1. What were your results from compare_cow_transport_algorithms? Which algorithm runs faster? Why?

Greedy - 6
3.170967102050781e-05
Brute Force - 5
0.6026601791381836

Greedy runs faster. Because brute force enumerates over all possibilities. For a list of size n, it generates Bell(n) number of partitions.

2. Does the greedy algorithm return the optimal solution? Why/why not?

No, because greedy does not guarantee optimality, it simply chooses the best optimization locally. And can get stuck in a local minima.

3. Does the brute force algorithm return the optimal solution? Why/why not?

Yes, we enumerate over all possibilities, so we hit the optimal case atleast once.

1. Explain why it would be difficult to use a brute force algorithm to solve this problem if there were 30 different egg weights. You do not need to implement a brute force algorithm in order to answer this.

We end up searching through alot of states if we use brute force.

2. If you were to implement a greedy algorithm for finding the minimum number of eggs needed, what would the objective function be? What would the constraints be? What strategy would your greedy algorithm follow to pick which coins to take? You do not need to implement a greedy algorithm in order to answer this.

The objective function would

Minimize z = x1 + x2 + x3 + … + xn

Where xi = number of eggs chosen of egg type i
Subject to

x1 * w1 + x2 * w2 + x3 * w3 + …. + xn * wn = target

Where wi = weight of egg of type i

Pick maximum weight egg as many times as possible until we can't no more.
Then next maximum, then so on.

3. Will a greedy algorithm always return the optimal solution to this problem? Explain why it is optimal or give an example of when it will not return the optimal solution. Again, you do not need to implement a greedy algorithm in order to answer this.

No, it does not guarantee optimal solution.
1 5 7 8 <- egg weights
Optimal = 13
Greedy = 12 + 1 + 1 + 1 = 15
Does not guarantee optimal solution.