# Question 1

* Part (a) is a value, i.e., a real number. Part (b) is a path, i.e., sequence of vertices or squares in this problem.
* This is like the difference between the shortest distance from vertex to in a graph, and a path whose sum of edge-weights is that shortest distance.

1. **Least Total Cost Recurrence**
2. **Least Cost Path Recurrence**

* The returns just the previous square that is in its min sum path, like for shortest paths. “…which is the immediately previous square…”

# Question 2

1. **Optimal Substructure**
2. **Feasibility**
3. **Fewest Stops Recurrence**

* The notation means you must leave by a certain time to ensure that you reach by the end of the day.
* Other wording: Assume that we always start to drive at a particular time of day, say 7 AM, and always end driving at a particular time, say 7 PM. So, when we say, “within a day,” we mean if we leave by 7 AM, then we arrive by 7 PM.
* What means is that if we leave at 7 AM, then we’re guaranteed to reach no later than 7 PM.

1. **Pseudocode**

* Space efficiency characterizes the space, in addition to that needed to encode the input, the algorithm consumes/requires.
* In other words, ignore the space needed to encode the input, but count every other space that is allocated. And yes, you would include the space needed to encode the output.

# Question 3

* To be clear, we do not need resources such as ways to measure time and distance.

1. **Lower and Upper Bounds on Alignment Length**
2. **Worst-Case Number of Possible Alignments**

* May end up with something more than . The problem asks only that you establish as a lower bound, as that is what means.