Two different algorithms were used in this project.

#1
The first (and less successful) algorithm worked as follows: For every constraint (w1, w2, w3), we know that w3 doesn't go in between w1 and w2. For these three wizards, there are also 3! = 6 possible orderings for their ages. e.g. w1 < w2 < w3. This first algorithm enumerated all possible conditions in that form, using a representative Base system that used num_wizards as the base (every number represents a condition), then got rid of them accordingly i.e. for constraint (w1, w2, w3), the algorithm would get rid of the numbers equivalent to conditions w1 < w3 < w2 and w2 < w3 < w1. After using up all of these conditions, it arbitrarily assumed certain remaining conditions (e.g. w1 < w2) were true, and then eliminated possibilities in that manner, eventually reaching a state where an order could be reconstructed without violating any of the assumed conditions.

#2
The second (more successful) algorithm worked by a randomized greedy-esque method. It began by randomizing an order for the wizards, and checking how many constraints it satisfied. It then got rid of the bottom half of orderings (by how many constraints were satisfied) and filled in the now-open spots with slightly modified versions of the successful orderings. The process was repeated until a satisfactory level, and a slightly more rigorous swapping algorithm (swapping all possible pairs, then triples in a particular order) applied at the end if all constraints had yet to be satisfied to maximize score.

The first algorithm solved entirely 4 or 5 of the input20s, and achieved 90%+ on everything else (below additional staff inputs). The second algorithm solved entirely all but one input20 and a few input 50s. It solved 2 of the additional staff inputs.