Bijlage C: Persoonlijke opdracht 2 – Melody Maker

ℹ️ Korte uitleg over genetische algoritmen (GAs)

Genetische algoritmen zijn zoek- en optimalisatietechnieken geïnspireerd door natuurlijke selectie. Je werkt met een populatie van mogelijke oplossingen (chromosomen), beoordeelt ze met een fitness-functie, selecteert de beste individuen, combineert hun 'genen' via crossover, en introduceert variatie met mutatie. Door meerdere generaties evolueert de populatie naar steeds betere oplossingen.

🎯 Doel

Ontwikkelen van een eigen genetisch algoritme voor muzikale compositie, waarbij je inzicht krijgt in de werking en samenhang van de kernelementen van GAs: building blocks, mutatie, recombinatie, selectiedruk en diversiteit.

🎓 Leerdoelen

Praktische ervaring met de bouwstenen van een genetisch algoritme:

- Definiëren en manipuleren van building blocks
- Ontwerpen van mutatie- en recombinatie-operators
- Toepassen van selectiedruk via een fitness-functie
- Bewaken van diversiteit in de populatie
- Integreren van heuristieken om muzikale kwaliteit te sturen

🧠 Opdrachtomschrijving

Maak in de door jou gekozen programmeertaal een GA dat korte muzikale fragmenten ("building blocks") combineert tot een smaakvolle compositie. Jouw rol als omgeving: bepaal zelf welke composities een hoge fitness krijgen door ze een score te geven.

1. Building blocks
   - Vooraf gedefinieerde clichés, d.w.z. bepaalde clichématige muziekfragmenten die je vaak in populaire songs tegenkomt (zie Songfestival als treffend voorbeeld)
2. Mutatie & Recombinatie
   - Ontwerp mutatie operatoren op individuele maten of toonladders.
   - Implementeer recombinatie (één punt of uniform crossover) van goed scorende fragmenten.
3. Selectie
   - Gebruik tournementsselectie of roulette wielselectie om ouders te kiezen.
   - Behoud fragmenten die in meerdere goed scorende composities voorkomen.
4. Diversiteit
   - Zorg dat je basisverzameling van de buildingblocks breed genoeg blijft.
5. Heuristiek
   - Voorzie de fitness-functie van muzikale heuristieken(dat wil zeggen voorkennis over wat zoal voor velen goed klinkt), zoals: 8 of 16 maten van 4 tellen, afwisseling van coupletten en refreinen, II–V–I-progressies en toonsoorten beperkt tot de witte toetsen met een afsluiting op

🔧 Fase 1 – Basale GA

1. Implementeer een eenvoudige populatie van verzamelingen building blocks.
2. Definieer een fitness-functie waarin je handmatig scores toekent.
3. Voer selectie, crossover en mutatie uit.

4. Documenteer in je verslag de werking en afstemming van de parameters (populatiegrootte, mutatiekans, crossover-percentage)

🔧 Fase 2 – Heuristieken & Geavanceerde operators

1. Breid je fitness-functie uit met geautomatiseerde heuristieken (lengte, akkoordprogressies, maatvoering).
2. Experimenteer met verschillende recombinatie- en mutatiestrategieën (bijv. pitch-mutatie vs. ritme-mutatie).
3. Analyseer de invloed op zowel de kwaliteit van de composities als de populatiediversiteit.

📌 Randvoorwaarden

- Gebruik geen kant en klare AI/ML – frameworks, standaard programmeerfunctionaliteit volstaat.
- Audio-bibliotheken mogen voor afspelen of schrijven van WAV/MP3-bestanden.
- Je moet je code uitleggen in eigen woorden.

🔡 Inputdata

In de meegeleverde map **music.zip** vind je alles wat je nodig hebt om je gegenereerde notenreeksen om te zetten in audio en een voorbeeld te beluisteren:

- bach.py
  Een voorbeeldscript waarin een tuple bach met twee tracks van (noot, duur)-paren staat. Het toont hoe je via onderstaande code tijdelijke WAV-bestanden maakt en mixt tot één song.wav.

```
from muser import Muser
muser = Muser()
muser.generate(bach)
```

- muser.py
  Bevat de Muser – klasse met de methode generate(song), die per track track_XXX.wav – bestanden genereert en deze mixt tot song.wav.
- song.wav
  Een kant-en-klaar voorbeeld van de output van bach.py. Luister hiernaar om een referentie te hebben voor de klank, timing en mix van je eigen GA – composities.

Als je de Python wave file generator uit music.zip wilt gebruiken vanuit een andere taal, is 't het eenvoudigst om de broncode van muziekstukken zoals bach.py in z'n geheel vanuit die andere taal te genereren. JSON over sockets mag ook.

Nodig is ook de Python library tomita, te installeren met:

```
python -m pip install tomita
```

Op sommige systemen in plaats van python: python3, python3.9, python39 of py39.

Eindproduct

- Je demonstreert en levert een werkend programma dat jouw GA uitvoert en meerdere composities genereert in.
- Je schrijft een technisch artikel (richtlijn: 2 tot 4 pagina's), met de structuur zoals voorgesteld in bijlage G: Structuur van een technisch artikel.