I tried to do the main loop in a recursive way, but it seems that Java gives up at one point: it returns a StackOverFlow error.

Therefore, it seems more appropriate to use a while loop, where in theory Java won't stop.

Note: the recursive method seems to work with a very initial population.

My guess is that the smaller the population is, the faster the loop executes but it requires more iteration (so many that Java returns Overflow error).

With a big population, I could clearly see that the loop was slower, but the number of itireations necessary to find the solution was much lower. ( When I set the population size to 100000, it takes only about 10 itireations.)

I started by coding the crossover method very simple: just take the first half of the first parent, then take the second half of second parent, and merge. For example, the two parents

PHILOSOPHY

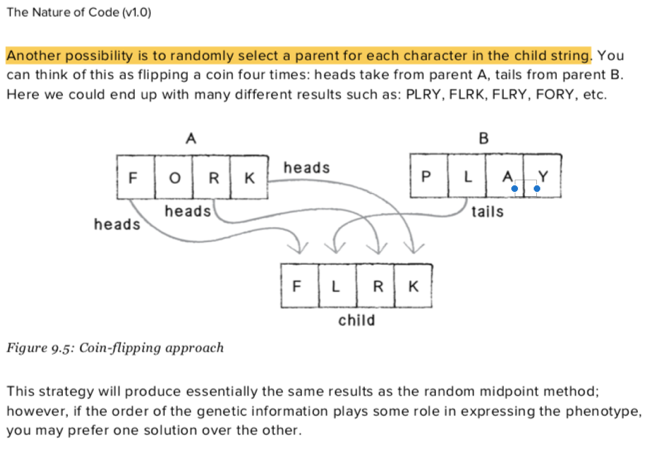
ASTONISHED

Illustration from the book

would have given the child

PHILOISHED.

However, the algorithm gets easily stuck with initial population < 100000. So I decided to change the crossover method.

I stumbled upon a book, called *The Nature of code* where I found another method: the coin-flipping approach.

For every character, the child will have 50% chance of inherit the character of parent 1 and 50% chance from parent2.

The book can be found online at <https://natureofcode.com>.