NOPT042 Constraint programming: Tutorial 13 - More problems

What was in Lecture 10?

Over-constrained problems

- no feasible solution
- enlarge some domain ("buy more clothes")
- enlarge some constraint relation ("dress less elegantly")
- remove some constraint ("no need to match shoes and shirt") - same as enlarging the relation by making *all* tuples feasible
- remove some variable ("do not wear shoes")

Partial CSP

- find solution of a problem "close" to the original problem
- metric - distance between CSPs (e.g. number of solutions that are feasible in one but not the other problem, or number of different tuples in constraint relations)
- sufficient distance < maximal allowed distance

Weighted/valued CSP

- each constraint has a value
- minimize sum of weights of violated constraints
- soft vs. hard constraints (weight=+infinity)

Probabilistic/semiring-based CSP

- each tuple of values is annotated by a preference (how well it satisfies the constraints)
- find solution with largest aggregated preference

Constraint hierarchies

- preferences of some constraints over others (e.g. required vs. strong vs. weak, hard vs. soft)
- satisfy all hard constrains, satisfy soft as well as possible
- formalization of this framework

In [1]:
```
%load_ext ipicat
```

```
Picat version 3.7
```

Exercise: 3D packing

We have several small boxes (cuboids, i.e. 3D rectangles) given as a list of triples of positive integers (length, width, height). We want to pack them all in a box of the shape of a cube. What is the smallest possible length of the side of such cube?

Each of the small boxes can be placed on any of their sides. Boxes can be placed on top of each other, but:

- the base of the top box must lie completely on another box,
- a box can only be placed on top of a box that is at least as heavy, assuming uniform density.

Exercise: Minimum common substring partition

We are given two strings. We want to partition the strings into substrings so that each string can be obtained from the other by rearranging its parts (if this is possible). The goal is to find a partition into the smallest number of parts.

For example, if `s1 = GAGACTA` and `s2 = AACTGAG`, then the optimal partition has size 3: `s1 = GAG|ACT|A`, `s2 = A|ACT|GAG` (the instance is taken from this paper).

Exercise: DNA double digest

Using enzyme A, a DNA sequence is cut at certain points into a number of fragments. Using enzyme B, it is cut at possibly (but not neccessarily!) different points and into a possibly different number of pieces. We know the lengths of the fragments when using enzyme A, enzyme B, and both the enzymes at once. Find a permutation of the fragments using enzyme A, and a permutation of the fragments using enzyme B, that fits with the fragments using both enzymes.

See this presentation for more details and the following sample instance:

```
A = {375, 282, 2205, 746, 2352, 9040}
B = {3518, 1887, 389, 5916, 2017, 1273}
AB = {375, 282, 2205, 656, 90, 1797, 389, 166, 5750, 2017, 1273}
```

for which a possible solution is `([3, 0, 1, 5, 2, 4], [1, 4, 3, 0, 2, 5])`.

Exercise: Tower of Hanoi

Solve the Tower of Hanoi puzzle for an arbitrary number of pegs and discs. Use the `planner` module. This is Exercise 6.9/7 in the book.