Luis Enrique Franco Marín

410921353

**Week 6: Knapsack Problem**

The knapsack problem is a combinatorial optimization problem that entails: Determine the quantity of each item to include in a collection given a set of objects, each with a weight and a value, so that the total weight is less than or equal to a given limit and the total value is as large as possible. It gets its name from the issue that someone with a fixed-size knapsack faces when they have to fill it with the most precious stuff. The issue frequently arises in resource allocation, as decision makers must choose among a set of non-divisible projects or activities while working under a strict budget or time constraint.

**Problem:**

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.

The first line of input contains a single integer between 1 and 100: the length of the ferry (in meters). For each car in the queue there is an additional line of input specifying the length of the car (in cm, an integer between 100 and 3000 inclusive). A final line of input contains the integer 00. The cars must be loaded in order, subject to the constraint that the total length of cars on either side does not exceed the length of the ferry. Subject to this constraint as many cars should be loaded as possible, starting with the first car in the queue and loading cars in order until no more can be loaded.

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

The first line of output should give the number of cars that can be loaded onto the ferry. For each car that can be loaded onto the ferry, in the order the cars appear in the input, output a line containing 'port' if the car is to be directed to the port side and 'starboard' if the car is to be directed to the starboard side. If several arrangements of the cars meet the criteria above, any one will do. Please guarantee first car be directed to port side.

**Code:**

Text

Description automatically generated

**Discussion:**

The problem is solved using dynamic programing, memory is allocated for the program first with memset but the analysis is done with the memory used. This is a knapsack problem in which the two sides of the ferry are filled to their maximum capacity. Just as a common knapsack problem it is pseudopolinomial in that it’s complexity depends on the length of the ferry, but it also depends on the number of entries in the input. Being V the length of the ferry (taking into account that it has a unit conversion multiplying it by 100) and n the number of entries. The complexities are as given:

Time: f(2n+nV)=O(nV)

Space: f(2nV+3n)=O(nV)

**Result:**

Logo, company name

Description automatically generated with medium confidence