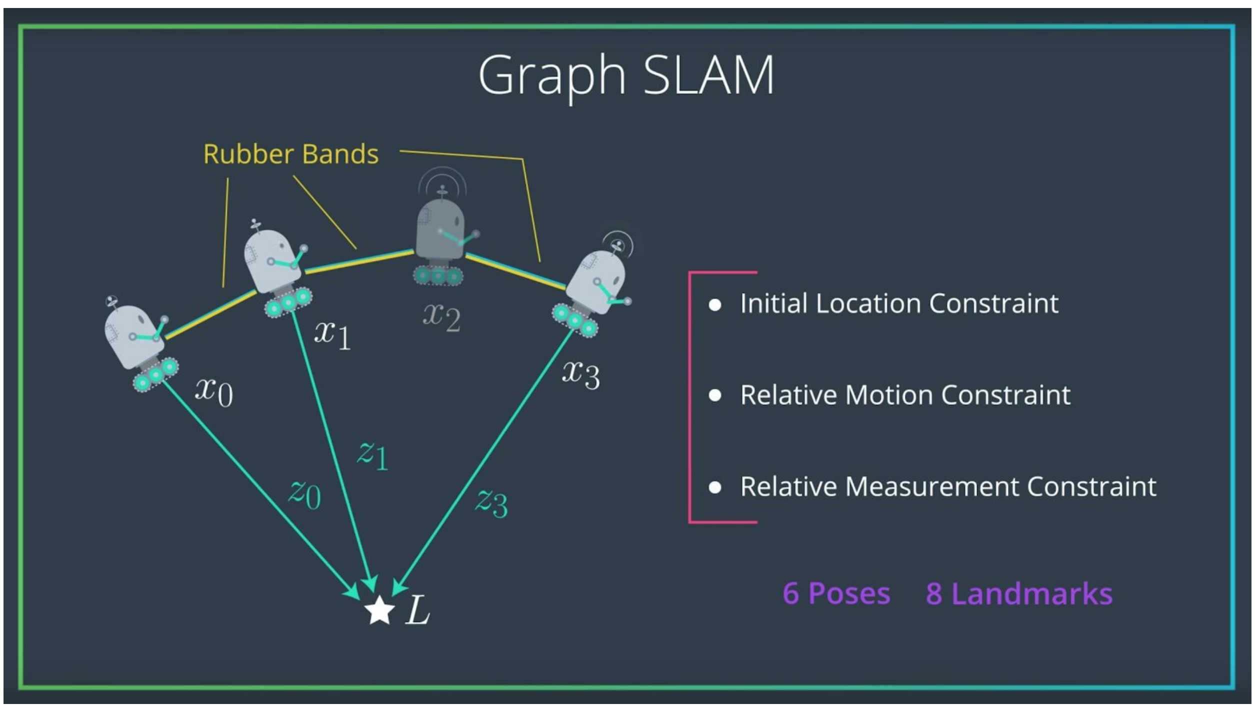
Simultaneous Localization and Mapping

In the previous lessons, you learned all about localization methods that aim to locate a robot or car in an environment given sensor readings and motion data, and we would start out knowing close to nothing about the surrounding environment. In practice, in addition to localization, we also want to build up a model of the robot's environment so that we have an idea of objects, and landmarks that surround it *and* so that we can use this map data to ensure that we are on the right path as the robot moves through the world!

In this lesson, you'll learn how to implement SLAM (Simultaneous Localization and Mapping) for a 2 dimensional world! You'll combine what you know about robot sensor measurements and movement to create locate a robot *and* create a map of an environment from only sensor and motion data gathered by a robot, over time. SLAM gives you a way to track the location of a robot in the world in real-time and identify the locations of landmarks such as buildings, trees, rocks, and other world features.

For a series of robot motions and landmark locations, how many constraints will there be for 6 total poses and 8 landmarks?

Ans: 14



Constraints

For our 6 poses, we have:

- 1 initial location constraint
- 5 additional, relative motion constraints, and finally,
- 8 relative measurement constraints for the 8 landmark locations

Adding all of these up gives us a total of **14** constraints.

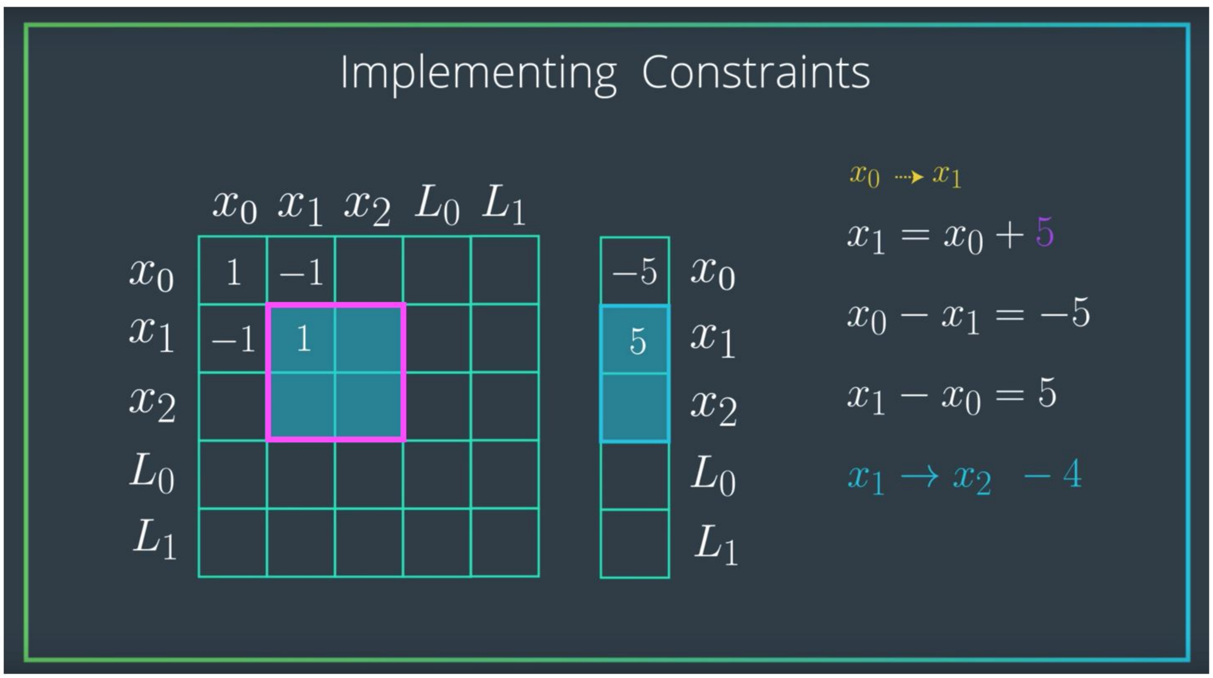
Now, consider the image above, with 4 poses (including the initial location `x0`) and one landmark. We can use this same math and say that there are 5 total constraints for the given image, but in practice there are usually many more measurements and motions!

Implementing Constraints

You also may have noticed that not all of these constraints will provide us with meaningful information, such as in our example: we do not have a measurement between the pose `x3` and the landmark location. Next, let's see how we can represent these constraints in a **matrix** and their relationships with values in that matrix and a constraint **vector**.

Constraint Matrices

Next, you'll see how to implement relationships between robot poses and landmark locations. These matrices should look familiar from the section of linear algebra, but I also find it helpful to think of the values in these matrices as **weights** kind of like you've seen in convolutional kernels, only these weights imply how much a pose or landmark should be weighted in a set of equations.



QUESTION 1 OF 2

For the highlighted values in the square above, what will the new weights relating x1 and x2 be after this motion update? They start as:

```
[ 1   0 ]
[ 0   0 ]
```

Which can also be read as: `[ 1 0 ], [ 0 0 ]`

`Ans: [ 2 -1 ], [ -1 1 ]`

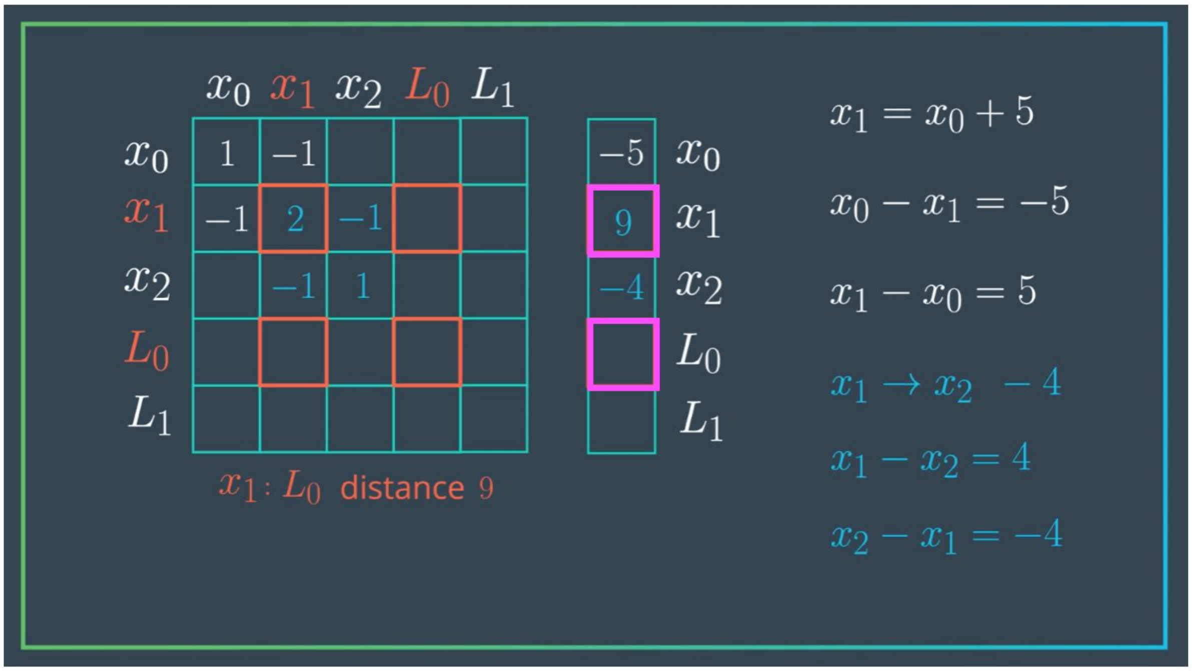
QUESTION 2 OF 2

For the highlighted values in the small rectangle above, what will the new values relating x1 and x2 be after this motion update? They start as:

```
[ 5 ]
[ 0 ]
```

Which can also be read as: `[ 5 ], [ 0 ]`

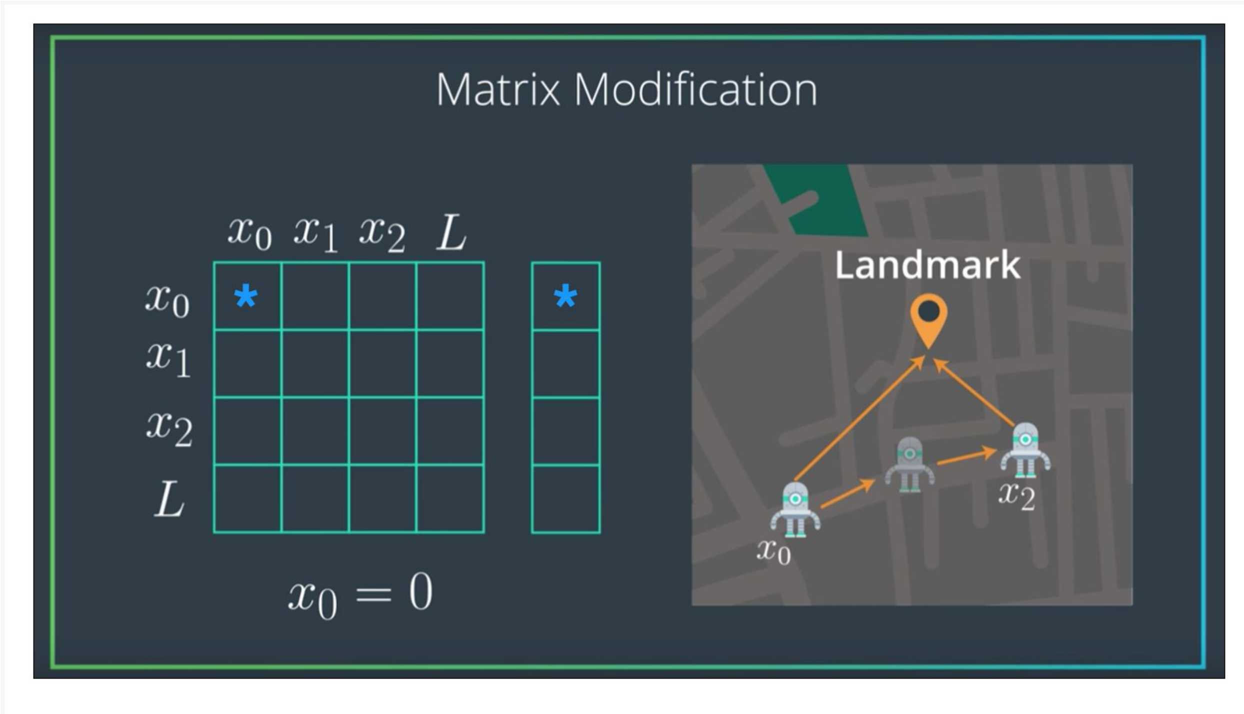
`Ans: [ 9 ], [ -4 ]`



QUIZ QUESTION

For the two highlighted values in the column vector above, what will there new value be after this landmark relationship is added to these constraint matrices? Right now the values are 9, 0.

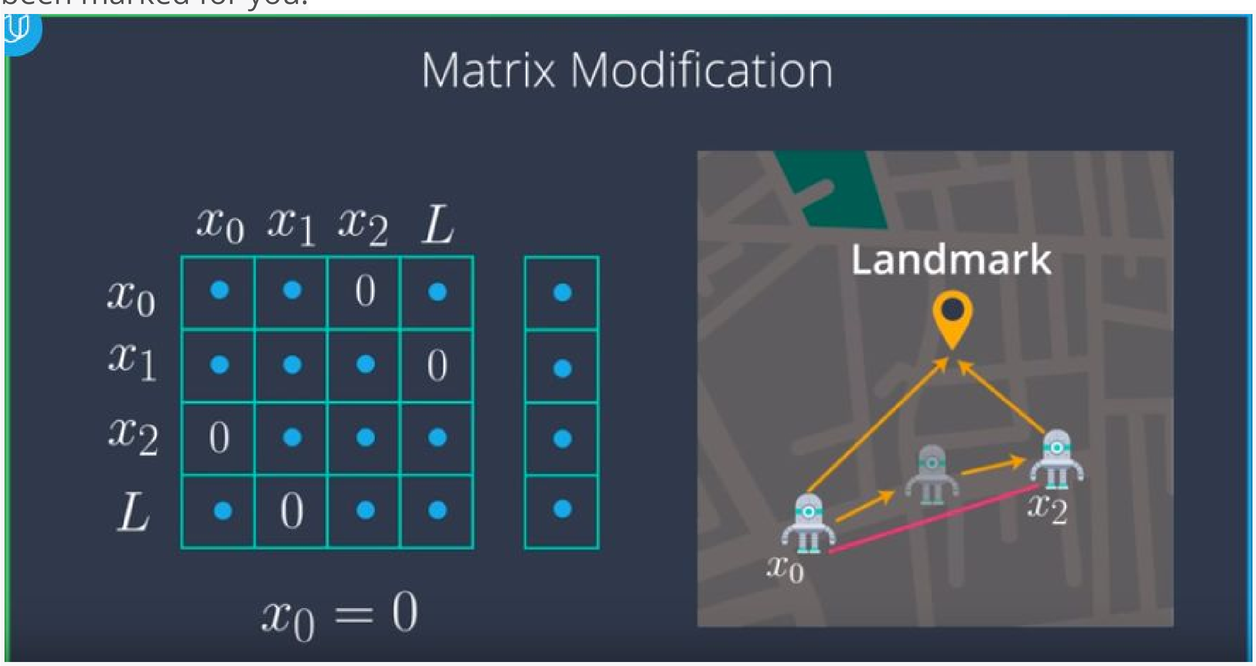
`Ans: [0,9]`

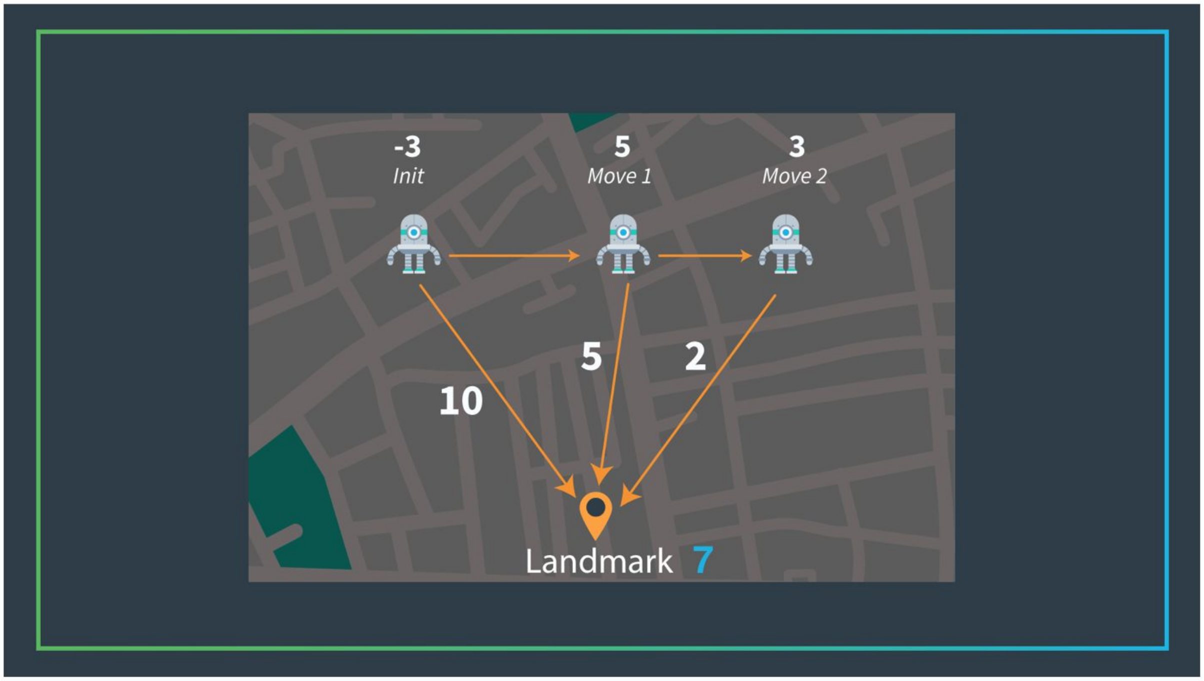


Mark the Relationships

For this quiz, take out a piece of paper and draw these 4x4 and 4x1 grids. Draw a dot (or other mark) wherever there is a relationship between poses and/or landmarks. If nothing will change about a cell, it will remain 0 and you can leave it blank.

An example has been started in the image below, note that the initial constraint has been marked for you!





Landmark Position

Given only the above constraints between poses (x0, x1, x2) and the landmark position. Where do you think the landmark is? (The answer should be a single integer value.)

The landmark, L, is at the value 7.

Think about the relationship between x0 and L. x0 is at -3 and sees L a distance of 10 away: `L = -3 + 10`.

And if we keep adding poses and sensor measurements this remains consistent. As x0 moves to x1, we get `L = -3 + 5 + 5`

And the final loop all the way to x2: `L = -3 + 5 + 3 + 2`