EE4483 CA-Project1 Report

Source Codes

Solution 1:

```python
x=float(input("Please enter the value of x: "))
if x==0:
    print("x^(3/5) is 0")
if x>0:
    if x<1:
        left = x
        right = 1
    else:
        left = 1
        right = x
else:
    if x>-1:
        left = -1
        right = x
    else:
        left = x
        right = -1
x3=x**3
error = 0.000001
while(True):
    mid = (left+right)/2
    mid5=mid**5
    if abs(mid5-x3)<error:
        print("x^(3/5) is "+str(mid))
        break
    if mid5<x3:
        left = mid
    else:
        right = mid
```

Solution 2:

```python
x=float(input("Please enter the value of x: "))
x3=x**3
error = 0.000001
y=x/2
while True:
    y5=y**5
    if abs(y5-x3)<error:
        print("x^(3/5) is " + str(y))
        break
    y=y-(y5-x3)/(5*(y**4))
```

Question a: Please briefly explain the search strategies of the two algorithms.

For solution 1, I used binary search algorithm. Before starting the loop, the value of left and right will be set to define the interval where x^(3/5) falls in. In which iteration of the while loop, we compare the mid point value to the power of 5 with the expected value, which is x cube. The mid point value will be printed out if it matches the expected value. Otherwise we will continue to search in either [left, mid] or [mid, right] according to the result of comparion between mid point value and the expected value.

For solution 2, I use Newton's method. We need to find y such that y^5 - x^3 =0. We can start with y0 = x/2 and apply the recursive formula in each iteration: yn+1 = yn - f(yn)/f'(yn). And y will keep getting closer to the expected value.

Question b: Prove that your algorithms guarantee to find the correct answer.

For solution 1, in each iteration, the size of the interval will become half of the previous one. And the target we are searching for will still be in the new interval. And the while loop will not stop until the target is found. So it guarantee to find the correct anser by the time when the size of interval becomes smaller than 2*deviation allowance.

For solution 2, the rate of convergence of Newton's Method is quadratic for our case. So it will find the answer in the end.

Question c: Compare the two search algorithms and their complexity; explain which one is better and why it is better.

The time complexity of the first solution is O(logN) where N = x/error_allowance. The time complexity of the second solution is approximate to O(logx). And it also depends on the initial value. So time complexity-wise, it is better to use Newton's Method. I have modify the codes to track the number of iterations for each solution as shown below:

Binary search

```
x=float(input("Please enter the value of x: "))
if x==0:
    print("x^(3/5) is 0")
if x>0:
    if x<1:
        left = x
        right = 1
    else:
        left = 1
        right = x
else:
    if x>-1:
        left = -1
        right = x
    else:
        left = x
        right = -1
x3=x**3
error = 0.000001
counter = 0
while(True):
    mid = (left+right)/2
    mid5=mid**5
    counter+=1
    if abs(mid5-x3)<error:
        print("number of iterations: "+str(counter),"x^(3/5) is "+str(mid))
        break
    if mid5<x3:
        left = mid
    else:
        right = mid
```

Newton’s Method

```
x=float(input("Please enter the value of x: "))
x3=x**3
error = 0.000001
y=x/2
counter=0
while True:
    y5=y**5
    counter+=1
    if abs(y5-x3)<error:
        print("number of iterations: "+str(counter),"x^(3/5) is " + str(y))
        break
    y=y-(y5-x3)/(5*(y**4))
```

And below is the output of each solution when the inputs are both 1000:

Binary search

```
number of iterations: 54 x^(3/5) is 63.095734448019314
```

Newton’s Method

```
number of iterations: 15 x^(3/5) is 63.09573444801932
```

So it is obvious that Newton’s Method outperforms the binary search algorithm.