Fitness Calculation = Final Capital - Initial Capital

Population need 4

1. -2, 5, $200
kototuku loss accepted, kototuku lav chai, intial capital theke koto invest korbo
2.
3.
4.

Lab Assignment 2

Part 1 [7 points]

CSE422: Artificial Intelligence [C02]

You are developing an AI-driven cryptocurrency trading bot that optimizes its **trading strategy** using a Genetic Algorithm (GA). The goal is to maximize profit while minimizing risk by evolving key trading parameters such as:

Gene
- **Stop-Loss (%)** – The percentage at which a position is automatically closed to prevent further loss.

Gene
- **Take-Profit (%)** – The percentage at which a position is closed to secure profits.

Gene
- **Trade Size (%)** – The portion of available capital allocated per trade.

Your task is to implement a GA-based approach to find the optimal set of trading parameters that generate the highest profit over a given set of historical price movements.

Chromosome Representation (Encoding):

{"stop_loss": 2, "take_profit": 5, "trade_size": 20},
{"stop_loss": 3, "take_profit": 7, "trade_size": 30},
{"stop_loss": 07, "take_profit": 4, "trade_size": 25},
{"stop_loss": 08, "take_profit": 6, "trade_size": 15}

String Representation of the chromosomes:

020520 *[Explanation: 02 represents 2% stop_loss, 05 represents 5% take_profit and 20 represents 20% of the total capital amount]*

030730

070425

080615

Fitness Calculation:

Initial Capital: $1000

Let's evaluate the chromosome **020520** which stands for:

Chromosome (Trading Strategy):

- Stop-Loss: 2%
- Take-Profit: 5%
- Trade Size: 20% of capital per trade

on the following historical price changes in percentage:

[-1.2, 3.4, -0.8, 2.1, -2.5, 1.7, -0.3, 5.8, -1.1, 3.5]

Day	Price Change (%)	Trade Size ($)	Exit Condition	Profit/Loss ($)	Updated Capital ($)
1	-1.2	200.00	No SL/TP hit	-2.40	997.60
2	3.4	199.52	No SL/TP hit	+6.78	1004.38
3	-0.8	200.88	No SL/TP hit	-1.61	1002.77
4	2.1	200.55	No SL/TP hit	+4.21	1006.98
5	-2.5	201.39	**Stop-Loss hit**	**-4.03**	1002.95
6	1.7	200.59	No SL/TP hit	+3.41	1006.36
7	-0.3	201.27	No SL/TP hit	-0.60	1005.76
8	5.8	201.15	**Take-Profit hit**	**+10.06**	1015.82
9	-1.1	203.164	No SL/TP hit	-2.23	1013.59
10	3.5	202.72	No SL/TP hit	+7.10	1020.69

use random library within given range

[Note: We start our trading on Day-01 with $1000 capital and invest $200 since the trade size is 20%. Then after incurring a loss of -$2.40, our capital reduces to

*997.60 ($1000-$2.40). Then for the 2nd day we again start our trade with 20% of the updated capital which is 997.60*20% = $199.52*
*Then on Day-5 we see the price change(%) is -2.5 which is greater than the stop_loss threshold, so we cap the loss at 2% instead of 2.5%, thus the loss for Day-05 is 201.39 * 0.02 = 4.03*
*On Day -8, we see the price change(%) is +5.8% which is greater than the take-profit threshold, so we cap the profit at 5% instead of 5.8%, thus the profit for Day-08 is 201.15*0.05 = 10.06]*

Final Fitness Score

The final capital after all trades: $1020.69

Fitness Score = (Final Capital - Initial Capital)
Fitness = $1020.69 - $1000 = **20.69**
This chromosome has a fitness score of 20.69, which represents its profitability.

Summary

- The **fitness function** is the **total profit** after simulated trading.
- A **chromosome** represents a trading strategy with stop-loss, take-profit, and trade size.
- Each chromosome is **evaluated** by simulating trading over historical price data.
- The **fitness score** is the **profit** gained from the starting capital.
- **Higher fitness values indicate better trading strategies**.

Task Breakdown:

Step 1: Define the Chromosome Structure

Part1- Single Point Crossover
Random Selection

Part2- 2 Point crossover
Random Selection

Each **chromosome** represents a trading strategy consisting of three key parameters:

Gene	Description	Range
Stop-Loss (%)	Maximum loss before auto-closing a trade	01% - 99%
Take-Profit (%)	Profit threshold before closing a trade	01% - 99%
Trade Size (%)	Percentage of capital allocated per trade	01% - 99%

Step 2: Initialize Population

Generate an **initial population** of 4 random chromosomes.

Each chromosome is created with **random values** within the defined ranges.

The population size should be the same for every generation.

Step 3: Evaluate Fitness (Profit Calculation)

The **fitness function** determines how profitable a strategy is over historical price movements.

Fitness Function Calculation:

For each chromosome:

1. Start with **$1000 initial capital**.
2. Simulate trades using **historical price movements**.
3. Apply **stop-loss** and **take-profit** rules.
4. Calculate **final capital** and return **profit as fitness score**.

Step 4: Select Parents (Random Selection)

Pick two random individual chromosome to produce two offspring

Step 5: Crossover (Recombine Parent Genes)

We **combine genes from two parents** to create an offspring.

- Use **single-point crossover**:
  - Pick a **random split point**.
  - Mix genes from both parents.

Step 6: Mutation (Introduce Random Changes)

Mutation ensures **genetic diversity** and prevents the algorithm from **getting stuck** in local optima.

- **Small random changes** to genes with a low probability (e.g., **5% mutation rate**).

Step 7: Generate New Population (Next Generation)

- **Select the best individuals** (elitism).
- **Crossover + Mutation** to create new individuals.
- Repeat until reaching a **termination condition** (e.g., **max generations**=10).

<u>Input</u>

"Capital to Start With": $1000

"**historical_prices**": [-1.2, 3.4, -0.8, 2.1, -2.5, 1.7, -0.3, 5.8, -1.1, 3.5],

"**initial_population**": [

{"stop_loss": 2, "take_profit": 5, "trade_size": 20}, {"stop_loss": 3, "take_profit": 7, "trade_size": 30},

{"stop_loss": 1.5, "take_profit": 4, "trade_size": 25},

{"stop_loss": 2.5, "take_profit": 6, "trade_size": 15}],

"**generations**": 10

<u>Output</u>

"**best_strategy**":

{ "stop_loss": 2.3, "take_profit": 6.2, "trade_size": 22 },
"**Final_profit**" : 12.5

Part 2 [3 points]

For this part randomly select two parents from the initial population of your problem statement. Then perform a ***two-point crossover*** to generate two children. The two points have to be chosen **randomly,** but it has to be made sure the second point always comes after the first point.

Here is an example of how ***two-point crossover*** works:

Parent 1: *000111000*
Parent 2: *111000111*

For two points crossover, we have randomly chosen the following points:

1st point:- between index 2 and index 3
2nd point:- between index 6 and index 7

So the two resultant offsprings are, *000000100* & *111111011*

[In this part, you just need to iterate once and print the resultant offspring after doing the crossover]

Part 3 [0 points]

In part 1, you selected parents through random sampling from the initial population. Another advanced technique for parent selection is known as ***Tournament Selection.*** Please take some time to research and understand this method at home. Might be helpful in the near future!