**Exercícios:**

**1: Problema Clássico - OneMax**

**Enunciado**: No problema OneMax, o objetivo é maximizar o número de "1s" em uma string binária. Ou seja, você vai trabalhar com uma população de indivíduos, onde cada indivíduo é representado por uma sequência de bits (0s e 1s). O objetivo do Algoritmo Genético é encontrar a melhor solução, que seria uma string contendo apenas "1s".

Esse problema é uma excelente forma de entender como os Algoritmos Genéticos funcionam, já que ele demonstra como os principais operadores — **seleção**, **cruzamento** e **mutação** — podem guiar a evolução de uma população para atingir um objetivo.

**Como Funciona:**

1. **População Inicial**: Começamos com uma população de soluções aleatórias, cada uma composta por uma sequência binária (ex: [0, 1, 0, 1, 1, 0]).
2. **Avaliação (Fitness)**: Cada solução é avaliada pelo número de "1s" que possui. Quanto mais "1s", melhor a aptidão (fitness) da solução.
3. **Seleção**: Soluções melhores (com mais "1s") têm maior chance de serem escolhidas como "pais" para gerar a próxima geração. Essa seleção é baseada na aptidão de cada indivíduo.
4. **Cruzamento (Recombinação)**: Os pais selecionados se combinam para gerar filhos. Isso é feito trocando partes das suas sequências binárias em um ponto aleatório.
5. **Mutação**: Algumas sequências sofrem mutação, ou seja, alguns bits podem ser invertidos aleatoriamente (de 0 para 1 ou de 1 para 0). Isso ajuda a manter a diversidade e evita que o algoritmo fique preso em uma solução ruim.
6. **Nova Geração**: A população de filhos substitui a população anterior, e o processo é repetido por várias gerações, até encontrarmos uma solução ideal ou o número máximo de gerações ser atingido.

**Tarefa:**

Implemente o Algoritmo Genético no Google Colab para resolver o problema OneMax. O algoritmo deve gerar uma população inicial, aplicar os operadores de seleção, cruzamento e mutação, e evoluir a população até que o número máximo de gerações seja atingido. (Mínimo de 100 gerações.)

**Objetivo**: Encontrar e printar a solução com o maior número de "1s" dentro de uma sequência binária de tamanho definido.

**2: Problema Prático - Otimização de Rotas de Entrega**

**Objetivo do problema:** Maximizar a eficiência da rota de entregas, minimizando a distância total percorrida por um entregador ao seguir uma série de destinos.

**Descrição:** Este problema simula um cenário prático de otimização de rotas de entrega, comum em empresas de logística. A tarefa é encontrar a sequência ideal de entregas para minimizar a distância total percorrida, utilizando um Algoritmo Genético.

**Definições importantes:**

* **Pontos de entrega:** Cada entrega tem um ponto de origem e destino, ambos representados como coordenadas (x, y).
* **Função de custo:** A distância total percorrida pelo entregador, que precisa ser minimizada.
* **Solução candidata:** Cada indivíduo da população representa uma rota de entrega, uma sequência de pontos a serem visitados.

**Objetivo do Algoritmo Genético:** O objetivo do Algoritmo Genético é encontrar a melhor sequência de entregas (rota), minimizando a distância total percorrida entre os pontos de entrega.

**Algoritmo Genético aplicado:**

1. **População inicial:** Representa diferentes sequências de entregas (rotas) geradas aleatoriamente.
2. **Fitness:** Avaliar o quão boa é cada rota calculando a distância total percorrida. A menor distância, melhor a rota.
3. **Seleção:** Selecionar rotas (pais) que têm maior chance de gerar descendentes melhores (com base na distância percorrida).
4. **Cruzamento:** Combinar duas rotas (pais) para gerar novas rotas (filhos).
5. **Mutação:** Alterar ligeiramente uma rota para gerar diversidade nas soluções.
6. **Iteração:** Executar o algoritmo por 100 gerações e monitorar a melhor solução em cada geração.

**Solução esperada:** Ao final das 100 gerações, o Algoritmo Genético retorna a melhor rota de entrega encontrada, juntamente com a menor distância total percorrida.

**Funções principais:**

* **Inicializar População:** Gera as rotas iniciais aleatórias.
* **Avaliar Aptidão (fitness):** Calcula a distância total percorrida para cada rota.
* **Selecionar Pais:** Baseado na aptidão (menor distância).
* **Cruzamento:** Gera novas rotas combinando pares de rotas existentes.
* **Mutação:** Introduz pequenas variações em algumas rotas.
* **Condição de Parada:** 100 gerações.