IFT 2015  
TP 2  
(25 points)  
21 février 2014

Transport operators calculate the price for sending a parcel based on its weight, but also its volume. For a company that sends multiple objects to a single recipient, so it is important to minimize the volume of the box to use to send these objects. The problem here is to find, for a list of given objects of different sizes, the spatial configuration which allows for a smaller box, possible, that is to say to minimize wasted space in the box . To simplify the problem somewhat, we will be reduced to two dimensions, but three-dimensional extrapolation is done directly. To solve this problem, you will implement a data structure which we call clipping plane.

A cutting plane divides a rectangle whose sides are horizontal and vertical using horizontal and vertical sections (see Figure 8.25 and P-8.67 the number Book, pp. 359-360). A clipping plane may be represented by a binary tree, called cutter shaft, whose nodes represent the internal sections and the external nodes (or leaves) the rectangles represent base in which the map is divided by the cuts. The compaction problem for a clipping plane is defined as follows. Suppose that each rectangle of a basic cutting plane is assigned a width w and a height h, corresponding to the dimensions of the object to insert. The compaction problem is to find the smallest width and the smallest height of the rectangle of each of the cutting plane which is compatible with dimensions of basic rectangles. Specifically, this problem requires the assignment of values ​​h (p) and w (p) for each position (or node) of the p cutting shaft so that:

You must create a binary tree structure that allows you, among other things create a plan for cutting, cut a rectangle horizontally or vertically, to assign a minimum width and height of a rectangle base, compact tree, produce a string describing the cutting shaft (see below). You are free to make all functions and any other classes that you think you need to solve this problem. Note that two connected nodes in the tree cannot represent two successive divisions in the same direction, otherwise the problem is found completely different.

Your main () function receives as input a list of tuples ("nom\_objet", w, h) and must return **a list containing at least one cutting tree** (as a string) associated with a cutting plane that minimizes the area box (there may be several). These trees will be represented as parenthesized prefix. For example, the tree of the page 360 would be represented by '(- (|, A (-, B, (|, C, D))) (|, E, F)) where ‘-‘ and ‘|’ respectively horizontal and vertical divisions, and letters the names of objects.

Your algorithm should, among all possible cuts, i.e. all possible cutting trees, determine the one or ones that produce the smallest box (area). Note that there are orders (permutations) possible leaves of a tree (including isomorphic trees).

In addition, each object (sheet) can be oriented in 2 ways (rotated 90 degrees), which can result in better compaction. There are therefore possible combinations of rotations of n objects. So for every possible tree topology, there are different trees, according to the rotation of objects. Your algorithm should return the form of a list of strings, all the trees that produce a box whose area is minimal. Each string describing a shaft must be in the form

0100:(-,(|,(-,A,B),C),D):12

Where 0100 represents which objects are turned (1 for yes, 0 for no) compared to original dimensions received. In this tree, only object B was shot. (- (| (-, A, B), C), D) represents the shaft, and the area 12 is obtained by shaft once compacted.

For example, if one has to place the objects ("A", 2, 7) and ("B", 5, 3), possible trees (and lowest areas thereof) are:

11:(-,A,B):49  
11:(|,A,B):50  
11:(-,B,A):49  
11:(|,B,A):50  
10:(-,A,B):35  
10:(|,A,B):36  
10:(-,B,A):35  
10:(|,B,A):36  
01:(-,A,B):36  
01:(|,A,B):35  
01:(-,B,A):36  
01:(|,B,A):35  
00:(-,A,B):50  
00:(|,A,B):49  
00:(-,B,A):50  
00:(|,B,A):49,

and thus the smallest possible area is 35, obtained by four different divisions. Your algorithm should return at least one item from the list:

[ "10:(-,A,B):35", "10:(-,B,A):35", "01:(|,A,B):35", "01:(|,B,A):35" ].

Rating Scale (25 points)

Your structure will be tested with 5 lists of objects (lists 5 × 5 points = 25 points), different from those currently provided in the unit test. For each of the five lists, here is the schedule:

5 points if you return at least a minimum tree cutting and no non-optimal tree;

3 points if you return at least a minimum tree cutting and no non-optimal tree, but the return format is not compatible with the unit test provided;

1 point if you return at least a minimum tree cutting and also one or more suboptimal trees;

0 points if you return at least one tree is not minimal, or if you do return no minimum tree.

In principle, if your structure works correctly, no errors should be obtained during the execution of the unit test.

Reminder

Put all your files in Studium. Py (executables under Python 3), making sure to identify teammates early TP2.py file as a compressed folder.

Good work!