Golden Search

April 13, 2024

1 Runtimes

The runtimes of all three algorithms are factors of one another because they are in the same time class.

In practice, the runtimes of the three algorithms sorted by least amount of evaluations needed is often GSS, binary search, and finally ternary. GSS using stock python dictionary ran relatively close to binary search. For example, with a seed=10 on the range(0,5000) with an x_tol = 0.001 and an epsilon = x_tol * 0.001, the following is the output:

```
Running Ternary
73 evaluations used and 0.7318482398986816 seconds taken

Running Binary
45 evaluations used and 0.44501304626464844 seconds taken

Running Stock GSS
42 evaluations used and 0.48239922523498535 seconds taken
21  hits  62  total:  0.3387096774193548

Running GSS with rounding
33 evaluations used and 0.39127469062805176 seconds taken
30  hits  62  total:  0.4838709677419355
```

Eval and other time was set to be 0.01, and 0.001.

Due to python's dictionary and muliplication, sometimes, values would not be found in the dictionary when it should resulting in GSS running about the same as binary.

In theory, because the recurrence for:

1. GSS is $T(n) = 1 + T(n/(1-\beta)) \rightarrow T(n) = 1.44log_2(n)$
2. Binary Search is $T(n) = 2 + T(n/2) \rightarrow T(n) = 2log_2(n)$
3. Ternary is $T(n) = 2 + T(n/3) \rightarrow T(n) = 2 * 1.709 * log_2(n)$

For the given range and x_tol, our $n = \frac{range}{x\_tol} = \frac{5000}{0.001}$. The numbers add up for binary search with $2log_2(n) = 44.5$ and $3.418log_2(n) = 76.01$

The GSS initally was written without any modifications to python's dictionary's mapping function. The rounding added rounding.

2 Time

Assuming a constant f(x) evaluation time and predictable code behavior, the time needed for f(x) is eval_time=q(x) and the query time for table is other_time=t(x) then we can modify the recurrence to get:

1. GSS is $T(n) = 1.44log_2(n)(q(x) + 2t(x))$ because each iteration does two table checks and one eval
2. Binary Search is $T(n) = 2log_2(n)q(x)$
3. Ternary is $T(n) = 2 * 1.709 * log_2(n)q(x)$

Given the following q(x) = 0.01 and t(x) = 0.001, checking on our input from section 1, we get

1. GSS should be 0.3845 seconds
2. Binary should be 0.445 seconds
3. Ternary should be 0.7606 seconds

which are relatively close to what happened.

3 Golden Section Search/GSS optimized for python

The following lines were added to calculate table hit rates:

```
tableHit = 0
tableTotal = 0

def get_f_eval(x,f_table):
    ...
    tableTotal+=1
    if x in f_table:
        tableHit+=1
    ...
```

The golden section search often(actually) always run slower than expected. Like stated, the look up table only returns true if the exact value is searched. Thus, the hit rate for the look up table is always less than 50%. Thus, it actually runs close to binary search, but can run worse if not hits were made. Thus the following modifications were made.

One fix is discretizing the number by flooring or rounding the input. In the following code, the code allows only tableDigits of decimal places.

```
def get_f_eval_opt(x,f_table):
    global tableHit,tableTotal
    x = math.floor(x * (10**tableDigits))/(10**tableDigits)
    ...
```

This fix helped increase the speed of GSS a lot. For example

```
Running Stock GSS
Returned opt at 3426.28929743541  with value  0.0006673872749161092
42 evaluations used and 0.48255491256713867 seconds taken
21  hits  62  total:  0.3387096774193548
Accuracy: 0.0006673872749161092

Running GSS with rounding
Returned opt at 3426.289931856886  with value  3.296579870948335e-05
33 evaluations used and 0.39141130447387695 seconds taken
30  hits  62  total:  0.4838709677419355
Accuracy: 3.296579870948335e-05
```

Or

```
Running Stock GSS
Returned opt at 928.6201174299365  with value  0.0007519387405636735
45 evaluations used and 0.510460615158081 seconds taken
18  hits  62  total:  0.2903225806451613
Accuracy: 0.0007519387405636735

Running GSS with rounding
Returned opt at 928.6207518514127  with value  0.00011751726435704768
33 evaluations used and 0.3904991149902344 seconds taken
30  hits  62  total:  0.4838709677419355
Accuracy: 0.00011751726435704768
```

We can see that with the rounding implemented, the hit rate approaches basically 50%. We should expect each call of golden section search to only call one get_f_eval and one f(x) or a 50-50 split.

4 Considerations

I think the biggest consideration for this algorithm is how to represent a continuous domain on a discrete domain.