CO1406 – Algorithms and Data Structures

Tutor: Panayiotis Andreou

Assignment 1 – Container Packing Algorithm

Serdyukov Antoniy – G20982067

Deadline 21/03/2023

Report

This report aims to analyze the requirements and complexity of the algorithms in general and functions of said algorithms in particular. Additionally, several heuristics will be discussed that were chosen to significantly speed up the algorithm.

Requirements Analysis

Problem outline: The algorithms is supplied with a set of boxes and a container. It should determine a possible solution/solutions (if any).

1. Datatypes

The first decision that had to be made, related to the data structures datatypes. Since the algorithms deals with a “real” example, it is safe to assume that the negative numbers would not be used. Thus, by using unsigned types, we can double our limits. From now on, every type is unsigned.

Furthermore, the data types themselves must be considered, i.e., the sizes of the box, container, stack and counters. It was decided that the size of the container will be stored in an unsigned short, and unsigned char. A bigger data type was not chosen because, a) it is unrealistic to have containers/boxes that are bigger than the Earth, b) if the container is big enough, and the box is small enough, it might take all the time in the Universe to calculate that: for example, container of type int and box type short, then there would be 4,295,098,369 boxes that would fit in the container, and iterating through all of their positions and rotations is infeasible. Thus, the size for container was chosen to be short and type char for box. Since, the smallest box size is 1, the container should be able to fit 4294836225 boxes, therefore number\_boxes should be int. This means almost 4 Gb of boxes, 8 Gb if the short is used instead of char. Nevertheless, the code is structured so that it could be scaled to bigger numbers, in case we need to accommodate more accurate dimensions, like cm, mm and so on.

For the stack, the same type is used as in the container (short) – automatically, with template – because the stack is storing the coordinates of the placed box and cannot exceed the size of the container. Additionally, the stack contains the pointer to the box. The alternative would be to store the box itself, which would be 4 times less space, but if the box size were to change – this changes accordingly. Therefore, the stability here is chosen over space complexity.

Complexity Analysis