MiniZinc Exercises (optional)

Prepared by Gustav Björdal
1st September 2020

Array comprehensions are a crucial part of writing efficient MiniZinc models. This document contains 5 MiniZinc exercises to help you get started with writing more complex comprehensions. For a more basic introduction to array comprehensions, see the MiniZinc Tutorial section 2.2.1: https://www.minizinc.org/doc-latest/en/modelling2.html.

Skeleton code for the exercises is available at http://user.it.uu.se/~gusbj192/courses/M4CO/exercises/.

1 Repeating Numbers

Consider a parameter array $A$ of length $m$, and a constant $n$. Create, using an array comprehension, the array $B = [A[1], A[1], \ldots, A[2], A[2], \ldots, A[m], A[m], \ldots]$ of length $n \cdot m$ containing $n$ repetitions of each element of $A$.

**Example:** $A = [1,2,1]$ and $n = 2$ should give $B = [1,1,2,2,1,1]$.

Listing 1: Skeleton code for Ex1.mzn

```plaintext
int: n = 2;
int: m = 3;
array[1..m] of int: A = [1,2,1];
array[1..m*n] of int: B = [... | ...];
```

2 Repeating Numbers Again

Consider a parameter array $A$ of length $m$, and a constant $n$. Create, using an array comprehension, the array $B = [A[1], A[2], \ldots, A[m], A[1], A[2], \ldots, A[m], \ldots]$ of length $n \cdot m$ consisting of $n$ concatenations of $A$.

**Example:** $A = [1,2,3]$ and $n = 3$ should give $B = [1,2,3,1,2,3,1,2,3]$.

Listing 2: Skeleton code for Ex2.mzn

```plaintext
int: n = 3;
int: m = 3;
array[1..m] of int: A = [1,2,3];
array[1..m*n] of int: B = [... | ...];
```
3 Counting

Consider a parameter 2D array $A$ of $n$ rows and $m$ columns. Create, using array comprehension, the array $B$ of length $n$ where each index $i$ in $B$ is the number of occurrences of the value 5 in row $i$ of $A$.

**Example:** $A = [1,2,3|4,5,6|7,8,9|5,5,5]$ should give $B = [0,1,0,3]$.

Listing 3: Skeleton code for Ex3.mzn

```mzn
int: n = 4;
int: m = 3;
array[1..n, 1..m] of int: A = [|1,2,3|4,5,6|7,8,9|5,5,5|];
array[1..n] of int: B = [ ... | ... ];
```

4 Expressions on Variables

Consider an array $X$ of $n$ variables. Define, using an array comprehension, the array $Y$ of the absolute differences between each distinct unordered pair of distinct variables in $X$. Create also the index set of array $Y$ (or use the keyword `int`). By constraining both $X$ and $Y$ to each have all-different values, this becomes a model for a Golumb ruler.

**Example:** If $X = [8, 4, 2, 1]$, then $Y$ should contain the values 4, 6, 7, 2, 3, and 1.

Listing 4: Skeleton code for Ex4.mzn

```mzn
include "globals.mzn";
int:n = 4;
array[1..n] of var 1..n*n: X;
array[???] of var int: Y = [... | ...];
constraint alldifferent(X);
constraint alldifferent(Y);
```

5 Sorting

Consider a parameter array $A$ and a parameter array $B$, both of length $n$. Create, using an array comprehension, the array $C$ containing the values of $B$ but sorted as if it was $A$. That is, the value $B[i]$ occurs before $B[j]$ in $C$ if and only if $A[i] \leq A[j]$.


Listing 5: Skeleton code for Ex5.mzn

```mzn
include "globals.mzn";
int: n = 3;
array[1..n] of int: A = [2,1,3];
array[1..n] of int: B = [9,7,5];
array[1..n] of int: C = [... | ...];
```