Homework 2 – CSP

TA Zae Myung Kim (zaemyung@kaist.ac.kr)
TA Jonghwan Hyeon (hyeon0145@kaist.ac.kr)

Date assigned: Wed 23 September 2015
Date due: Wed 7 October 2015

Submit one zip file containing your report and program codes via KLMS (naming format of the zip file: <student#>_<name>.zip, for example "20150724_john.zip"). Late submissions will not be accepted. (Write your name and student ID in your report.) This homework should be done individually and written in English.

1. Solving a seating arrangement puzzle as a CSP (50 points)

At an annual alumni gathering, the following five people – **Frank, Clark, Karen, Isabel and Elly** – must be seated in a row.

However, some of these people are not on good terms with each other, and do not want to seat next to each other:

- No two people share the same chair.
- Clark does not like Karen.
- Elly does not like Frank and Karen.
- Neither Karen nor Elly likes Isabel.
- But Frank likes Isabel, and wants to sit at her left side.
- But Isabel likes Clark, and wants to sit at his left side.

If a person(A) does not like another person(B), then person(A) **must not** be seated next to the person(B). Assuming a bird's-eye view, i.e. you are looking down at the people from the top, we want to find a seating arrangement from left to right that satisfies everyone's (dis)liking.

We formulate this problem as a CSP, treating people as variables and the set of chair numbers as their domains. For example, "domain(K) = {1, 2, 3, 4, 5}" represents that all five chairs are available for Karen (in the current problem-solving state).

<u>Questions</u>

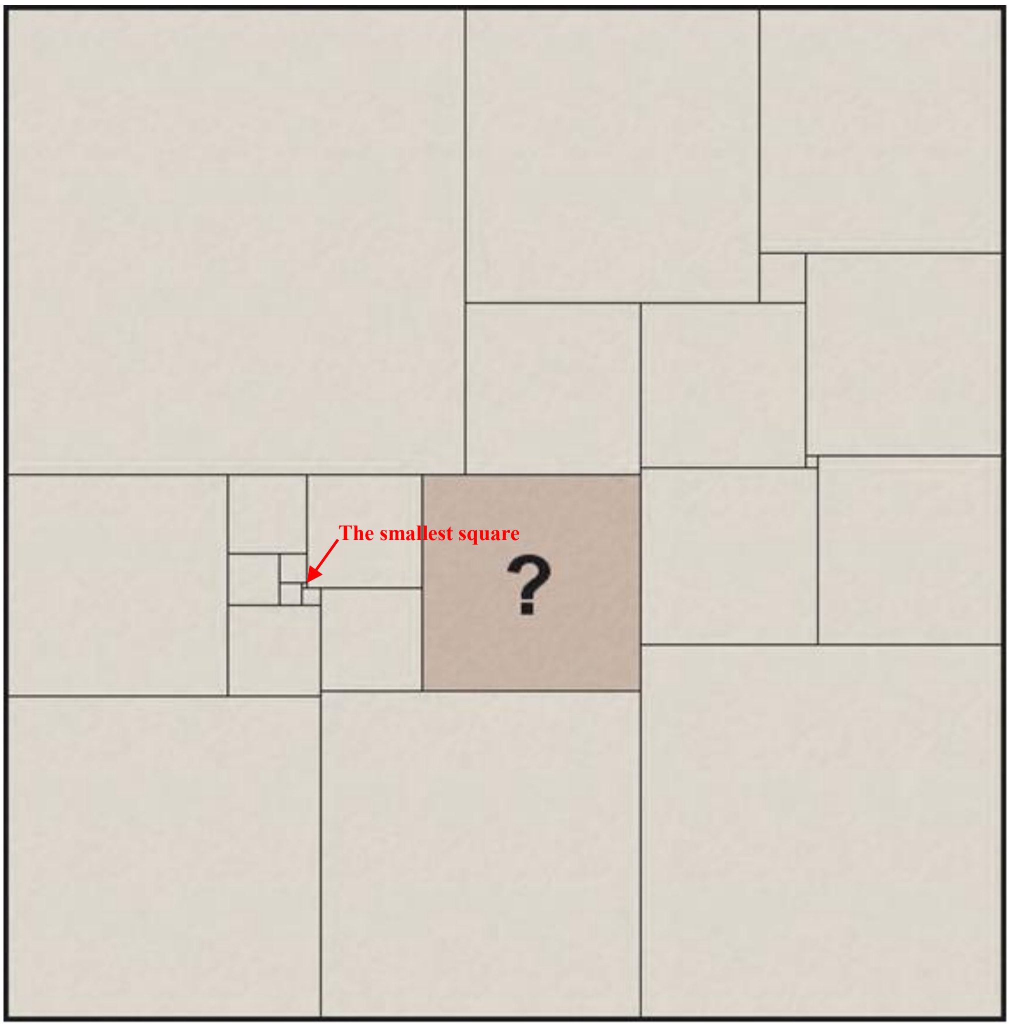
A. Let us represent "A sits next to B" arithmetically as "$|\boldsymbol{A} - \boldsymbol{B}| = \mathbf{1}$". By similar token and using inequality, identify all the constraints between any pair of people, and state these constraints as arithmetic expressions.
(You do not have to expand the first rule into multiple arithmetic expressions, but use a global constraint instead. You do not need to expand the determinants.) (14 points)

B. Enforce arc consistency fully on the domain of each variable. What are the remaining values in the domains? (15 points)

C. Keeping in mind the heuristics that we learned in the lecture, select the first variable to assign, and assign a value. Justify your answer fully. (6 points)

D. Continue solving the problem until the solution is found. At each stage, enforce arc consistency, choose a variable, and assign a value to the variable heuristically. Illustrate the process fully. (In case of a tie in values, pick the smallest value.) (15 points)

2. Finding the side length of the shaded square (50 points)



The square above consists of 24 smaller squares, and the size of each square is all different to each other. The goal is to find the side length of the shaded square. Assume that the side length of the smallest square is 1.

This puzzle is known to be one of the toughest ones to solve – that is, if it were to be solved by hands. We will solve this puzzle by modeling it as a CSP, and letting a CSP solver to do the hard work for us.

We utilize the Python *numberjack* module to solve this problem: http://numberjack.ucc.ie/
Install the module according to the instructions in http://numberjack.ucc.ie/download
To run this module, you need to have either Linux or Mac OS environment, and have the necessary packages listed in the website installed.
Before implementing, do familiarize yourself with the *numberjack* module by trying out examples

presented in the website. You are also welcome to look at its API documentation as well: http://numberjack.ucc.ie/doc/

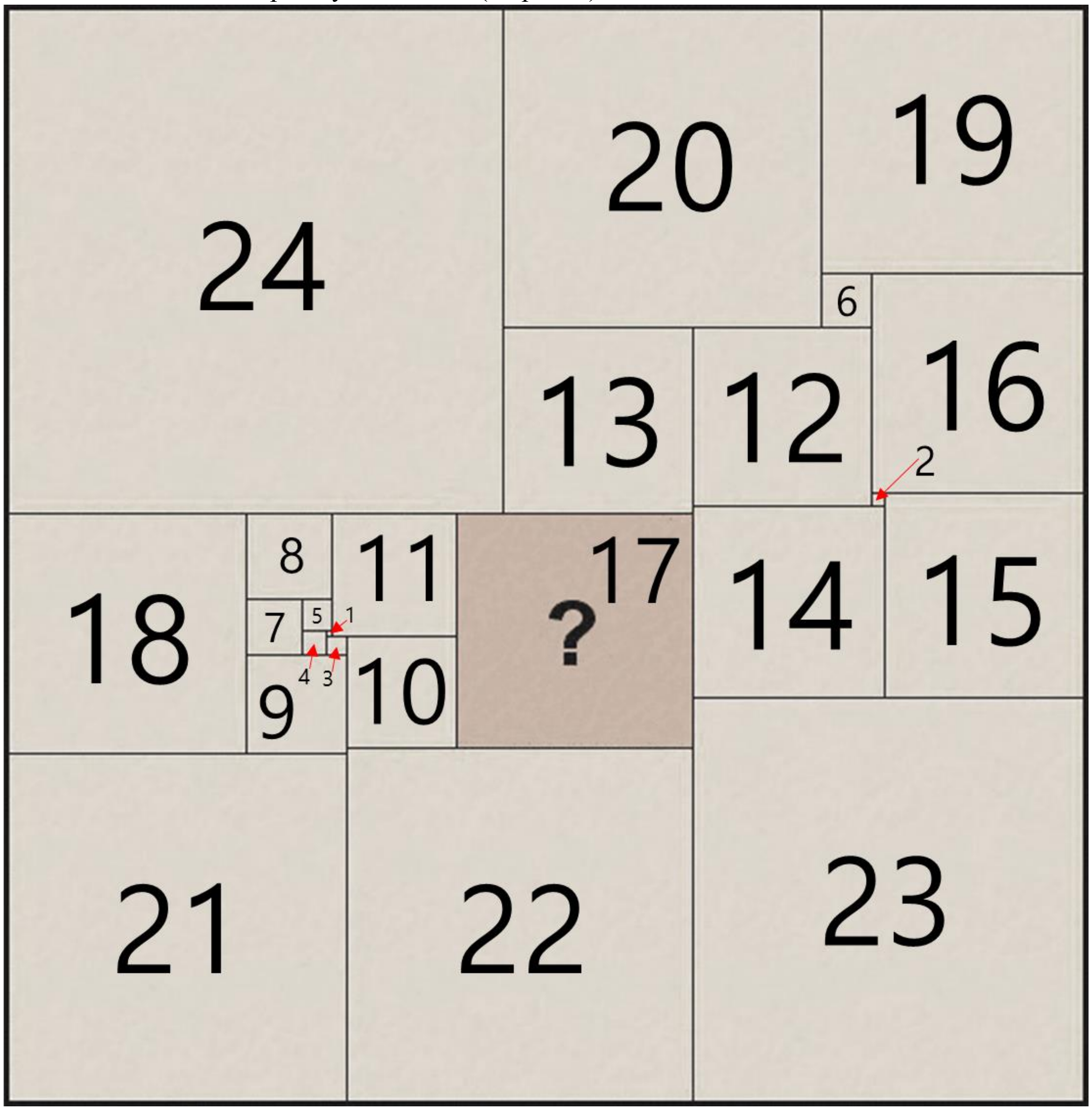
Questions

A. Implement the **def model(solver, Top, N)** function in the *square.py* file, and solve the puzzle. (30 points)

   You need to consider two kinds of constraints that define the puzzle:

   - Constraints on the **length of squares**
   - Constraints on the **area of squares**

B. The diagram below shows the ascending order of square sizes. In your **def model**, devise and add the ordering constraints. Does the addition of order constraints make the search faster/slower? Explain your answer. (20 points)



24
20
19
6
13
12
16
2
18
8
11
17
?
14
15
7
5
1
4
3
9
10
21
22
23