Discrete Optimization Assignment:
Puzzle Challenge (Extra Credit)

1 Problem Statement

Throughout the course interesting puzzles are used to illustrate various techniques (e.g. dual modeling, symmetry breaking, redundant constraints, and global constraints, to name a few). This assignment gives you an opportunity to experiment with those techniques directly on the problems discussed in the lectures.

In this assignment you are presented with four puzzles, N-Queens, All Interval Series, Magic Series, and Magic Square. All of these puzzles are feasibility problems. That is, you either have a solution or you do not, there is no middle ground. This differs from the other assignments where a trivially feasible low quality solution is often easy to construct. All of these puzzles are arbitrary in size. That is, for any positive integer n, you can construct a puzzle of size n. The goal of this assignment is to produce the a solution to each puzzle for a very large value of n. You can earn up to 20 extra credit points on each puzzle, and together all of the puzzles can replace one whole assignment! The following sections described each puzzle in detail.

2 N-Queens

2.1 Assignment

This problem is mathematically formulated in the following way: We are given a positive integer n. We are to place $N = 0 \dots n-1$ queens on an $n \times n$ chess board so that no two queens attack each other. Notice that each queen can be assigned to a particular column (otherwise they would attack each other). For each queen $i \in N$, place that queen in column i and let $x_i \in N$ be the row that the queen is placed on. Using this representation, the n-queens problem must satisfy the following constraints,

$$\begin{array}{ll} x_i \neq x_j & (i \in N, j \in N, i < j) \\ x_i \neq x_j + (j-i) & (i \in N, j \in N, i < j) \\ x_i \neq x_j - (j-i) & (i \in N, j \in N, i < j) \end{array}$$

The first constraints prevent two queens from being placed on the same row. The remaining constraints prevent the queens from attacking on the diagonals.

2.2 Data Format Specification

The output has 2 lines. The first line contains one value n, the size of the problem solved. The next line is a list of n values in N, one for each of the x_i variables. This line encodes the solution.

Output Format

```
|N|
x_0  x_1 ... x_|N|-1
```

Output Example

```
5
0 2 4 1 3
```

This output represents the following solution: the first queen placed in row 0, the second queen placed in row 2, the third queen placed in row 4, the fourth queen placed in row 1, and fifth queen placed in row 3.

3 All Interval Series

3.1 Assignment

This problem is mathematically formulated in the following way: We are given a positive integer n. We are to find a permutation of the values $N = 0 \ldots n-1$ such that the absolute difference between adjacent values are all different. Let x_i denote the permutation of N, then the all interval series problem must satisfy the following constraints,

$$\begin{array}{ll} |x_i - x_{i+1}| \neq |x_j - x_{j+1}| & (i \in N \setminus n-1, j \in N \setminus n-1, i < j) \\ x_i \text{ are a permutation of } N & \end{array}$$

The first constraints ensure all of the adjacent differences are unique. The remaining constraint ensures that x_i is a permutation.

3.2 Data Format Specification

The output has 2 lines. The first line contains one value n, the size of the problem solved. The next line is a list of n values in N, one for each of the x_i variables. This line encodes the solution.

Output Format

```
|N|
x_0  x_1 ... x_|N|-1
```

Output Example

```
5
0 4 1 3 2
```

We can check that this is an all interval series by computing the absolute difference of all the values: $|0-4| = 4, |4-1| = 3, |1-3| = 2, |3-2| = 1$ and checking that all the values are different.

4 Magic Series

This problem is mathematically formulated in the following way: We are given a positive integer n. Let x_i for $i \in N = 0 \ldots n-1$ be an array. We are to assign values to x_i so that the number of occurrences of i in the array is equal to the value of x_i. In other words, a magic series of size n must satisfy the following constraints,

$$x_i = \sum_{j \in N} (x_j = i) \quad (i \in N)$$

4.1 Data Format Specification

The output has 2 lines. The first line contains one value n, the size of the problem solved. The next line is a list of n values in N, one for each of the x_i variables. This line encodes the solution.

Output Format

```
|N|
x_0  x_1 ... x_|N|-1
```

Output Example

```
5
2 1 2 0 0
```

We can check that this is a magic series by counting the occurrences of each value in the solution. The solution has 2 zeros, 1 one, and 2 twos. This matches what is indicated by the solution.

5 Magic Square

This problem is mathematically formulated in the following way: We are given a positive integer n, let $N = 0 \dots n-1$. We are to assign a the values $M = 1 \dots n^2$ to an $n \times n$ square such that the sum of the rows, columns and diagonals are all the same.[1] Let x_{ij} be the value stored in i-th row and j-th column of the square, then the all interval series problem must satisfy the following constraints,

$$\begin{aligned}
&\sum_{j \in N} x_{ij} = \frac{n(n^2+1)}{2} && (i \in N)\\
&\sum_{j \in N} x_{ji} = \frac{n(n^2+1)}{2} && (i \in N)\\
&\sum_{i \in N} x_{ii} = \frac{n(n^2+1)}{2} &&\\
&\sum_{i \in N} x_{i,n-i-1} = \frac{n(n^2+1)}{2} &&\\
&x_{ii} \le x_{i+1,i+1} && (i \in N \setminus n-1)\\
&x_{ij} \text{ are a permutation of } M &&
\end{aligned}$$

The first two constraints ensure that the rows and columns sum to the magic number, respectively. The following two constraint ensure that the lower and upper diagonals sum to the magic number, respectively. The second to last constraint requires the download diagonal is increasing, and the last constraint ensures that the values are a permutation of M.

5.1 Data Format Specification

The output has $|N|+1$ lines. The first line contains one value n, the size of the problem solved. The following $|N|$ lines represent the rows of the magic square solution. Each line contains $|N|$ values from the set M, thus making a square. Per the problem definition, each value in M should appear in exactly one of these lines.

Output Format

```
|N|
x_0_0 x_0_1 ... x_0_|N|-1
x_1_0 x_1_1 ... x_1_|N|-1
...
x_|N|-1_0 x_|N|-1_1 ... x_|N|-1_|N|-1
```

Output Example

```
3
2 7 6
9 5 1
4 3 8
```

[1]We know that the sum must be the magic number, $\frac{n(n^2+1)}{2}$.

We can verify that this is a magic square by checking the sum of each row, column, and diagonal is equal to the magic number, $\frac{3(3^2+1)}{2} = 15$.

6 Instructions

The assignment comes with one solver for each of the problems. Edit the appropriate solver source file,

`queensSolver.py`, `allIntervalSeriesSolver.py`, `magicSeriesSolver.py`, `magicSquareSolver.py`

and modify the `solveIt(n)` function to solve the problems described above. The function argument, `n`, is an integer representing the size of problem to solve. The return value of `solveIt` is a solution to the problem in the output format described above. Your `solveIt` implementation can be tested with a command similar to,

```
python ./queensSolver.py <n>
```

You should limit the `solveIt` method to terminate within 5 hours, otherwise the submission will not be eligible for full credit. You may choose to implement your solver directly in python or modify the `solveIt` function to call an external application.

Resources Each solver includes a very naive depth-first search, which can solve problems for $n = 3 \ldots 5$, but not much more.

Handin Run `submit.pyc` with the command, `python ./submit.pyc`. Follow the instructions to run your `solveIt` method on a particular problem for a given size. You can submit multiple times and your grade will be the best of all submissions. However, it may take several minutes before your assignment is graded; please be patient. You can track the status of your submission on the *feedback* section of the assignment website.

Grading Infeasible solutions (i.e. those that do not conform to the output format or violate problem constraints) will receive 0 points. Feasible solutions for n-queens, all interval series, and magic series will receive $\min(\log_2(n), 20)$ points. Feasible solutions for magic squares will receive $\min(2\log_2(n), 20)$ points, as the problem grows quadratically with n. If your solution is infeasible, the grading feedback will indicate which constraints your solution violated.

Collaboration Rules In all assignments we encourage collaboration and the exchange of ideas on the discussion forums. However, please refrain from the following:

1. Posting code or pseudo-code related to the assignments.
2. Using code which is not your own.
3. Posting problem solutions.

Discussion of solution quality (i.e. objective value) and algorithm performance (i.e. run time) is allowed and the assignment leader board is designed to encourage such discussions.

Warnings

1. You cannot rename any of the the `*Solver.py` files or the `solveIt` methods.
2. `*Solver.py` must remain in the same directory as `submit.pyc`.

7 Technical Requirements

You will need to have python 2.7.x installed on your system (installation instructions, `http://www.python.org/getit/`).