1. Short answer.
   1. True, remains the MST. Every MST of a graph has -1 edges. The cost of this MST goes down by collectively, and so does every other potential MST.
   2. False:
   3. Recognizable but not decidable. A program decides the halting problem. Write a program whose inputs are a program and its input where it runs on line of but before everything originally on line . will return true if halts because it can get to that line, and will return false if it doesn’t halt, because it can never get to that line.
2. We do this using dynamic programming. The base cases, which we set at the beginning of the code, is

The recurrence schemes are

Because you either do just that one obstacle at step or do two obstacles, and .

We fill those two arrays in from to .

The solution is

This algorithm is because you iterate over each number from to , in each of which you perform several constant time operations. Damn it I overthought it; I had the correct answer before. The recurrence scheme is just . This relies on the assumption that we stopped at obstacle . I think mine is still correct, though.

1. I don’t know. We won’t start using a new truck unless . When we are on the m­th truck, the total weight is . So, we need more than that divided by many trucks (i.e. ). QED.
2. We won’t