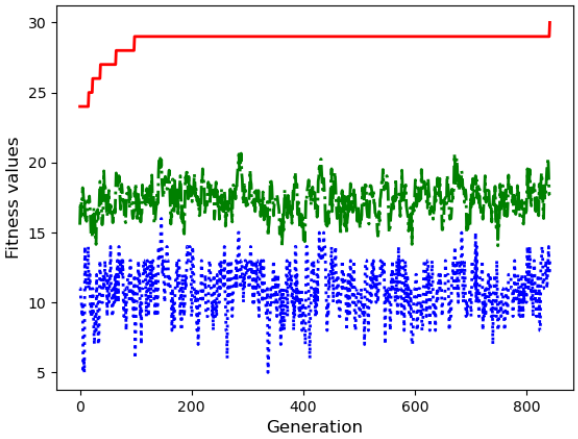
A few comments about the GA experiments

Q1 and Q3

The original version of GA.py does not implement the *_OnePointCrossover* method.

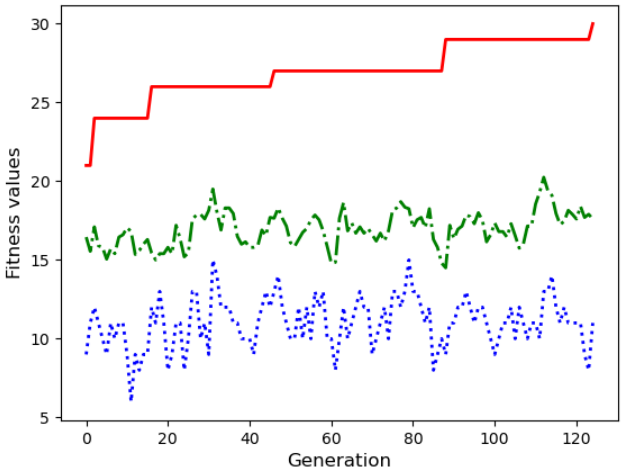
After implementing the one point crossover

```
def _onePointCrossover(self, other):
  point = random.randrange(1, self.length)
  self.genome[point:] = other.genome[point:]
```



generation: 840
best: 29
average: 19.15
worst: 14
Solved, exiting! Generations: 842

Over 10 runs, the quickest run was 124 generations



The algorithm converges after 100 generations at best. The average fitness increases slightly with the number of generations.

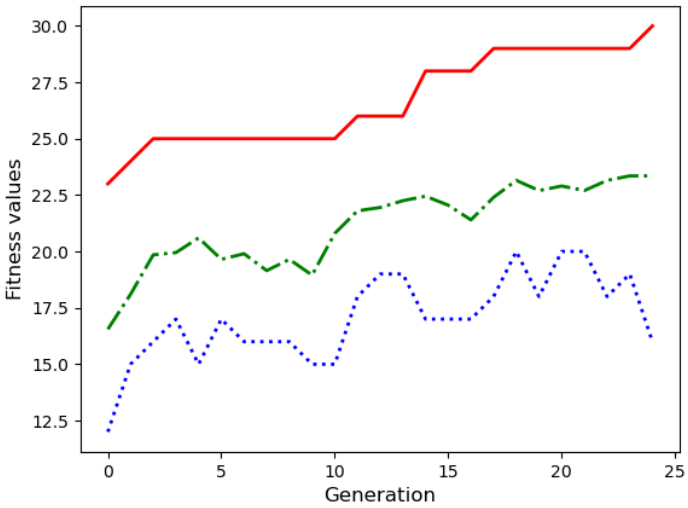
Q2

```
 def _printIndividual(self):
    print(f'fitness: {self.fitness}, genome: {self.genome}')
```

Q4

```
#select using a tournament
def tournamentSelect(select_best_prob=1.0):
  #randomly choose a set of TOURNAMENT_SIZE individuals
  #pick the best one from that set to be the child
  #return the selected child
  tourney = [random.choice(population) for i in range(TOURNAMENT_SIZE)]
  tourney.sort(key=lambda x: x.fitness, reverse=True)

  if random.random() < select_best_prob:
    return tourney[0]
  else:
    return random.choice(tourney[1:])
```



with tournament selection the average fitness increases significantly faster.

As the size of the tournament increases, the selected individuals are fitter. However, this might comes at the price of a lower diversity if the tournament becomes too large.

Q5
The combination mu:0.01, chi:0.2 seems to give the best results for this problem.

Q6
For the original values of mu and chi, the one-point crossover performs better.

Q8
When the elite becomes large, the population becomes less diverse and the speed is reduced.
If the ELITISM_NUMBER is 0, then the best solution can be forgotten.