Tian Qiu

00265 25063

CS 314 HW 2

**Problem 1: Warm up problems (15 points)**

1. Assign x = rand (0 : 1)

Then, target value y = 6 \* x

If y <= 1 then result = 1

Else if y <= 2 then result = 2

Else if y <= 3 then result = 3

Else if y <= 4 then result = 4

Else if y <= 5 then result = 5

Else if y <= 6 then result = 6

1. (a)Using round up for 1.5

Since 1.5 = ,

Sign | Exponent | mantissa

0 | 01111111111| 100000000000000000000000000000000000000000000000000000

(b)Using round to nearest for 5.1

Since 5.1 = = 1.010001100b \* 2^2

1025 – 1023 = 2

1. | 100000000001 | 0100011001100110011001100110011001100110011001100111

(c) -5.1 rounds to zero

1. | 100000000001 | 0100011001100110011001100110011001100110011001100110

(d) -5.1 round down

0 | 100000000001 | 0100011001100110011001100110011001100110011001100111

3.

decimal (201) = 11001001b, so the point is shifted to right by 7.

Gap of 201 to next larger double precision number:

2^-52 \* 2^7 = 2^-45

**Problem 2: Floating point mathematics (20 points)**

1.

When c = 0 and a ⊗ b > 0, result is + ∞.

When c = 0 and a ⊗ b < 0, result is - ∞.

When c = 0 and a ⊗ b = 0, result is not a number.

(a ⊗ b) ⊘ c = (a \* b / c) \* (1 + δ1 )( 1 + δ2 ) where δ1 < ε and δ2 <ε

The relative error = abs ((1 + δ1 )( 1 + δ2 ) – 1) = abs(δ1 +δ2 +δ1δ2)

2.

Because 1 / 0 in floating point system will be positive infinity. One divided by positive infinity is zero.

So T = 0, the correct answer will be got by IEEE arithmetic.

Problem 3:



Base case for n=3

For n = 3,

A = F3 =1+1=2,

b = 2,

c = F1 =1

b^2−4ac =2^2−4\*2\*1 = −4

(−1)3 \*4 = −1 \* 4 satisfied.

n = 4,

b =2\*F3 =2\*2 = 4,

a= F4 =2+1=3,

c =F2 =1

b^2−4\*a\*c =4^2−4\*3\*1=4

(−1)4 \*4 = 4 satisfied.

Tackle the general problem by induction.

Assume for n = k the equation is valid.

Then to prove n = k + 1 the equation is still valid.

Proofed.

1. N = 80

When N = 80

julia> include("/Users/tianqiu/Desktop/Purdue/CS314/HW2/3\_2")

0.6180339887498948

0.6180339887498948

1

When N = 79

julia> include("/Users/tianqiu/Desktop/Purdue/CS314/HW2/3\_2")

0.618033988749895

0.6180339887498948

0

# COMPLETE and play around with N

N = 79

fib = zeros(N)

roots1 = zeros(Complex{Float64}, N)

roots2 = zeros(Complex{Float64}, N)

fib[1] = 1; roots1[1] = 1 # avoid reporting these equal

fib[2] = 1; roots1[2] = 1 # because we don't compute them

for i=3:N

fib[i] = fib[i-2] + fib[i-1]

roots1 = (fib[i-1] + 1) / fib[i]

roots2 = (fib[i-1] - 1) / fib[i]

# COMPLETE this section to fill in

end

bign = findfirst(roots1.==roots2) # finds the first zero

println(roots1)

println(roots2)

println(bign)

1. N = 40

function myroots(c)

r = zeros(1:3)

println(c[2]^2 - 4 \* c[1] \* c[3])

r[1] = (- c[2] + sqrt((c[2]^2 - 4 \* c[1] \* c[3])) / (2\*c[1]))

r[2] = (- c[2] - sqrt((c[2]^2 - 4 \* c[1] \* c[3])) / (2\*c[1]))

return r

end

N = 39

fib = zeros(N)

roots1 = zeros(Complex{Float64}, N)

roots2 = zeros(Complex{Float64}, N)

fib[1] = 1;

roots1[1] = 1 # avoid reporting these equal

fib[2] = 1;

roots1[2] = 1 # because we don't compute them

for i=3:N

fib[i] = fib[i-2] + fib[i-1]

r = myroots([fib[i]; -2\*fib[i-1]; fib[i-2]])

roots1[i] = r[1]

roots2[i] = r[2]

end

bign = findfirst(roots1.==roots2) # finds the first zero

Problem 4:

1.

I did this question in MATLAB because the following error:

julia> a = (1782^12 + 1841^12) ^ (1/12)

33.87528458729734

julia> Cshort(a)

ERROR: InexactError()

in call at essentials.jl:56

short: 1.922e+03

long: 1.921999999955867e+03

2.

Absolute Error: 908051659002030145

Relative Error: 0.6602000421790176

Code:

Absulote\_Error = (1782^12+1841^12)­1922^12

Relative\_Error = Absulote\_Error/(1922^12)

3.

1782^12 is an even number

1841^12 is an odd number

So 1782^12 + 1841^12 should be odd. However, 1922^12 is an even number.

So the equation cannot be true.

4.

Both 3987 and 4365 can be divided by 3, so 3987^12 + 43651^2 can be divided by 3.

However, 4472^12 cannot be divided by 3.

So the equation cannot be true.

Problem 5:



Original equation = = 1024

2.

def online\_variance(data):

n=0

mean = 0

M2 = 0

N = length(c)

for I = i:N:

n = n+1

delta = c[i] - mean

mean = mean + delta/n

M2 = M2 + delta\*(c[i] - mean)

end

variance = M2/(n - 1)

std = variance ^ 0.5

3.

N = 10000

input = zeros(Complex{Float64}, 1)

r = zeros(Complex{Float64}, 1)

for i = 1:N

input[1] = 2\*rand()

r[1] += input[1]

end

r = (r/N)\*2^10

when N = 10,000 , , c(4) = 1024;

c(1) = 9.829865986337059e+02;

when N = 40,000 ,

c(2) = 1.013256235292290e+03;

when N = 160,000

c(3) = 1.024406798692284e+03;

stdDiv = sd(c) = 19.480571

Problem 6:

histogram(d,bins=12)

[Plots.jl] Initializing backend: pyplot

WARNING: Couldn't initialize pyplot. (might need to install it?)

ERROR: PyError (:PyImport\_ImportModule) <type 'exceptions.ImportError'>

ImportError('No module named matplotlib.colors',)

[inlined code] from /Users/tianqiu/.julia/v0.4/PyCall/src/exception.jl:81

in pyimport at /Users/tianqiu/.julia/v0.4/PyCall/src/PyCall.jl:386

in \_initialize\_backend at /Users/tianqiu/.julia/v0.4/Plots/src/backends/pyplot.jl:52

in backend at /Users/tianqiu/.julia/v0.4/Plots/src/backends.jl:148

in call at /Users/tianqiu/.julia/v0.4/Plots/src/types.jl:81

in histogram at /Users/tianqiu/.julia/v0.4/Plots/src/Plots.jl:151



n = 10000

N = 23

Get = 0

For I = 1:n

C = get\_birthdays(n);

Get += length(unique(c) < length(c);

End

Win/n

Output = 0.503 which is larger than 0.5

Problem 7:

1. di = [5 9 2 3 2 8 8 3 9 3 8 9 2 3] and C= 4

# nodes is the sum of di from I = 1 to I = 15 times sum of 1/di from 1 to 15

So the result will be 38.9187 which is the same as 311.35/8

1. @show get\_links\_small(0)

function sample\_vertex\_big(k)

a = get\_links\_big

position = 0

for I =1:k

m = a(position)

b = rand(1:length(matrix))

position = matrix(b)

end

return position, length(a(position))

end

function sample\_vertex\_small(k)

a = get\_links\_small

position = 0

for I =1:k

m = a(position)

b = rand(1:length(matrix))

position = matrix(b)

end

return position, length(a(position))

end



nsamp = 125

X = zeros(Int64,nsamp)

d = zeros(Int64,nsamp)

C = Dict{Int64,Int64}() # this is a hash-table or dictionary

ncollisions = 0

for i=1:nsamp

X[i],d[i] = sample\_vertex\_small(25)

@show X[i]

if haskey(C,X[i])

# we have already seen X[i] before, update the collisions

# COMPLETE

else

# then we haven't seen vertex X(i) before, record that

# we've seen it.

C[X[i]] = 0

end end

# estimate the number of nodes

# COMPLETE

node = (sum(d)\*sum(dd)) / (2 \* ncollisions)

nodes = 4357.1765