Parknav Intro Puzzle

You are given a task of installing video cameras that will detect where cars are parked on one side of the street. Each camera can cover one or more parking spaces around it, as defined by cameraRange parameter. Camera will always be installed on top of a specific parking space.

Your job is to implement a function that, given a list of all possible parking space locations on the street, will return a minimum number of cameras needed to detect cars in all those parking spaces. Each parking space spans 1 unit of space, and spaces are numbered starting at 1 - so no elements in parkingSpaces array will be less than 1. parkingSpaces array will always be sorted from low to high, as in examples.

We're looking for implementation with the fastest runtime and smallest space complexity you can write. We haven't proved it, but if you end up with constant time runtime complexity, most likely you have a mistake somewhere.

Please make sure your code compiles, and works correctly for more than just the 3 examples outlined below. Once you're done, please send your function to sergei@parknav.com
In addition, make sure you're only sending one file that will have your implementation, the implementation has the required function in it, with the same signature, and it has no library dependencies on anything else apart from core Java or Swift.

In Java or Swift, implement the following function:

static int findMinimumNumberOfCameras(int cameraRange, int[] parkingSpaces)

OR

func findMinimumNumberOfCameras(cameraRange: Int, parkingSpaces: [Int]) -> Int

REMEMBER that camera can only be installed on top of an existing parking space.

Example 1:
cameraRange = 1
parkingSpaces = [1 , 2, 3, 4, 5]

The result here is 2. If you place one camera on space 2, it will cover spaces 1, 2, and 3 (covers both ways), and camera on space 4 will cover space 4 and 5. It's ok to "double-cover" spaces, and note that we **ARE** looking at parkingSpaces[0].

Example 2:

cameraRange = 2
parkingSpaces = [2, 4, 5, 6, 7, 9, 11, 12]

The result is 3. The optimal coverage will be achieved if you install cameras at locations 4, 9 and 12. Camera at 4 will cover spaces 2, 4, 5, and 6. Camera at 9 will cover 7, 9, and 11. Finally, camera at 12 will cover 12.

Example 3:
cameraRange = 10
parkingSpaces = [1, 15, 30, 40, 50]

The result is 3. You would need one camera at 1, another at 15, and 3rd one at 40.

cameraRange can go up to 10 spaces, and each street can have 100 parking spaces. You **don't** need to write code to validate this.
While this might seem like no one would care about runtime complexity for 100 spaces, there are 100000+ streets in some cities ;)