CoinPuzzle

We are going to implement a solution to a fun little coin puzzle. (See the other pdf with a scan from a classic puzzle book.) The idea is that you start with a row of squares, in which each can hold one coin. There are pennies and dimes on the squares and one blank square. You are allowed just two kinds of moves, slides and jumps. A **slide** move is where a coin adjacent to the blank space is slid into that space. A **jump** move is where a coin jumps over a neighboring coin into the blank space.

The standard version of this puzzle has five squares, two pennies on one side, two dimes on the other side, and the blank space in the middle. The challenge is to swap the pennies and dimes in only eight moves.

In our version, we will consider a more general setting in which there could be other numbers of pennies and dimes. The arrangement of these pennies and dimes is called a **configuration**. The goal will be to find the fewest number of moves to get from one configuration to another.

The Lab

Make a class that will store a configuration.

Configuration:

- `__init__(self, t)` - initialize a new configuration from an iterable `t`. Use the convention that pennies are `1`, dimes are `10` and he blank space is `0`. For example, a valid configuration is `(1,0,10,1)`. There are no conditions on the length of `t`.
- `moves(self)` - returns an iterable of configurations that can be obtained in one move from the self.
- `__eq__(self, other)` - two configurations are equal if they have the same tuple
- `__hash__(self)` - hash the configuration based on its tuple

What to submit

You should submit a file called `configuration.py` containind the code for your `Configuration` class.

Working towards the homework problem

Once you get a working configuration class. Begin working on a function that can generate all possible configurations for a given number of pennies and dimes. Assume that there is only one blank space. There are a couple ways to do this. Recursion might be a good choice. This function will not be tested in the lab.

In the homework assignment, we will build the graph of all configurations, where the edges are valid moves.