Báo cáo học tập - Thứ 2 - 10 / 10 / 2022

Kiểm thử chức năng

Sinh viên : Nguyễn Tiến Hoàng - 21020123

---

1 Mô tả bài toán

Bài toán được em triển khai trong báo cáo lần này mang chủ đề Sử dụng Giải thuật Di truyền (Genetic Algorithms) trong việc Tối ưu hóa dự đoán giá cổ phiếu trong Thị trường chứng khoán .

1.1 Tổng quan

Chủ thể trong bài toán cố gắng dự đoán giá sắp tới của cổ phiếu sao cho gần với giá thực tế nhất , thông qua việc "hỏi những hàng xóm xung quanh" và đưa ra quyết định mua hay bán để tối ưu hóa lợi nhuận nhất có thể .

Ta xem xét giá cổ phiếu ban đầu là P = 1 và tăng hay giảm bao nhiêu chỉ phụ thuộc vào số lượng người mua và người bán , được tính theo công thức của thị trường chứng khoán :

$$bt = \frac{buyers\ agents - sellers\ agents}{Total\ amount\ of\ agents}$$

Formula 2.: bt variable

$$1 + \frac{e^{bt} - e^{-bt}}{e^{bt} + e^{-bt}}$$

Formula 3.: Correction coeficient

Chủ thể chỉ có thể hỏi một vài người , và những người đó đã đi hỏi những người khác và có những câu trả lời , tạo thành một cộng đồng nhỏ trong quần thể bài toán .

Từ lượng thông tin đã có từ cộng đồng nhỏ , có thể dự đoán được giá của cổ phiếu .

Ví dụ : Quần thể bài toán xem xét với 40 người , 21 người mua và 19 người bán , từ công thức trên ta có thể tính được Giá thực tế của cổ phiếu sẽ tăng lên 1.05

Trong cộng đồng nhỏ của chủ thể có 10 người , 3 người mua và 7 người bán , Giá dự đoán của cổ phiếu sẽ giảm xuống 0.62 , lệch đến 0.43 so với Giá thực tế .

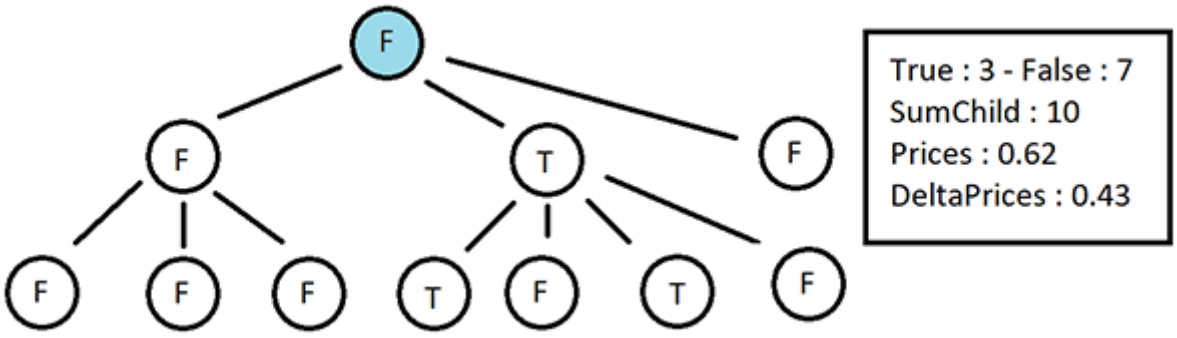
Mục tiêu của bài toán là làm cho độ lệch này nhỏ nhất có thể phụ thuộc vào chủ thể sẽ hỏi những ai .

Độ lệch này thể hiện khả năng đoán đúng và độ uy tín của cộng đồng . Khi chủ thể hỏi những người uy tín , ở trong những cộng đồng chứng khoán uy tín , tỉ lệ thành công sẽ cao hơn rất nhiều so với các trường hợp còn lại .

1.2 Khởi tạo quần thể ban đầu

Quần thể khởi tạo bao gồm 40 người , chia làm 4 cộng đồng nhỏ , mỗi cộng đồng có 10 người , tương ứng với 4 trường hợp chủ thể bài toán ở trong cộng đồng nào , đi hỏi những ai .

Mỗi cộng đồng được biểu diễn theo mô hình cây (Tree) , thể hiện thông tin sau việc đi hỏi và đưa ra quyết định như hình dưới đây .



Quyết định được đưa ra dựa vào số người trả lời là mua (true) lớn hơn hay bán (false) lớn hơn .

1.3 Chọn lọc tự nhiên

Những cộng đồng sau khi khởi tạo , trả về những thông tin như sau :

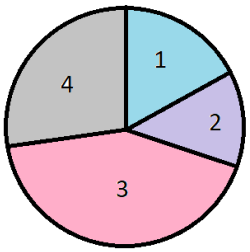
Tree	TrueMinusFalse	SumChild	Prices
0	-4.0	10.0	0.6200510377447751
1	6.0	10.0	1.537049566998035
2	2.0	10.0	1.197375320224904
3	-2.0	10.0	0.802624679775096

DeltaPrices	1/DeltaPrices	Percent
0.4299073372131049	2.326082654189036	15.29745112792527 (%)
0.48709119204015505	2.053003659974951	13.501550814353067 (%)
0.14741694526702398	6.7834806791624125	44.611468977610606 (%)
0.24733369518278403	4.043120769537616	26.589529080111056 (%)

GlobalPrices : 1.04995837495788

Sum : 15.205687762864015

Việc chọn lọc tự nhiên sẽ được thực hiện dựa trên kết quả chênh lệch giá .



Giá chênh lệch càng ít thì cá thể càng tinh túy và cần được giữ lại để lai ghép .

Cá thể tinh túy sẽ chiếm nhiều tỉ lệ % hơn trong vòng quay chọn lọc và đồng nghĩa với xắc xuất được giữ lại là cao hơn .

Ta mong muốn độ chênh lệch giá (DeltaPrices) càng nhỏ , và bằng 0 là tối ưu , nên (1 / DeltaPrices) càng lớn sẽ càng tốt . Cá thể nào có (1 / DeltaPrices) càng lớn thì càng chiếm nhiều tỉ lệ % trong vòng quay .

Sau khi đã hoàn thành Chọn lọc tự nhiên , ta "tái tạo" lại quần thể ban đầu gồm 4 cá thể tinh túy nhất .

1.4 Lai ghép

Tiến hành lai ghép các cặp cá thể (1 và 2) , (3 và 4) để tạo thành 4 cá thể mới mang gen của cả bố và mẹ

Đồng thời hi vọng rằng quần thể con sẽ xuất sắc hơn quần thể trước khi lai tạo .

Mô hình được triển khai bằng cách ngắt cây con trái của cây này gắn vào bên phải của cây kia .

1.5 Nghiệm thu

Cuối cùng là tính lại thông số và tiếp tục lặp lại việc Chọn lọc - Lai ghép cho đến khi chênh lệch không quá 5% so với Giá thực tế hoặc các cộng đồng giống hệt nhau thì dừng lại .

Kết quả thu được là một cộng đồng uy tín , cho ra kết quả Giá dự đoán gần sát nhất so với Giá thực tế .

Nếu chủ thể hỏi đúng những hàng xóm này , họ sẽ có chiến lược và quyết định đúng đắn để đạt lợi nhuận tối ưu nhất .

1.6 Một lưu ý nhỏ

Vì chương trình được triển khai trong báo cáo lần này là chương trình Học Máy (Machine Learning) nên đôi khi kết quả đầu ra không đáp ứng được yêu cầu , chưa tối ưu dẫn đến ca kiểm thử không thành công .

Nhưng khi đó chương trình báo " Chưa tối ưu . Hãy thử lại ! " mà không phát sinh lỗi

Lượng Datasets của chương trình là không lớn (quần thể 40 người trong khi cần 10000 người để train mới hy vọng có hiệu suất tốt) vì thế kết quả thu được sẽ rất thất thường (đôi khi đạt được dưới 2% , đôi khi có thể lên tới 50

2 Kiểm thử chức năng

Hàm main nhận đầu vào là giá trị của 4 cây cộng đồng trong quần thể khởi tạo .

Đầu ra là độ chênh lệch Giá dự đoán so với Giá thực tế .

Kết quả đầu ra với mỗi lần chạy là khác nhau do việc lai tạo có yếu tố ngẫu nhiên , bài toán được thực hiện trên cơ sở cố gắng tối ưu nhất có thể và hi vọng rằng sẽ đạt được kết quả thực nghiệm tốt .

Nếu đầu ra đạt được nhỏ hơn 0.25 (25%) đồng nghĩa với ca kiểm thử thành công .

Nếu ca kiểm thử thất bại , ta cần kiểm thử lại một vài lần nữa .

Thực nghiệm cho thấy xác suất đạt được kết quả tốt rơi vào 60 (%) .

Nên không cần kiểm thử quá 3 lần để ca kiểm thử có thể thành công .

```java
@Test
public void testMain() throws Exception {
  boolean[] tree0 = { false, false, false, false, true, true, false, true, false, false };
  boolean[] tree1 = { true, true, false, true, true, true, true, true, true, false };
  boolean[] tree2 = { true, false, true, false, false, false, true, true, true, true };
  boolean[] tree3 = { true, true, true, true, false, false, false, false, false, false };

  assertTrue(" Chưa tối ưu . Hãy thử lại 1 lần nữa . ", Main.main(tree0, tree1, tree2, tree3) < 0.05);
}
```

2.1 Kiểm thử giá trị biên

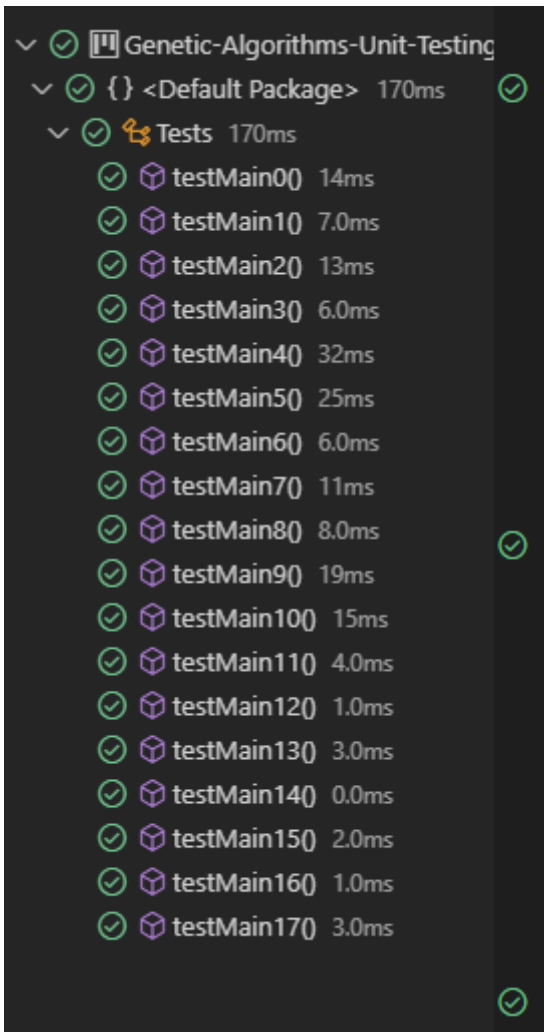
Các giá trị đầu vào : 4 bảng kiểu Boolean , mỗi mảng chứa

[(x) giá trị True và (10 - x) giá trị False (0 <= x <= 10)]

Khi đó ,

min(x) = 0 , max(x) = 10 , min(x)+ = 1 , max(x)- = 9 , nom(x) = 5

Số lượng ca kiểm thử cần thực hiện : (4x4 + 1 = 17 ca)



Kiểm thử giá trị biên mạnh không được sử dụng trong báo cáo lần này vì min- và max+ không tồn tại như các nút trên cây và không có giá trị True / False để gán .

*> Kết quả triển khai :

Xác suất 80% tất cả các ca kiểm thử đều hoàn thành và cho kết quả tối ưu .

```
// nom - nom - nom - nom

@Test
public void testMain3() throws Exception {
  boolean[] tree0 = { true, true, true, true, true, false, false, false, false, false };
  boolean[] tree1 = { true, true, true, true, true, false, false, false, false, false };
  boolean[] tree2 = { true, true, true, true, true, false, false, false, false, false };
  boolean[] tree3 = { true, true, true, true, true, false, false, false, false, false };

  assertTrue(" Chưa tối ưu . Hãy thử lại 1 lần nữa . ", Main.main(tree0, tree1, tree2, tree3) < 0.25);
}

// nom - nom - nom - max-

@Test
public void testMain4() throws Exception {
  boolean[] tree0 = { true, true, true, true, true, false, false, false, false, false };
  boolean[] tree1 = { true, true, true, true, true, false, false, false, false, false };
  boolean[] tree2 = { true, true, true, true, true, false, false, false, false, false };
  boolean[] tree3 = { true, true, true, true, true, true, true, true, true, false };

  assertTrue(" Chưa tối ưu . Hãy thử lại 1 lần nữa . ", Main.main(tree0, tree1, tree2, tree3) < 0.25);
}

// nom - nom - nom - max

@Test
public void testMain5() throws Exception {
  boolean[] tree0 = { true, true, true, true, true, false, false, false, false, false };
  boolean[] tree1 = { true, true, true, true, true, false, false, false, false, false };
  boolean[] tree2 = { true, true, true, true, true, false, false, false, false, false };
  boolean[] tree3 = { true, true, true, true, true, true, true, true, true, true };

  assertTrue(" Chưa tối ưu . Hãy thử lại 1 lần nữa . ", Main.main(tree0, tree1, tree2, tree3) < 0.25);
}

// nom - nom - min - nom

@Test
public void testMain6() throws Exception {
  boolean[] tree0 = { true, true, true, true, true, false, false, false, false, false };
  boolean[] tree1 = { true, true, true, true, true, false, false, false, false, false };
  boolean[] tree2 = { false, false, false, false, false, false, false, false, false, false };
  boolean[] tree3 = { true, true, true, true, true, false, false, false, false, false };
```

2.2 Kiểm thử phân hoạch tương đương

Như đã trình bày ở mục bên trên , bài toán có miền xác định 0 <= x <= 10.

Ta phân chia x làm 3 miền giá trị tương ứng với trạng thái của cây :

- x thuộc [0,4] <=> TrueMinusFalse < 0 <=> Giá cổ phiếu giảm , P(t) < 1

- x thuộc 5 <=> TrueMinusFalse = 0 <=> Giá cổ phiếu không đổi , P(t) = 1

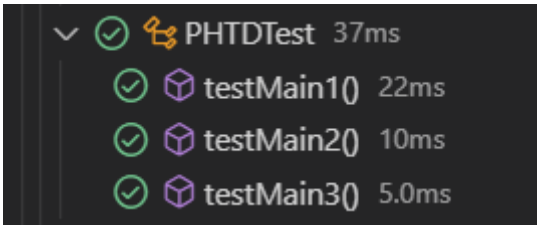
- x thuộc [6,10] <=> TrueMinusFalse > 0 <=> Giá cổ phiếu tăng , P(t) > 1

Nếu kiểm thử phân hoạch tương đương mạnh , ta cần đến 4*4*4 = 64 (ca Kiểm thử)

Vì thế để đơn giản hóa bài toán , ta chỉ kiểm thử phân hoạch tương đương yếu .

Số ca kiểm thử : 3 (ca)

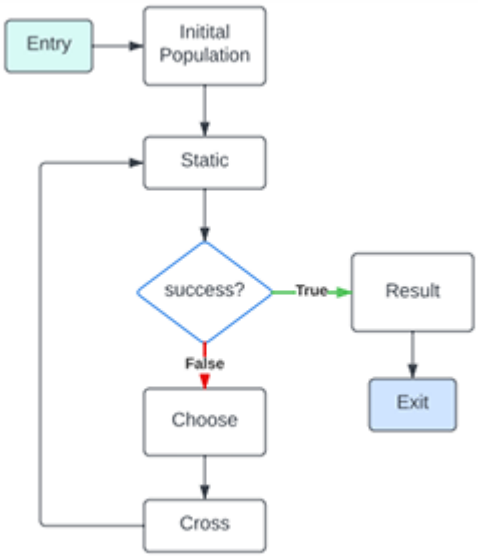
Lựa chọn ca kiểm thử : (3,5,9,10) , (8,2,1,5) , (5,7,5,4)



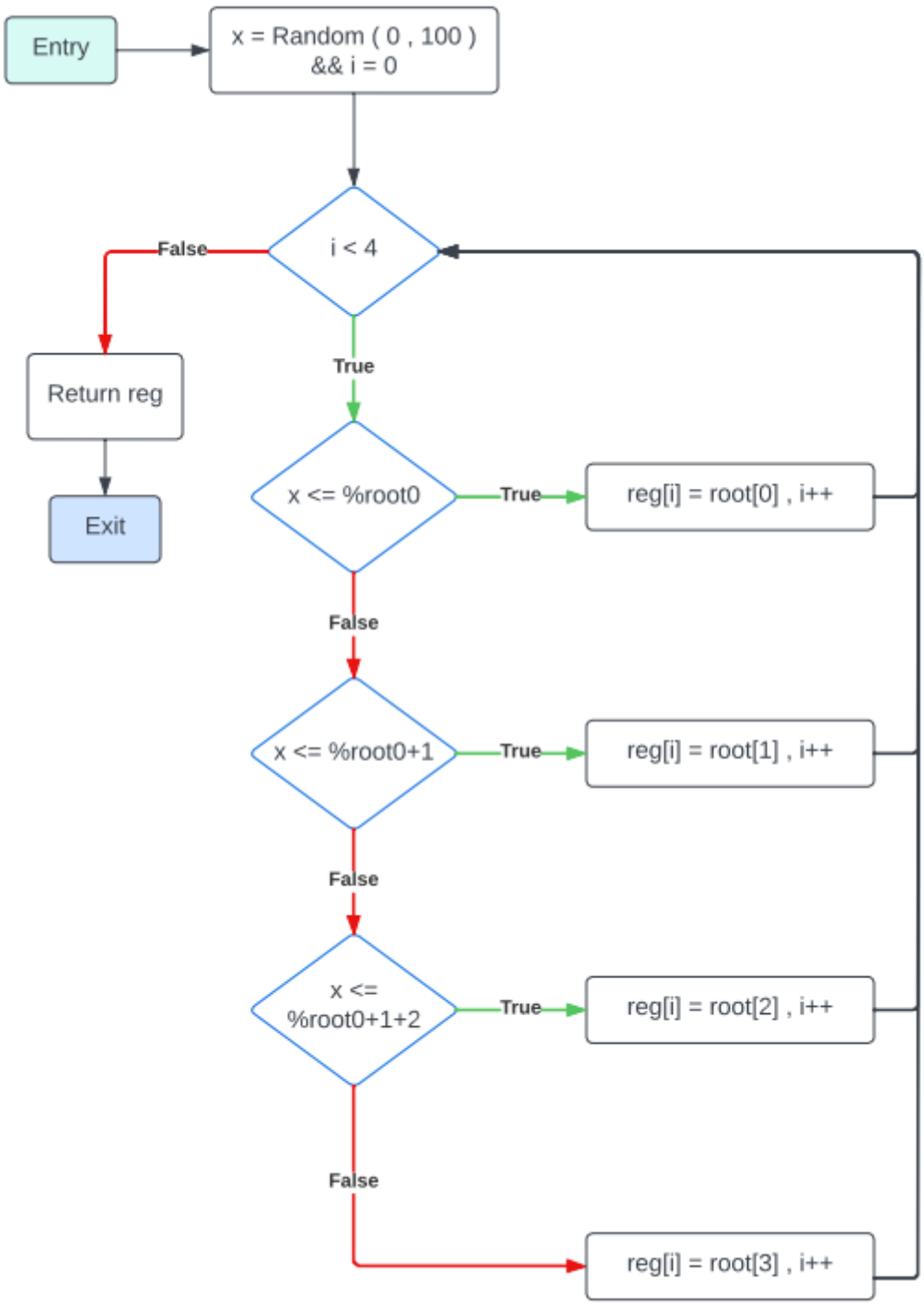
2.3 Kiểm thử dòng điều khiển

Sử dụng độ phủ C2 100% , kiểm tra toàn độ nhánh có thể có của chương trình để sinh ca kiểm thử phù hợp .

2.3.1 Dòng điều khiển của chương trình



Chương trình được hoạt động phần lớn theo tuần tự , ít phân nhánh , chỉ duy nhất hàm choose (random) có sử dụng khối lệnh if else để chọn lọc các cá thể trong quần thể.

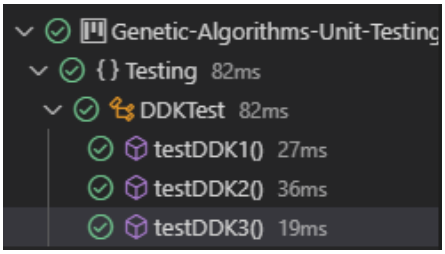


Hàm có tác dụng random 1 số ngẫu nhiên từ 0 - 100 mô phỏng việc quay vòng quay may mắn chọn ra các cá thể phù hợp , cá thể càng tinh túy thì càng chiếm nhiều % , xác suất chọn trúng sẽ cao hơn .

Kết quả trả về là một quần thể mới tinh túy hơn quần thể trước đó .

2.3.2 Triển khai các ca kiểm thử dòng điều khiển

Vì ta không thể quyết định được giá trị của x (ngẫu nhiên) nên ta cố gắng tạo quần thể đầu vào sao cho mỗi cá thể trong quần thể chiếm % gần như nhau (quanh 25%) để xác suất cả 4 luồng đều hoạt động là nhiều nhất .



Ta sinh 4 ca kiểm thử như vừa nêu trên để có thể gần như chắc chắn cả 4 nhánh đều được chạy .

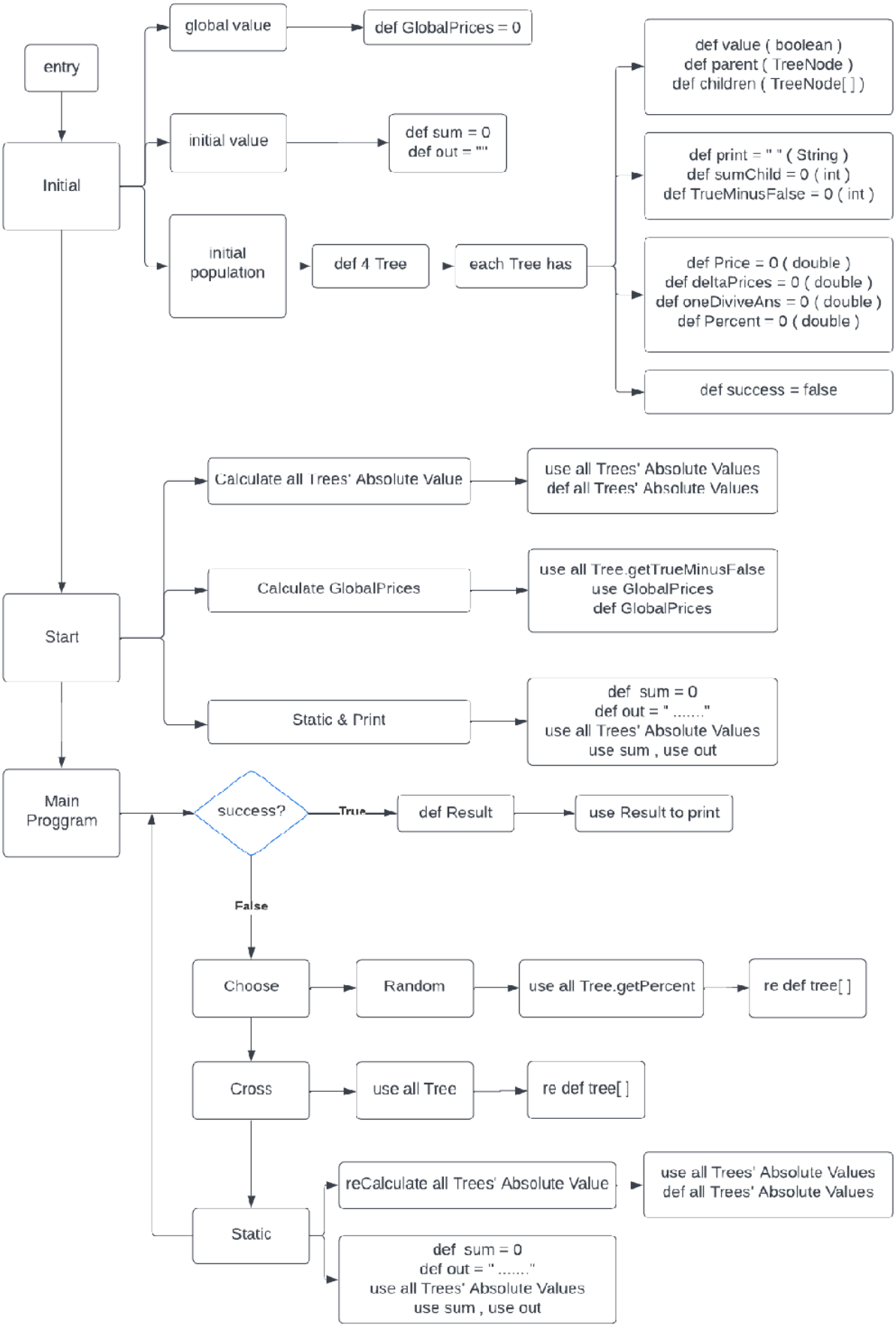
```
// 15.3% - 13.5% - 44.6% - 26.6% ( dải đều xung quanh 25% )
@Test
public void testDDK1() throws Exception {
  boolean[] tree0 = { false, false, false, false, true, true, false, true, false, false };
  boolean[] tree1 = { true, true, false, true, true, true, true, true, true, false };
  boolean[] tree2 = { true, false, true, false, false, false, true, true, true, true };
  boolean[] tree3 = { true, true, true, true, false, false, false, false, false, false };

  assertTrue(" Chưa tối ưu . Hãy thử lại 1 lần nữa . ", Main.main(tree0, tree1, tree2, tree3) < 0.4);
}

@Test
public void testDDK2() throws Exception {
  boolean[] tree0 = { true, false, true, false, false, false, true, true, true, true };
  boolean[] tree1 = { true, true, false, true, true, true, true, true, true, false };
  boolean[] tree2 = { true, true, true, true, false, false, false, false, false, false };
  boolean[] tree3 = { false, false, false, false, true, true, false, true, false, false };

  assertTrue(" Chưa tối ưu . Hãy thử lại 1 lần nữa . ", Main.main(tree0, tree1, tree2, tree3) < 0.4);
}

@Test
public void testDDK3() throws Exception {
  boolean[] tree0 = { false, false, false, false, true, true, false, true, false, false };
  boolean[] tree1 = { true, false, true, false, false, false, true, true, true, true };
  boolean[] tree2 = { true, true, false, true, true, true, true, true, true, false };
  boolean[] tree3 = { true, true, true, true, false, false, false, false, false, false };

  assertTrue(" Chưa tối ưu . Hãy thử lại 1 lần nữa . ", Main.main(tree0, tree1, tree2, tree3) < 0.4);
}
```

Kết quả : Cả 3 nhánh 0 , 1 , 2 , 3 đều được triển khai và cho kết quả tốt ở các ca kiểm thử .

```
Random 0 : 0.5692420995124681 -  Tree : 0
Random 1 : 54.410886727253036 -  Tree : 2
Random 2 : 64.17119292990607 -  Tree : 2
Random 3 : 27.596807518156396 -  Tree : 1
```

```
Random 0 : 38.7553459947427 -  Tree : 2
Random 1 : 44.64158700711194 -  Tree : 2
Random 2 : 97.98969598012212 -  Tree : 3
Random 3 : 5.457357683745245 -  Tree : 0
```

2.4 Kiểm thử dòng dữ liệu



2.4.1 Tổng quan

Lược đồ ở trên biểu diễn khái quát dòng dữ liệu chính của chương trình .

Kiểm thử All-c-uses / Some-p-uses cho phép sinh các ca kiểm thử thỏa mãn đi qua ít nhất 1 def-clear-path từ mọi def đến mọi c-uses , nếu def đó không có c-uses , ta sử dụng p-uses để thay thế .

2.4.2 Danh sách các đường đi sẽ khảo sát

Def	C-uses / P-uses
def GlobalPrices = 0	use it to Calculate (Cal) itself
re def GlobalPrices	use in Cal all Trees' absolute values
def sum = 0 , out = " "	use in Statics
def 4 Trees	use in Crossing
def 4 Trees' absolutes values	use them to Cal themself
re def 4 Trees' absolutes values	use in Static , Print & Choose
def success = false	use in while loop (p - uses)
.....	.....
.....	.....
re def success = true	use to stop while loop (p - uses)
def Result	use Result to Print

Vì chương trình hoạt động gần như toàn bộ là tuần tự nên chỉ cần chương trình có thể dừng và cho ra kết quả , thì toàn bộ def-clear-path được kể trên đều được đi qua .

Vì thế ta cần một ca kiểm thử mà xác suất thành công cao .

```
public class DDLTest {
  // Kiểm thử Dòng dữ liệu

  @Test
  public void testRandom0() throws Exception {
    boolean[] tree0 = { false, false, false, false, true, true, false, true, false, false };
    boolean[] tree1 = { true, true, false, true, true, true, true, true, true, false };
    boolean[] tree2 = { true, false, true, false, false, false, true, true, true, true };
    boolean[] tree3 = { true, true, true, true, false, false, false, false, false, false };

    assertTrue(" Chưa tối ưu . Hãy thử lại 1 lần nữa . ", Main.main(tree0, tree1, tree2, tree3) < 0.4);
  }
}
```

3 Tổng kết

Báo cáo ứng dụng kiểm thử hộp đen (Kiểm thử chức năng) để sinh ca kiểm thử cho chương trình bài toán Sử dụng Giải thuật Di truyền trong việc Tối ưu hóa dự đoán giá cổ phiếu.

Sử dụng 2 kĩ thuật :

Kiểm thử Giá trị biên và Kiểm thử phân hoạch tương đương .

Rất mong báo cáo của em lần này có thể gây ấn tượng với cô và các bạn .

4 Tài liệu

[1] Bài giảng lớp Kiểm thử và Đảm bảo chất lượng phần mềm - Cô Nguyễn Thị Thu Trang https://courses.uet.vnu.edu.vn/course/view.php?id=8435.