CASE 3: INDEX TRACKING

1. Introduction

Consider a **cap-weighted** index comprised of 30 stocks (UC30). You are the manager of a fund aimed at mimicking the returns of the index. However, due to the constantly changing regulatory environment, you are never allowed to trade or hold all 30 of the stocks at any one time.

Your goal is to have your fund track the returns of the index as closely as possible, while trading only the legally permissible stocks and subject to transaction costs.

2. Data

The case will run for 1,000 ticks (days) over 3 rounds. At the start of each round, and at every tick thereafter, participants will be provided with the prices of all 30 stocks as well as the value of the index. Participants' algorithms will take this data and return their desired portfolio weights. The platform will adjust their portfolio weights and calculate transaction costs accordingly.

Occasionally (approximately once every 50 ticks), announcements will be made regarding the tradability of non-tradability of certain stocks. The new regulations will kick in 20 ticks after they're announced, so participants should adjust their positions accordingly before the new laws apply. Holding positions in stocks that are no longer allowed to be traded will lead to penalties. Note that it will be possible for multiple announcements to occur in a short period (though not multiple in the same tick t).

To assist participants in designing their portfolio-tracking algorithm, teams will be provided with 10,000 ticks of historical data comprising of all 30 stocks prices and the index value for each period. Teams will also be provided with the true weights for the indices. Competition data will be drawn from the same distribution as the historical data.

The underlying true index weights and the distribution of underlying stocks will differ across all 3 rounds. (Think of each case as a completely different index of completely different stocks.) Hence, participants will be provided with three separate sets of historical data, and should tweak their algorithms accordingly for each round.

The index will be calculated as the following:

$$INDEX_t = k \sum_{i=1}^{30} \varphi_i S_{i,t}$$

where $S_{i,t}$ is the price of stock i at time t, and φ_i is the number of outstanding shares of the stock, and k is some constant. While you won't be getting φ_i. Note that φ_i and k are constant for all time t, and the $k\varphi_i$ corresponds to the true index weights mentioned above.

3. Penalty & Transaction Costs

Transaction costs apply when a trade is made in either direction, i.e. whenever the portfolio weight for any stock is modified from period-to-period. These will be incurred per-period. The transaction cost function is defined as the following:

$$TCOST(\Delta_{i,t}) = \exp\left(|\Delta_{i,t}|\right) - 1$$

where $\Delta_{i,t}$ is the change in portfolio weight of stock i at time t. For example, if the portfolio weight for stock A changes from 0.05 to 0.06, $\Delta_{i,t}) = 0.01$. Transaction costs are summed over all periods and all stocks.

Penalties for holding onto a stock that is no longer tradable will be calculated as follows:

$$PENALTY(w_{i,t}) = \begin{cases} 0 \text{ if stock } i \text{ is tradable at time } t \\ exp(|Cw_{i,t}|) \end{cases}$$

where $w_{i,t}$ is the portfolio weight of stock i at time t, and C is a very, very large constant. E.g. $w_{i,t} = 0.05$ means 5% of the portfolio is allocated to stock i.

4. Scoring

Teams will be scored on the following formula (summed over 3 rounds):

$$\text{Score} = -\sum_{t=1}^{1000}(r_{I,t} - r_{P,t})^2 - \text{TOTAL TCOST} - \text{TOTAL PENALTIES}$$

where $r_{I,t}$ is the 1-period log-return of the Index between time t and time $t-1$, and $r_{P,t}$ is the 1-period log-return of your portfolio between time t and time $t-1$. In other words,

$$r_{I,t} = log\frac{\text{INDEX}_t}{\text{INDEX}_{t-1}}$$

$$r_{P,t} = log\frac{\text{PORTFOLIO}_t}{\text{PORTFOLIO}_{t-1}}$$

5. Case Objects & Interface

Teams will have to implement the following methods for this case:

```
double[] InitializePosition(double[] underlyingPrices, double indexValue,
                            double[] trueProductWeights, boolean[] tradables)
```

This method will be invoked at the beginning of each round for participants to initialize their positions (no transaction costs for establishing initial positions).

The argument `underlyingPrices`contains information for the starting prices of the 30 assets at the start of the case and `indexValue` is the value of the index at the start of the case. `trueWeights`is an array of doubles representing the true weights of the portfolio that the index is based on, and sums to 1. `tradables`is an array of booleans, indicating what stocks are tradable at the start. The implementation should return an array of doubles corresponding to the portfolio weights that must sum to 1.

```
public double[] updatePosition(int currentTime, double[] underlyingPrices,
                               double indexValue);
```

This method is called every period and participants will get updates of the prices and index values from the previous day, and return their *desired* positions for the day. Transaction costs are calculated accordingly. Arguments contain information for the prices of assets and index value at time `currentTime`. The implementation should return an array corresponding to the portfolio weights that must sum to 1.

```
public void regulationAnnouncement(int currentTime, int timeTakeEffect,
                                   boolean[] tradableProducts);
```

Participants are informed about what stocks are tradable/untradable through this method, taking effect 20 ticks later. This method is called before `updatePosition`for the same period `currentTime`. The boolean array `tradableProducts`contains information about whether the stock is tradable or not at time `timeTakeEffect`. Note that

$$\texttt{timeTakeEffect} = \texttt{currentTime} + 20$$

always.

```
public void penaltyNotification(int currentTime, boolean[] tradableProducts);
```

This method is called when the system detects that you have incurred a penalty by holding an illegal position (i.e. non-zero weight in a nontradable product). `currentTime`is the time period when the violation occured, and `tradableProducts`has the latest information of what is tradable. Use this method primarily for testing and logging purposes.