OPT-B BED OPTIMIZATION

“DEVELOP AN ALGORITHM, THAT ASSIGNS A SET OF ELEMENTS (RECTANGLES) TO A MINIMAL NUMBER OF BEDS”

Bertiana Balliu & Mario Comboni

Version 1: brute force **NO** 49 elements → 6 *10^62 possible permutations

~~Brute force: algorithm parts:~~

1. ~~State of all possible «bed lists» → all permutations of elements charging order : n!~~
2. ~~Verify the «stack constraint» for each «bed list» → limit of opened stacks : eg. Max 2~~
3. ~~Unify each 2+ beds respecting the «bed constraint» → elements fit in the bed~~

1° Algorithm - optBeds

```
optBeds ( el[] Elements, ebd[] Beds , stack[] Stacks )
        forEach el in Elements:
                checkStackConstraint()
                if checkStackConstraint==True ; checkConcreteConstrain();

        end
```

2° Algorithm - checkStackConstraint()

```
checkStackConstraint ( Element, ListaStack, Pass)
            if el.stackId == OpenedStackId
                  then
                        listaStacks.remininingElements – 1
                        if listaStacks.remininingElements ==0
                                    stackFalg = close
                        Pass = TRUE
            else        if OpenedStacks < maxStacs
                        then        openedStacks +1
                                    stack (el.stack).stackFal = open
                                    stack (el.stack).remainingElement = -1
                                    Pass = TRUE
                        else
                                    Pass = FALSE
```

3° Algorithm - checkConcreteConstrain()

```
checkConcreteConstraint ( Element , bed)
    if el.concrete =! bed.concrete
        then
            openNewBed.id +1
    insertElementIntoBed()


insertElementIntoBed()
    TODO (included the optimization of the number of beds)
```

New objects

```
Stack: {  id;
          stackFalg
          remininingElements
           }


OpenedStacks: { }
```