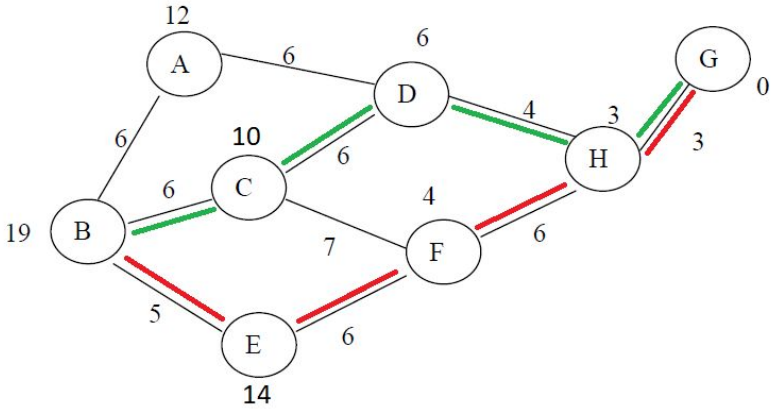
Greedy algorithm in stock market

Eric K. Ribeiro

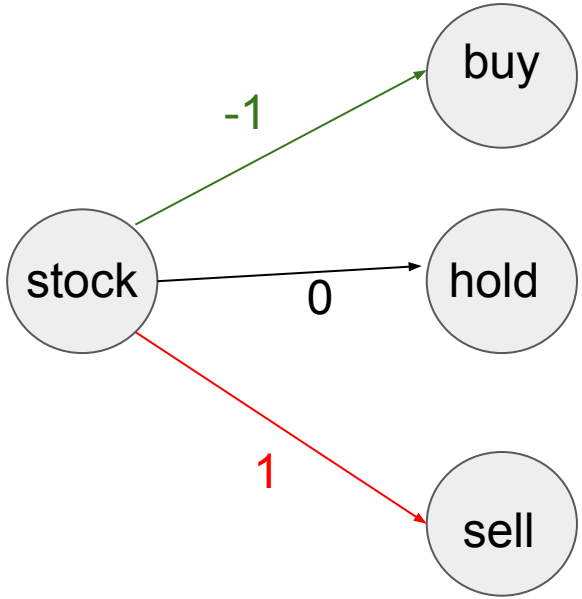
Greedy algorithm:

Figure 1:



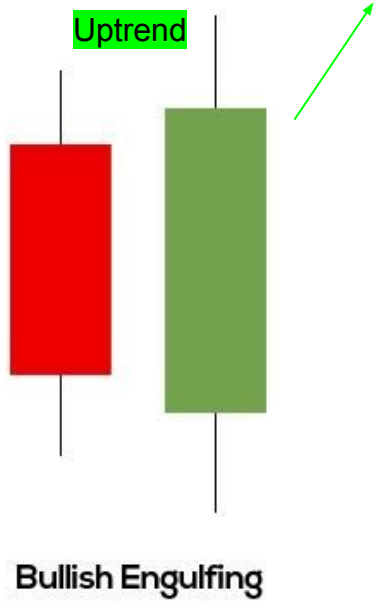
Green: lowest cost path
Red: greedy algorithm path

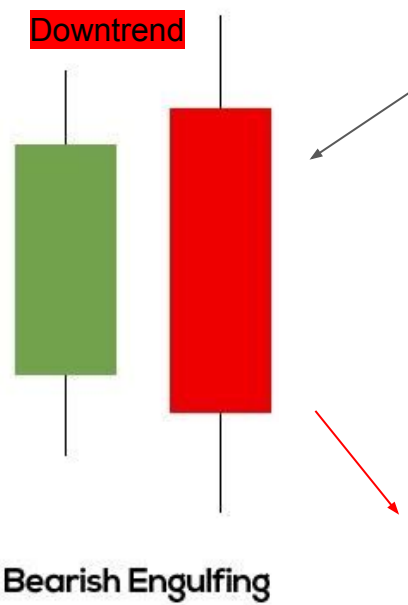
Adaptation of greedy algorithm for stock market:



Has a value of -1. Greedy algorithm will look at this cost and read as a buy signal.

Example of candle pattern:





Has a value of +1. Greedy algorithm will look at this cost and read as a sell signal.

If value < 0 means buy, if value > 0 means sell and if = 0 means hold in our greedy algorithm

```java
public static TradingState greedyTradingWithPatterns(double[][] data, TradingState initialState) {
    TradingState currentState = initialState;

    // Define costs for each pattern
    Map<String, Integer> patternCosts = new HashMap<>();
    patternCosts.put("Bullish Engulfing", -1); // Lower cost is better (buy signal)
    patternCosts.put("Bearish Engulfing", 1);  // Higher cost is worse (sell signal)
    patternCosts.put("Hammer", -1);            // Lower cost is better (buy signal)
    patternCosts.put("Shooting Star", 1);      // Higher cost is worse (sell signal)
    patternCosts.put("Doji", 0);               // Neutral (hold signal)
    patternCosts.put("Morning Star", -1);      // Lower cost is better (buy signal)
    patternCosts.put("None", 0);               // Neutral (hold signal)

    for (int day = 1; day < data.length; day++) {
        double[] currentDayData = data[day];
        double[] previousDayData = data[day - 1];
        String pattern = identifyPattern(previousDayData, currentDayData, day);

        // Decision making based on pattern costs
        int cost = patternCosts.get(pattern);

        if (cost < 0 && currentState.cash >= currentDayData[3]) {
            // Buy if the pattern indicates a buying opportunity
            currentState = new TradingState(day, currentDayData[3], currentState.cash - currentDayData[3], currentState.stocksHeld + 1);
            System.out.println("Day " + day + ": Buy at " + currentDayData[3] + " due to " + pattern);
        } else if (cost > 0 && currentState.stocksHeld > 0) {
            // Sell if the pattern indicates a selling opportunity
            currentState = new TradingState(day, currentDayData[3], currentState.cash + currentDayData[3], currentState.stocksHeld - 1);
            System.out.println("Day " + day + ": Sell at " + currentDayData[3] + " due to " + pattern);
        } else {
            // Hold if the pattern is neutral or if no action can be taken
            currentState = new TradingState(day, currentDayData[3], currentState.cash, currentState.stocksHeld);
            System.out.println("Day " + day + ": Hold at " + currentDayData[3]);
        }
    }
```

Input:

```java
public class TradingGreedyWithPatterns {
    Run | Debug
    public static void main(String[] args) {
        // Example stock price data (Open, High, Low, Close)
        double[][] data = {
            {102.03, 106.32, 101.52, 103.79}, // May 23, 2024
            {104.45, 106.47, 103.00, 106.46}, // May 24, 2024
            {110.24, 114.94, 109.88, 113.89}, // May 28, 2024
            {113.05, 115.49, 110.90, 114.82}, // May 29, 2024
            {114.65, 115.82, 109.66, 110.49}, // May 30, 2024
            {112.52, 112.72, 106.94, 109.62}, // May 31, 2024
            {113.62, 115.00, 112.00, 114.99}, // June 3, 2024
            {115.72, 116.60, 114.04, 116.43}, // June 4, 2024
            {118.37, 122.45, 117.47, 122.43}, // June 5, 2024
            {124.05, 125.59, 118.32, 120.99}  // June 6, 2024
        };
```

Output:

```
PS C:\Users\Nanii>  & 'C:\Users\Nanii\AppData\Local\Programs\Eclipse Adoptium\jdk-
 'TradingGreedyWithPatterns'
Day 1: Hold at 106.46
Day 2: Buy at 113.89 due to Hammer
Day 3: Buy at 114.82 due to Bullish Engulfing
Day 4: Sell at 110.49 due to Shooting Star
Day 5: Sell at 109.62 due to Bearish Engulfing
Day 6: Hold at 114.99
Day 7: Hold at 116.43
Day 8: Buy at 122.43 due to Hammer
Day 9: Hold at 120.99
Final state: Day 9: Price 120.99, Cash 868.97, Stocks 1
Total money: 989.96
```

Initial money: $1000

Conclusion

```
22  public class TradingGreedyWithPatterns {
        Run | Debug
23      public static void main(String[] args) {
24          // Example stock price data (Open, High, Low, Close)
25          double[][] data = {
26              {102.03, 106.32, 101.52, 103.79}, // May 23, 2024
27              {104.45, 106.47, 103.00, 106.46}, // May 24, 2024
28              {110.24, 114.94, 109.88, 113.89}, // May 28, 2024
29              {113.05, 115.49, 110.90, 114.82}, // May 29, 2024
30              {114.65, 115.82, 109.66, 110.49}, // May 30, 2024
31              {112.52, 112.72, 106.94, 109.62}, // May 31, 2024
32              {113.62, 115.00, 112.00, 114.99}, // June 3, 2024
33              {115.72, 116.60, 114.04, 116.43}, // June 4, 2024
34              {118.37, 122.45, 117.47, 122.43}, // June 5, 2024
35              {124.05, 125.59, 118.32, 120.99}  // June 6, 2024
36          };
```

NVIDIA stock market opened in $102.03 and closed in $120.99.
INCREASE IN 18.6%

```
cash:1000, stocksHeld:0);
```

```
Total money: 989.96
```

DECREASE IN 1.004%

How to make the algorithm better?

- Lack of time for the stock to increase in value
- Amount of stocks being traded