PS1 writeup

1. The algorithm that ran the fastest was the greedy algorithm, this is due to the program efficiency, with the sorting of the list taking O(nlogn) as uses variant of merge sort. Then I go through the list only once in order, thereofre linear complexity, which can be ignored for big O notation. So O(nlogn). However, it did not find the perfect solution, taking 6 trips as opposed to 5 for the brute force algorithm. The mistake was the algorithm put "moo moo" on first to ship 4, so it could only take two cows, whereas the brute force algorithm was able to find the solution where three cows went onto that ship, saving the last trip.
However, the brute force was considerably slower, taking 3.2 seconds compared to the 0.00013 seconds of the greedy algorithm. this is because all the partitions 2**n+1, making it exponential as an algorithm, with the exponential list then having to be looped over for every cow, to check it was valid and the total number of ships. So O(2**n) - exponential.
2 & 3. The greedy algorithm did not find the perfect solution, taking 6 trips as opposed to 5 for the brute force algorithm. The mistake was the algorithm put "moo moo" on first to ship 4, so it could only take two cows, whereas the brute force algorithm was able to find the solution where three cows went onto that ship, saving the last trip. This is because the brute force algorithm will have compared the two different solutions and would have chosen the one with less trips. While the greedy algorithm put Oreo and Moo Moo together as they were the heaviest, hoping that would be the optimal solution, but not checking that it was.

Problem B writeup

1. There would be 2**30 different possibilities as the brute force algorithm is an exponential algorithm that would create an extra layer per weight in the search tree, which would have double the possibilities per layer. This would be too computationally expensive.

2. The objective function would be to have the smallest number of eggs that could be added to the ship, with the constraint that the ship has to have the target weight. The greedy algorithm, would therefore take as many of the heaviest egg that could fit without exceeding the target wieght, then take the second smallest and so on till it has the target weight.

3. the greedy algorithm will always return the optimal solution as there are no combinations of lighter eggs that can take up less space and weight more than the heavier egg.