Sadiq S.

Algorithm for Optimization

Practical No.2

AIM: Implement Fibonacci search.

1. Code for Function Creation:

```
function f(x)
return x*x-x+1
end
```

```
julia> function f(x)
       return x*x-x+1
       end
f (generic function with 1 method)
```

2. Code for Algorithm:

```
function fibonacci_search(f, a, b, n; ϵ=0.01)
s = (1-√5)/(1+√5)
ρ = 1 / (1.618*(1-s^(n+1))/(1-s^n))
d = ρ*b + (1-ρ)*a
yd = f(d)
for i in 1 : n-1
print(a)
print("\n")
print(b)
print("\n")
if i == n-1
c = ϵ*a + (1-ϵ)*d
else
c = ρ*a + (1-ρ)*b
end
yc = f(c)
if yc < yd
b, d, yd = d, c, yc
```

else

a, b = b, c

end

ρ = 1 / (1.618*(1-s^(n-i+1))/(1-s^(n-i)))

end

return a < b ? (a, b) : (b, a)

end

```
julia> function fibonacci_search(f, a, b, n; ϵ=0.01)
       s = (1-√5)/(1+√5)
       ρ = 1 / (1.618*(1-s^(n+1))/(1-s^n))
       d = ρ*b + (1-ρ)*a
       yd = f(d)
       for i in 1 : n-1
       print(a)
       print("\n")
       print(b)
       print("\n")
       if i == n-1
       c = ϵ*a + (1-ϵ)*d
       else
       c = ρ*a + (1-ρ)*b
       end
       yc = f(c)
       if yc < yd
       b, d, yd = d, c, yc
       else
       a, b = b, c
       end
       ρ = 1 / (1.618*(1-s^(n-i+1))/(1-s^(n-i)))
       end
       return a < b ? (a, b) : (b, a)
       end
fibonacci_search (generic function with 1 method)
```

3. Output For Code:

fibonacci_search(f,2,4,5)

```
julia> fibonacci_search(f,2,4,5)
2
4
2
3.2500262583049246
2
2.7499737416950754
2
2.4999947480080555
(2, 2.4999807442774546)
```