0.1 The Packing Algorithm

The algorithm used for packing the item in this project is based on the known theory of the Bin Packing Problem (see section **??**). The algorithm has been a mix of some elements from the First Fit Decreasing and the Best Fit method. These has been combined to make the algorithm used in the project. The algorithm is inspired of **?**. The following sections will describe the used code. The code will try to place all the items in the luggage so it is packed most optimal in both sized and weight.

0.1.1 Optimization of weight

The optimization of the weight is done, so no luggage is exeeting the weight limit if is could be distributed different. Furthermore it would be preferable for the user, that no luggage is very heavy and another very light. To optimize the distribution of the weight, the average weight per luggage should be calculated. It is calculated as seen on equation 1, where N = Number of items.

$$\text{Avarage Weigt} = \frac{\sum_{i=1}^{N} I_{weight}}{N} \tag{1}$$

It is possible to distribute the weight average in the suitcases, when the optimal weight for each suitcase (the average weight calculated by equation 1) is known. The program will try to distribute the weight equally, but not if it will mean that the luggage cannot be packed. Therefore the weight distribution is a optimization goal, but not as important as the volume of the luggage. This part of the optimization is a Best Fit, because it find the best luggage for an optimal weight distribution.

0.1.2 Description of the algorithm

The algorithm use the First Fit Decreasing (FFD) method to pack the items in the luggage. This means, that the algorithm at first sorts the items by size. It will start by packing the largest items first, which will give a better result for the packing. When the list is sorted, it will then also sort the list of luggages by size. The algorithm can now go to the actual packing.

The general algorithm is:

- Sort the items by size
- Check if the item can be in the first luggage, while the luggages total weight does not exit the average weight per luggage
- Check if the item can fit in the luggage, else check the next
- If the item cannot be fitted, exclude it from the list, and notify the user

This is a very general overview of the algorithm, and does not explain the process of the algorithm detailed enough. To explain the algorithm, a flowchart can be seen on figure 0.1.

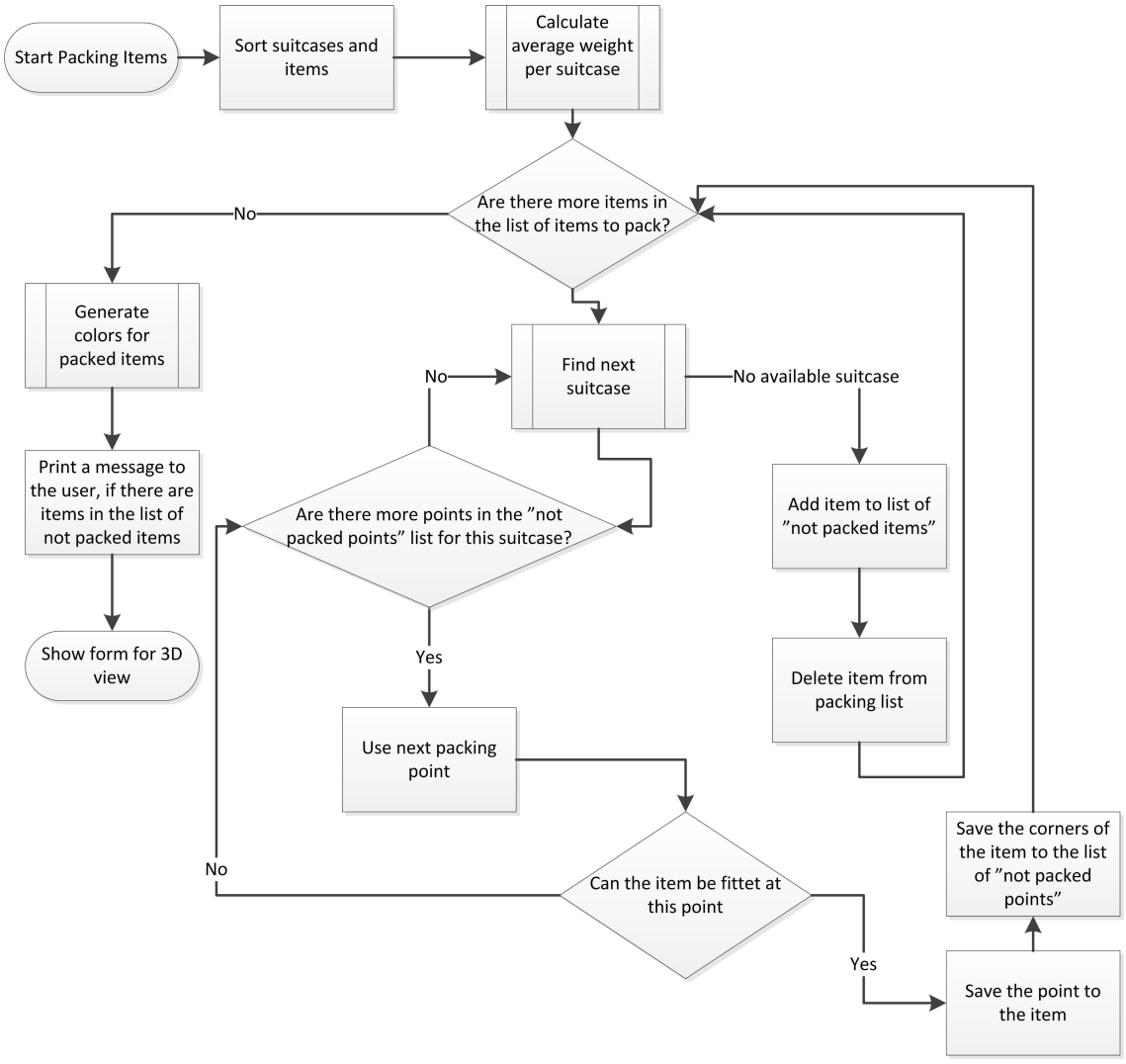

Figure 0.1: The flowchart for the packing algorithm

As seen on the flowchart, the first process which is done, is to sort the suitcases and items. They are sorted by size, so the biggest items are stored in the biggest suitcases. This will ensure the most effective way of packing, because it will fill the items, which is the hardest to fit first. The next step for this algorithm is to call the function, which will calculate the average weight per suitcase for an even weight distribution. The algorithm can now start the actual packing of the items. It will make a loop, checking all the items i n the list of items to pack. In the function "find next suitcase", the algorithm should find next available suitcase, checking for optimal weight distribution. To pack the item, each suitcase have a list of possible packing points, called "not packed points". This list should contain the points in the suitcase, where the next item can be fitted. This will ensure that the items are packed as effective as possible. The items will always be packed in a corner of an other items, so there will be no space between two items. This list of not packed points is individual for each suitcase, and if no item has been packed yet, the point [0,0,0] will be used. If the item cannot be fitted in the suitcase, it will try the next suitcase. In the case that the item cannot be fitted in any suitcase, it will add the item to the list of not packed items, and delete the item from the list of items to pack. This will make it possible to separate the items which is packed and those which is not packed. If the item could be packed it will be saved the point to the item, so the program know

its position further on. The corners of the item will be saved on the "Not packed points" list, and the next item will be packed. When all the points are packed, it will generate the colors for the items. The items must have a color, but no color must be the same as the item next to it, so they will not be mixed for the user. If some items couldn't be packed, they will be shown to the user, and the form to show the suitcase in 3D will appear.

0.1.3 Round up

This is the theory of the algorithm to pack the suitcase. It is only a skeleton of the algorithm, but the theory about how to pack is in place. This algorithm is the base for the packing program in this project, and the mix of FFD and Best Fit will ensure an equal distribution of weight and space at the same time.

```
1  for (i = 0; i <= Number; i++)
2  {
3      tmp_list = new cbox();
4      /*Loops that goes through the 3 koordinates*/
5      for (x = (luggage_items_to_pack[i].saved_point_x - marking); x <=
           (luggage_items_to_pack[i].saved_point_x +
           luggage_items_to_pack[i].width + marking); x++)
6      {
7          for (y = (luggage_items_to_pack[i].saved_point_y - marking); y <=
               (luggage_items_to_pack[i].saved_point_y +
               luggage_items_to_pack[i].depth + marking); y++)
8          {
9              for (z = (luggage_items_to_pack[i].saved_point_z - marking); z
                   <= (luggage_items_to_pack[i].saved_point_z +
                   luggage_items_to_pack[i].height + marking); z++)
10             {
```

Listing 1: hejcap