Java – Credit Card Debt Part C

Purpose

This lab was designed to teach you how to use computational thinking to solve a real-world problem.

Description

Part C - Using Bisection Search to Make the Program Faster

You'll notice that in Part B, your monthly payment had to be a multiple of $10. Why did we make it that way? You can try running your code locally so that the payment can be any dollar and cent amount (in other words, the monthly payment is a multiple of $0.01). Does your code still work? It should, but you may notice that your code runs more slowly, especially in cases with very large balances and interest rates.

Well then, how can we calculate a more accurate fixed monthly payment than we did in Part B without running into the problem of slow code? We can make this program run faster using a technique introduced in lecture - bisection search! The following variables contain values as described below:

balance - the outstanding balance on the credit card

annualInterestRate - annual interest rate as a decimal

To recap the problem: we are searching for the smallest monthly payment such that we can pay off the entire balance within a year. What is a reasonable lower bound for this payment value? $0 is the obvious answer, but you can do better than that. If there was no interest, the debt can be paid off by monthly payments of one-twelfth of the original balance, so we must pay at least this much every month. One-twelfth of the original balance is a good lower bound.

What is a good upper bound? Imagine that instead of paying monthly, we paid off the entire balance at the end of the year. What we ultimately pay must be greater than what we would've paid in monthly installments, because the interest was compounded on the balance we didn't pay off each month. So a good upper bound for the monthly payment would be one-twelfth of the balance, after having its interest compounded monthly for an entire year.

In short:

Monthly interest rate = (Annual interest rate) / 12.0
Monthly payment lower bound = Balance / 12
Monthly payment upper bound = (Balance x (1 + Monthly interest rate)^12) / 12.0

Created by MITx: 6.00.1x Introduction to Computer Science and Programming

Write a program that uses these bounds and bisection search (for more info check out the Wikipedia page on bisection search) to find the smallest monthly payment to the cent (no more multiples of $10) such that we can pay off the debt within a year. Try it out with large inputs, and notice how fast it is (try the same large inputs in your solution to Part B to compare!). Produce the same return value as you did in Problem 2.

Test Case 1

```
balance = 320000; annualInterestRate = 0.2
```

Output:

```
29157.09
```

Test Case 2

```
balance = 999999; annualInterestRate = 0.18
```

Output:

```
90325.03
```

Test Case 3

```
balance = 44681; annualInterestRate = 0.2
```

Output:

```
4071.15
```

Test Case 4

```
balance = 282651; annualInterestRate = 0.15
```

Output:

```
25196.65
```

Test Case 5

```
balance = 297119; annualInterestRate = 0.15
```

Output:

```
26486.38
```

Test Case 6

```
balance = 289655; annualInterestRate = 0.21
```

Output:

```
26507.0
```

Test Case 7

```
balance = 98461; annualInterestRate = 0.18
```

Output:

```
8893.5
```

Test Case 8

```
balance = 374521; annualInterestRate = 0.22
```

Output:

```
34421.99
```

Test Case 9

```
balance = 191204; annualInterestRate = 0.15
```

Output:

```
17044.7
```

Test Case 10

```
balance = 351742; annualInterestRate = 0.15
```

Output:

```
31355.69
```

Test Case 11

```
balance = 277620; annualInterestRate = 0.18
```

Output:

```
25076.06
```

Test Case 12

```
balance = 497628; annualInterestRate = 0.15
```

Output:

```
44360.55
```