5.313 sort

DESCRIPTION LINKS GRAPH

Origin [267]

Constraint sort(VARIABLES1, VARIABLES2)

Synonyms sortedness, sorted, sorting.

Arguments

VARIABLES1	:	collection(var−dvar)
VARIABLES2	:	collection(var−dvar)

Restrictions

|VARIABLES1| = |VARIABLES2|
required(VARIABLES1, var)
required(VARIABLES2, var)

Purpose The variables of the collection VARIABLES2 correspond to the variables of VARIABLES1 according to a permutation. The variables of VARIABLES2 are also sorted in increasing order.

Example

$$\left\langle \begin{array}{c} \langle \mathtt{var}-1, \mathtt{var}-9, \mathtt{var}-1, \mathtt{var}-5, \mathtt{var}-2 \rangle, \\ \mathtt{var}-1, \\ \langle \mathtt{var}-1, \mathtt{var}-1, \mathtt{var}-2, \mathtt{var}-5, \mathtt{var}-9 \rangle \end{array} \right\rangle$$

The sort constraint holds since:

- Values 1, 2, 5 and 9 have the same number of occurrences within both collections $\langle 1, 9, 1, 5, 2, 1 \rangle$ and $\langle 1, 1, 1, 2, 5, 9 \rangle$. Figure 5.557 illustrates this correspondence.
- The items of collection $\langle 1, 1, 1, 2, 5, 9 \rangle$ are sorted in increasing order.

Symmetries

- Items of VARIABLES1 are permutable.
- One and the same constant can be added to the var attributes of all items of VARIABLES1 and VARIABLES2.

Remark A variant of this constraint was introduced in [409]. In this variant an additional list of domain variables represents the permutation that allows to go from VARIABLES1 to VARIABLES2.

Algorithm [71, 252].

Systems `sorting` in **Choco**, `sorted` in **Gecode**, `sorting` in **SICStus**.

See also **generalisation:** `sort_permutation` *(*`PERMUTATION` *parameter added)*.
implies: `same`.

Keywords **characteristic of a constraint:** sort.
combinatorial object: permutation.
constraint arguments: constraint between two collections of variables.
filtering: bound-consistency.

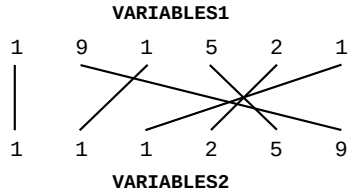


Figure 5.557: Correspondence between collection $\langle 1, 9, 1, 5, 2, 1 \rangle$ and collection $\langle 1, 1, 1, 2, 5, 9 \rangle$

1606 $\overline{\mathbf{NSINK}}$, $\overline{\mathbf{NSOURCE}}$, CC($\overline{\mathbf{NSINK}}$, $\overline{\mathbf{NSOURCE}}$), $PRODUCT$; $\overline{\mathbf{NARC}}$, $PATH$

Arc input(s)	VARIABLES1 VARIABLES2
Arc generator	$PRODUCT \mapsto$ collection(variables1, variables2)
Arc arity	2
Arc constraint(s)	variables1.var = variables2.var
Graph property(ies)	• for all connected components: **NSOURCE**=**NSINK** • **NSOURCE**= |VARIABLES1| • **NSINK**= |VARIABLES2|

Arc input(s)	VARIABLES2
Arc generator	$PATH \mapsto$ collection(variables1, variables2)
Arc arity	2
Arc constraint(s)	variables1.var $\leq$ variables2.var
Graph property(ies)	**NARC**= |VARIABLES2| − 1

Graph model

Parts (A) and (B) of Figure 5.558 respectively show the initial and final graph associated with the first graph constraint of the **Example** slot. Since it uses the **NSOURCE** and **NSINK** graph properties, the source and sink vertices of this final graph are stressed with a double circle. Since there is a constraint on each connected component of the final graph we also show the different connected components. The sort constraint holds since:

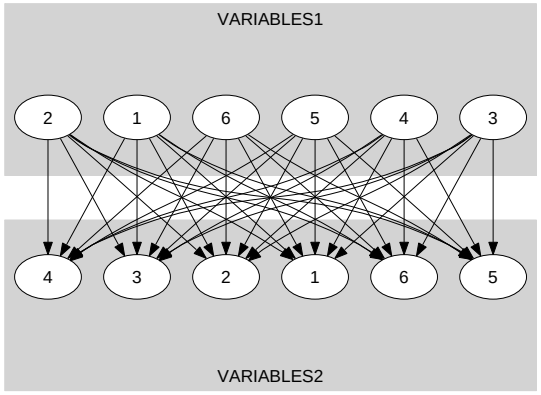
- Each connected component of the final graph of the first graph constraint has the same number of sources and of sinks.
- The number of sources of the final graph of the first graph constraint is equal to |VARIABLES1|.
- The number of sinks of the final graph of the first graph constraint is equal to |VARIABLES2|.
- Finally the second graph constraint holds also since its corresponding final graph contains exactly |VARIABLES1 − 1| arcs: all the inequalities constraints between consecutive variables of VARIABLES2 holds.

Signature

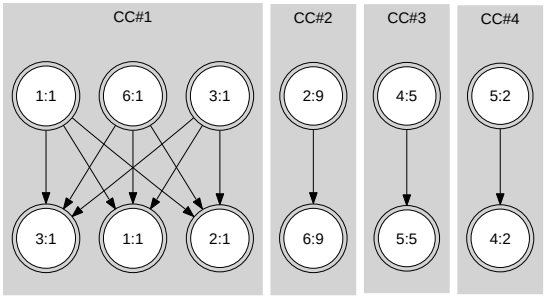
Consider the first graph constraint. Since the initial graph contains only sources and sinks, and since isolated vertices are eliminated from the final graph, we make the following observations:

- Sources of the initial graph cannot become sinks of the final graph,
- Sinks of the initial graph cannot become sources of the final graph.

From the previous observations and since we use the $PRODUCT$ arc generator on the collections VARIABLES1 and VARIABLES2, we have that the maximum number of sources and sinks of the final graph is respectively equal to |VARIABLES1| and |VARIABLES2|. Therefore we can rewrite **NSOURCE** = |VARIABLES1| to **NSOURCE** $\geq$ |VARIABLES1| and simplify $\overline{\mathbf{NSOURCE}}$ to $\overline{\mathbf{NSOURCE}}$. In a similar way, we can rewrite **NSINK** = |VARIABLES2| to **NSINK** $\geq$ |VARIABLES2| and simplify $\overline{\mathbf{NSINK}}$ to $\overline{\mathbf{NSINK}}$.



(A)



CC#1:NSOURCE=3,NSINK=3
CC#2:NSOURCE=1,NSINK=1
CC#3:NSOURCE=1,NSINK=1
CC#4:NSOURCE=1,NSINK=1
NSOURCE=6,NSINK=6

(B)

Figure 5.558: Initial and final graph of the `sort` constraint

Consider now the second graph constraint. Since we use the $PATH$ arc generator with an arity of 2 on the VARIABLES2 collection, the maximum number of arcs of the final graph is equal to $|\mathtt{VARIABLES2}| - 1$. Therefore we can rewrite the graph property $\mathbf{NARC} = |\mathtt{VARIABLES2}| - 1$ to $\mathbf{NARC} \geq |\mathtt{VARIABLES2}| - 1$ and simplify $\overline{\mathbf{NARC}}$ to $\overline{\mathbf{NARC}}$.