Computer Science 384
St. George Campus

Version S22.1
University of Toronto

Programming Assignment: CSPs

Silent Policy: A silent policy will take effect 24 hours before this assignment is due, i.e. no question about this assignment will be answered, whether it is asked on the discussion board, via email or in person.

Late Policy: Late submissions will not be accepted unless grace days are used.

Submission Instructions: You must submit your assignment electronically through MarkUs. You will submit the following files:

- `propagators.py`
- `models.py`

Your login to MarkUs is your teach.cs username and password. It is your responsibility to include all necessary files in your submission. You can submit a new version of any file at any time. Only your latest submission will be considered. Ensure that:

- your code runs on teach.cs using `python3` (version 3.10) using only standard imports. Your code will be tested using this version and you will receive zero marks if it does not run using this version.
- you do not add any non-standard imports from within the `Python` file you submit (the imports that are already in the template files must remain). Once again, non-standard imports will cause your code to fail the testing and you will receive zero marks.
- you do not change the supplied starter code. Your code will be tested using the original starter code, and if it relies on changes you made to the starter code, you will receive zero marks.

Clarifications: Important corrections (hopefully few or none) and clarifications to the assignment will be posted on Quercus. You are responsible for monitoring the clarification page.

Questions: Questions about the assignment should be posted to Piazza.

Introduction

Amazon has hired you to program their new fleet of warehouse robots to store packages. The warehouse is an $N \times N$ grid divided into several buildings. Each building is further divided into rooms. Each room consists of several storage locations. You decide to break this task up into two parts:

1. Identify how many packages should be stored in each room.
2. Move the packages into the the storage points in a given room.

In this assignment, we will be focusing on the first part.

There are several types of rooms in the warehouse, but none of them have a capacity of more than N. In fact, most rooms have a capacity of exactly N. However, the rooms in some buildings have a smaller capacity. Moreover, some rooms have a minimum occupancy requirement. The rooms can vary from building to building, but all of the rooms in any particular building are the same. For legal reasons, some buildings must hold a specific number of packages and the rooms in any given row/column of the warehouse must contain a different number of packages.

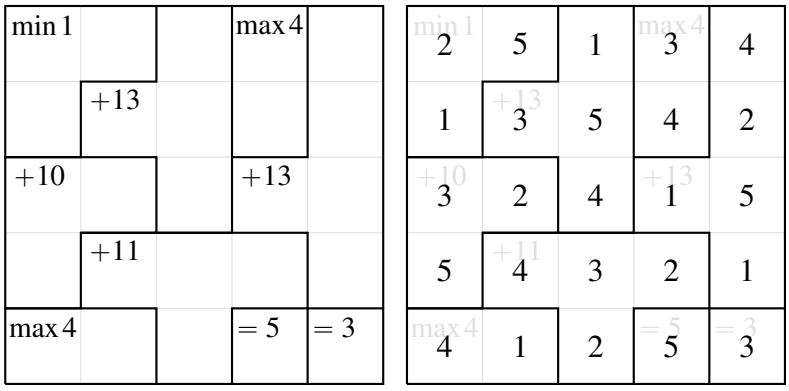

Figure 1: Example of a 5×5 warehouse with 9 buildings (left) and the number of packages that should go in each room (right).

Formally, the problem is encoded using a list of lists;

- the first list contains a single number, namely, N, and
- each following list represents a room; it contains a sequence of cells, followed by an operation chosen from $\{0,1,2,3\}$ (0 means $=$, 1 means $+$, 2 means min, 3 means max) and a target value, v.

The cell in the i^{th} column and j^{th} row is indexed, ij, where $i, j \in \{1,\ldots,N\}$. For example, the board in Figure 1 is represented as

```
((5),(11,21,3,4),(12,13,23,1,10),(14,15,25, 2,1),(31,22,32,42,1,11),
      (33,34,35,24,1,13), (41,0,5), (51,0,3),(52,43,53,54,55,1,13), (44,45,3,4))
```

Our robots should organize everything before the supervisor arrives, which could be very soon. This means that they should aim to complete the task as quickly as possible.

The Starter Code

The code for this assignment consists of several `Python` files, some of which you will need to read and understand in order to complete the assignment. You have been provided:

1. `cspbase.py`; provides a generic framework to formulate and solve a CSP
2. `propagators.py`; should contain routines to perform constraint propagation in a CSP
3. `models.py`; should define a specific CSP (the warehouse problem in this case)
4. `propagators_test.py`; a testing script to verify the correctness of the propagators
5. `models_test.py`; a testing script to verify the correctness of the models

The only files you will submit are `propagators.py` and `models.py`. We consider the other files to be starter code, and we will test your code using the original versions of those files. Therefore, when testing your code, you should not modify the starter code.

Your Tasks

For this assignment, you must complete the following tasks:

1. Build the constraint propagators for a generic CSP:
   (a) Implement a propagator function, `prop_FC(csp, newVar)`, that propagates that checks constraints that have exactly one uninstantiated variable in their scope, and prune appropriately. If `newVar` is `None`, check all constraints. Otherwise only check only constraints whose scopes contain `newVar`.
   (b) Implement a propagator function, `prop_GAC(csp, newVar)`, that propagates that checks constraints that any variables in thier scope uninstantied, and prune appropriately. If `newVar` is `None`, check all constraints. Otherwise only check constraints whose scopes contain `newVar`.
2. Cast the warehouse problem as a CSP:
   (a) Formulate a CSP which models a relaxed version of the warehouse problem. In particular, consider the row/column constraints only and
      i. using only binary not-equal constraints (`warehouse_binary_ne_grid`)
      ii. using only n-ary all-different constraints (`warehouse_nary_ad_grid`)
   (b) Formulate a CSP which models the warehouse problem (`warehouse_full_model`). For the row/column constraints, you can use either binary not-equal or n-ary all-different constraints.