Abstract:

This program contains mainly 3 parts. Initializing constants, dealing with input, running evolution and dealing with output. In this report I will go over each part in detail. In the end of this report you will find (almost) every resource I have used.

First: Initializing Constants

In initialize_constant() function I added different chord to CHORDS_NAMES to be latter used as a chromosome in my GA.
Then I added different keys and their chords into CHORD_PROGRESSION dictionary to be used latter to check correct chord progression in the generated accompaniment for a better fitness function

Second: Analyzing Input

In analyze_input() function it mainly extract all the needed information from the input to be latter used in fitness function
By converting the midi file of the input into a stream from Music21 library I am able to get the input's key, the key's correlation coefficient , notes played at each beat , minimum octave at each beat, and how many chords the accompaniment should contain.

Third: The GA:

The GA mainly consists of Generation , fitness function, crossover and mutation.

Generation:

My chromosome is a randomly chosen chord name done by generate_chromosome() function. And this chromosomes are used to make an individual that is a list of chord names , this individual is a representation of the accompaniment . This is done by generate_individual() function. Then several individual makes up my generation

Crossover:

In single_point_crossover() function i am doing a simple crossover by choosing a random point to divide the parents into two parts each . then returning two children composed from the parts of the parents

Mutation:

In mutation() function I am doing mutation with 50% probability
Where i substitute a chromosome with a randomly generated one

Fitness Function:

Indeed my favorite part and most challenging!
I wanted to measure mainly 3 things about the generated accompaniment.

1. Similarity with input melody:
    a. That was achieved in similarity_score() function. To illustrate mathematically how close the generated accompaniment to the input melody , I compare input key coefficient correlation in both of them by a builtin function from Music21.
    b. The equation I used is `score = ALPHA - ALPHA * abs(INPUT_KEY_CC - generated_key.correlationCoefficient)`
    c. Where alpha is some constant , and INPUT_KEY_CC is the input key correlationCoeff in input melody and generated_key.correlationCoeffiecnt is the input key correlationCoeff in generated accompaniment
2. Note Intersection:
    a. I noticed that when notes - of two chords playing in same beat - has intersection between them , the end results become better and more nicer to the ear
    b. So to get a mathematical idea of what that meant I used this equation:`score = 2 * ALPHA * cnt/(CHORDS_PER_MELODY * 3)`
    c. Where alpha is some constant and cnt is how many notes in the accompaniment intersect with input melody and CHORDS_PER_MELODY * 3 represent how many notes in the generated accompaniment
3. Chord progression:
    a. One of the most important aspects of generating any music is making sure that it follows a correct chord progression
    b. I have notices that the most common chord progression follow definite set of rules
    c. Where the progression starts with a tonic then goes to a dominant or subdominant (or even tonic again!) , a dominant chord goes to a tonic and a subdominant goes to a dominant chord
    d. And that what I am doing in correct_chord_progression_score() where I eliminate repetitive chords and sus chords, then I count how many times the accompaniment broke the chord progression either by a chord that doesn't

belong to the input key or by wrong sequence with chords that belong in the same key

e. The equation to represent this mathematically is ALPHA * (1 - (broke/len(gen))) where alpha is some constant and broke is how many times the accompaniment broken the chord progression and gen is list of chords in accompaniment without sus chords not repetitions

4. Finally , I add all this three parts to get a final fitness score

Fourth : Dealing with output

1. Printing progress:
   a. To demonstrate the status of the GA I used print_progress() function where i I compute the fitness of the whole population and print generation number , average fitness of the population, best fitness of the population and worst fitness.
2. Final output :
   a. In individual_to_midi() function I take an individual , the best one generated from GA, then converting it to a chordified stream from Music21 library. Then praising the input midi file to a stream also, then appending the two streams together.
3. Playing the magic!
   a. In play_music() I use it to play the result by using pygame library

Finally: recourses

1. Collab notebooks given in Labs
2. Wikipages for different kind of chords
3. Music21 library https://web.mit.edu/music21/doc/index.html
4. Pygame library https://www.pygame.org/news
5. Mido library https://mido.readthedocs.io/en/latest/midi_files.html
6. https://www.youtube.com/watch?v=sjddXIc_B20&t=329s
7. https://www.youtube.com/watch?v=zbPyKP3_pGo&t=439s
8. https://www.youtube.com/watch?v=uQj5UNhCPuo&t=508s
9. https://www.youtube.com/watch?v=MacVqujSXWE&t=134s
10. https://www.youtube.com/watch?v=nhT56blfRpE&t=34s
11. https://www.youtube.com/watch?v=aOsET8KapQQ&t=1s

12. (1118) AI Music Creativity 2020 Paper: Style Composition With An Evolutionary Algorithm - YouTube