Bombora Data Science: *Interview Exam*

Welcome

Welcome! This notebook contains interview exam questions referenced in the *Instructions* section in the `README.md` Please read that first, *before* attempting to answer questions here.

ADVICE
Do not read these questions, and panic, *before* reading the instructions in `README.md`.
WARNING
If using try.jupyter.org do not rely on the server for anything you want to last - your server will be *deleted after 10 minutes of inactivity*. Save often and rember download notebook when you step away (you can always re-upload and start again)!

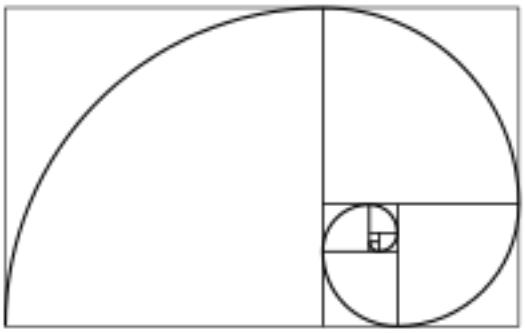
Have fun!

Regardless of outcome, getting to know you is important. Give it your best shot and we'll look forward to following up!

Exam Questions

1. Algo + Data Structures

Q 1.1: Fibionacci



fib image

Q 1.1.1

Given n where $n \in \mathbb{N}$ (i.e., n is an integer and $n > 0$), write a function `fibonacci(n)` that computes the Fibonacci number F_n, where F_n is defined by the recurrence relation:

$$F_n = F_{n-1} + F_{n-2}$$

with initial conditions of:

$$F_1 = 1, F_2 = 1$$

```
f <- function(n){
    if (n==1|n==2)
       return(1)
    else
        return (f(n-1)+f(n-2))
}
```

Q 1.1.2

What's the complexity of your implementation?

The time complexity is $O(F_n)$ and the space complexity is $O(1)$. To explain it in detail, we define the $T(n)$ as the calculation times of number n. So we can get the eqaution:

$T(n) = T(n-1) + T(n-2) + 1$

From the Fibonacci equation, we have:

$F(n) = F(n-1) + F(n-2)$

Adding number one in both sides of the equation $T(n) = T(n-1) + T(n-2) + 1$, we can get:

$[T(n) + 1] = [T(n-1) + 1] + [T(n-2) + 1]$

So it is easy for us to get $F(n) = T(n) + 1$, then the calculation times approximately equals to $T(n) = F(n) - 1$.

Q 1.1.3

Consider an alternative implementation to compute Fibonacci number F_n and write a new function, `fibonacci2(n)`.

```
f <- function(n){
    if (n==1|n==2)
       return(1)
    else{
         f=c(1,1)
        for(i in 3:n){
            f[i] = f[i-1]+f[i-2]
```

```
        }
    }
    return(f[n])
}
```

Q 1.1.4

What's the complexity of your implementation?

From the code above, we can see that the calculation time is n which is less than $T(n) = F(n) - 1$. So this method can reduce the calculation time.

Q 1.1.5

What are some examples of optimizations that could improve computational performance?

We all know that method Floyd arshall algorithm solves all pairs shortest paths. But it takes a long time to do calculation. So we can use method Dijkstra. Dijkstra's algorithm solves the single-source shortest path problem.The calculation time by Dijkstra's algorithm is faster than method Floyd arshall algorithm. Because the time complexity of Dijkstra's algorithm is $O(|V|^2)$. The time complexity of Floyd-Warshall algorithm is $O(|V|^3)$.

2. Prob + Stats

Q 2.5: Uber vs. Lyft

uber vs lyft

You request 2 UberX and 3 Lyfts. If the time that each takes to reach you is IID, what is the probability that all the Lyfts arrive first? What is the probability that all the UberX arrive first?

The probability that all the Lyfts arrive first:

$$\frac{3!\,2!}{5!} = \frac{1}{10}$$

The probability that all the UberX arrive first:

$$\frac{2!\,3!}{5!} = \frac{1}{10}$$