AP Computer Science
Programming Project
Genetic Algorithms

BREEDING GIANT RATS WITH GENETIC ALGORITHMS
- Adapted from *Impractical Python Projects*, Lee Vaughan

Genetic algorithms are general-purpose optimization programs designed to solve complex problems. Invented in the 1970s, they belong to the class of evolutionary algorithms, so named because they mimic the Darwinian process of natural selection. They are especially useful when little is known about a problem, when you're dealing with a nonlinear problem, or when searching for brute-force-type solutions in a large search space. Best of all, they are easy algorithms to grasp and implement.

Genetic algorithms optimize, which means that they select the best solution (with regard to some criteria) from a set of available alternatives. For example, if you're looking for the fastest route to drive from New York to Los Angeles, a genetic algorithm will never suggest you fly. It can choose only from within an allowed set of conditions that you provide. As optimizers, these algorithms are faster than traditional methods and can avoid premature convergence to a suboptimal answer. In other words, they efficiently search the solution space yet do so thoroughly enough to avoid picking a good answer when a better one is available.

Unlike exhaustive search engines, which use pure brute force, genetic algorithms don't try every possible solution. Instead, they continuously grade solutions and then use them to make "informed guesses" going forward. A simple example is the "warmer-colder" game, where you search for a hidden item as someone tells you whether you are getting warmer or colder based on your proximity or search direction. Genetic algorithms use a fitness function, analogous to natural selection, to discard "colder" solutions and build on the "warmer" ones. The basic process is as follows:

l. Randomly generate a population of solutions.

2. Measure the "fitness" of each solution.

3. Select the best (warmest) solutions and discard the rest.

4. Cross over (recombine) elements in the best solutions to make new solutions.

5. Mutate a small number of elements in the solutions by changing their value.

6. Return to step 2 and repeat.

The *select -> cross over -> mutate* loop continues until it reaches a top condition, like finding a known answer, finding a "good enough" answer (based on a minimum threshold), completing a set number of iterations, or reaching a time deadline. Because these steps closely resemble the process of evolution, complete with survival of the fittest, the terminology used with genetic algorithms is often more biological than computational.

Assumptions for this project:

We will ignore life span in this exercise (due to the breeding frequency of rats).

Probability of Male/Female gender at birth is 50%.

For each breeding cycle, each offspring will "inherit" a random weight between the weight of its parents. We will, however, introduce mutation through probabilistic behavior (i.e. each offspring has a 1% chance of mutation in terms of weight). If an offspring mutates, its weight will be a product of its current weight and a random number between [0.5, 1.2].

For each breeding cycle we will select the top 10 rats of each gender for the next breeding cycle.

Criteria for the next breeding cycle will be the largest males and females of the entire population (new offspring, as well as the previous mating pairs), however we must also ensure that the largest female is no larger than the smallest male (sexual dimorphism in terms of body weight).

Genetic Algorithm for Breeding Giant Rats

l. Randomly generate an initial population of rats.
   - Build a collection of 10 male rats that have a weight ± 30% of most common rat weight.
   - Build a collection of 10 female rats that have a weight ± 10% of most common rat weight.

2. Select the "best" rats.
   - Select the top 10 largest male rats as breeders for next generation.
   - Select the top 10 largest female rats as breeders for next generation.
   - Ensure that no female is larger than the smallest male.

3. Breed rats by randomly pairing a male and a female rat among the breeders.
   - The new generation of potential breeders will be made up of the litters of the breeding pairs, as well as the breeders themselves.
   - All offspring have a 1% probability of mutating in terms of weight, as discussed.

4. Return to step 2 and repeat until you have bred a male rat that weighs 50,000 grams! Or ~110 pounds!

Now go and channel your Inner Mad-Scientist!

Real-World Brown Rat Statistics

Minimum weight	200 grams
Average weight (female)	250 grams
Average weight (male)	300-350 grams
Maximum weight	600 grams
Number of pups in litter	8-12
Number of litters per year	4-13
Life span (wild, captivity)	1-3 years, 4-6 years

Algorithm Assumptions

Target weight	50,000 grams
Number of rats in lab	20 (10 male / 10 female)
Most common weight	300 grams
Odds of mutation	1%
Minimum mutate factor	50%
Maximum mutate factor	120%
Litter size	8
Maximum number of generations	1000

```
public class Rat implements Comparable<Rat>
{
  private String gender;
  private double weight;
  private int age;

  public Rat(String g, double wt)
  {
    gender = g.toUpperCase();
    weight = wt;
    age = 0;
  }

  public String getGender()
  {
    return gender;
  }

  public double getWeight()
  {
    return weight;
  }

  public int getAge()
  {
    return age;
  }

  public void addYear()
  {
    age++;
  }

  public int compareTo(Rat other)
  {
    if(getWeight() == other.getWeight())
      return 0;
    if(getWeight() > other.getWeight())
      return 1;
    return -1;
  }

  public String toString()
  {
    String s = (gender.equals("M")) ? "Male" : "Female";
    s += " rat weighting " + weight;
    s += " grams and is " + age + " in age";
    return s;
  }
}
```