coast, which only have 1.

To solve this problem, you will write a function `best_itinerary(profit, quarantine_time, home)`.

2.1 Input

We think of the `n` cities as being numbered `0...n-1`. In one day, the salesperson can travel from city `i` to either city `i+1` or `i-1`. From city `0` they can only travel to city `1`, and from city `n-1` they can only travel to city `n-2`.

`profit` is a list of lists. All interior lists are length n. Each interior list represents a different day. `profit[d][c]` is the profit that the salesperson will make by working in city `c` on day `d`.

`quarantine_time` is a list of non-negative integers. `quarantine_time[i]` is the number of days city `i` requires visitors to quarantine before they can work there.

`home` is an integer between `0` and `n-1` inclusive, which represents the city that the salesperson starts in. They can start working in this city without needing to quarantine on the first day. If they leave and come back later, they will need to quarantine.

2.2 Output

`best_itinerary` returns an integer, which is the maximum amount of money that can be earned by the salesperson.

2.3 Example

```
profit = [
[6, 9, 7, 5, 9]
[4, 7, 3, 10, 9]
[7, 5, 4, 2, 8]
[2, 7, 10, 9, 5]
[2, 5, 2, 6, 1]
[4, 9, 4, 10, 6]
[2, 2, 4, 8, 7]
[4, 10, 2, 7, 4]]

quarantine_time = [3,1,1,1,1]
best_itinerary(profit, quarantine_time, 0)
>>> 39
best_itinerary(profit, quarantine_time, 1)
>>> 54
best_itinerary(profit, quarantine_time, 2)
>>> 47
best_itinerary(profit, quarantine_time, 3)
>>> 57
best_itinerary(profit, quarantine_time, 4)
>>> 51
```

2.4 Explanation of example

- h=0. The salesperson works in city 0 on the first day, then travels to city 1 on the second day, quarantines on the third day, then works in city 1 for the remaining days.
- h=1. The salesperson works in city 1 every day.
- h = 2. The salesperson works in city 2 on the first day, then travels to city 3 on the second day, quarantines on the third day, then works in city 3 for the remaining days.
- h=3. The salesperson works in city 3 every day.
- h=4. The salesperson works in city 4 for 3 days, then travels to city 3 on the fourth day, quarantines for a day, then works in city 3 for 3 days.

2.5 More examples

Since it is difficult to construct test cases for this question, I will be posting several additional test cases on the forums. As usual, students should feel free to create even more test cases and post them.

2.6 Complexity

`best_itinerary` should run in $O(nd)$ time and space, where n is the number of cities, and d is the number of days (i.e. the number of interior lists in `profit`). In the above example, n = 5, d = 8.

Warning

For all assignments in this unit, you may **not** use python **dictionaries** or **sets**. This is because the complexity requirements for the assignment are all deterministic worst case requirements, and dictionaries/sets are based on hash tables, for which it is difficult to determine the deterministic worst case behaviour.

Please ensure that you carefully check the complexity of each inbuilt python function and data structure that you use, as many of them make the complexities of your algorithms worse. Common examples which cause students to lose marks are **list slicing**, inserting or deleting elements **in the middle or front of a list** (linear time), using the `in` keyword to **check for membership** of an iterable (linear time), or building a string using **repeated concatenation** of characters. Note that use of these functions/techniques is **not forbidden**, however you should exercise care when using them.

These are just a few examples, so be careful. Remember, you are responsible for the complexity of every line of code you write!