# Q1: DPV .3a

Prove $F_n \geq 2^{.5n}$ for $n \geq 6$.

First I will adjust the initial statement with strong induction:
$P(n) = F_n \geq 2^{.5n}$ and $F_{n+1} \geq 2^{.5(n+1)}$

I will prove P(n) for $n \geq 6$ with induction which in turn proves $F_n \geq 2^{.5n}$ for $n \geq 6$.

First, the base case.
P(6): $F_6 \geq 2^{.5*6}$ and $F_7 \geq 2^{.5*7}$

The sixth Fibonacci number is 8. $2^3 = 8$. The first part of the statement checks out.

The seventh Fibonacci number is 13. $2^{3.5} = 11.31$. The second part of the statement checks out.

Now the induction step.

I will prove $P(n) = F_n \geq 2^{.5n}$ and $F_{n+1} \geq 2^{.5(n+1)} \rightarrow P(n+1) = F_{n+1} \geq 2^{.5(n+1)}$ and $F_{n+2} \geq 2^{.5(n+2)}$

I will assume P(n) to be true and use it to prove P(n+1).

The first statement in P(n+1), $F_{n+1} \geq 2^{.5(n+1)}$ we know is true because it is an assumption made via the induction hypothesis.

The second statement in P(n+1), $F_{n+2} \geq 2^{.5(n+2)}$ can be proven like so.

$F_{n+2} \geq F_{n+1} + F_n$ (this uses the properties of the Fibonacci series)

$F_{n+2} \geq 2F_n$ (since we know the Fibonacci series is increasing, $2F_n$ is less than $F_{n+1} + F_n$)

$2^{.5(n+2)} \geq 2^1 * 2^{.5n}$ (I just subbed in the exponential representations of the numbers)

$2^{.5n} * 2^1 = 2^1 * 2^{.5n}$ (It becomes clear that the two equations equal each other)

That proves that $F_{n+2} \geq 2^{.5(n+2)}$, which proves $P(n) \rightarrow P(n+1)$, which proves P(n) is true for all n greater than 6, which proves the initial statement of $F_n \geq 2^{.5n}$ for $n \geq 6$.

**Q1: DPV .3c**

Following the same setup as before: $F_n \geq 2^{cn}$ and $F_{n+1} \geq 2^{c(n+1)} \rightarrow F_{n+2} \geq 2^{c(n+2)}$

Solve for the highest value of C possible.

$F_{n+2} \geq F_{n+1} + F_n$ (Using the properties of the Fibonacci series)
$2^{c(n+2)} \geq 2^{c(n+1)} + 2^{cn}$ (Subbing in the exponential forms)
$2^{cn} * 2^{2c} \geq 2^{cn} * 2^{c} + 2^{cn}$ (Using the properties of exponents and common bases)
$2^{cn} * 2^{2c} \geq 2^{cn} (2^{c} + 1)$ (Using the distributive property)
$2^{2c} \geq 2^{c} + 1$ (Canceling out terms)
$u^2 \geq u + 1$ (subbing in u for $2^c$)
$u^2 - u - 1 \geq 0$ (rearranging function)
$u = 2^c = \frac{1+\sqrt{5}}{2}$ (quadratic formula)

$$c = \frac{\ln\left(\frac{1+\sqrt{5}}{2}\right)}{\ln(2)} \approx .69424$$

# Q2

I typed up this short Python program to mimic the function bar:

```
def bar(n):
   print("*", end="")
   if n==0:
      return
   else:
      for i in range(0,n):
         bar(i)
      return

if __name__=='__main__':
   for nums in range(0,10):
      print("n=",nums," ",end="")
      bar(nums)
      print("")
```

I counted the number of stars in each row of output to construct this table.

| n | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| # stars | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 |

For here, it's obvious to see that the function T(n) = $2^n$ gives the number of stars printed as a function of n.

One can also write the function, bar(n), out recursively as bar(n) = 1 + bar(0) + bar(1) + bar(2) + … + bar(n-1)

I'll prove that T(n) accurately models that bar function with structural induction – proving that the bar function will double the value of the previous result and that this property is preserved.

The base case bar(0) = 1 star

The constructor rule / property of the function is that bar(n) = 2bar(n-1). Since the lowest value is 1, and each value will be doubled, all successive values will be multiples of 2. This is consistent with the definition of T(n) = 2^n, so the functions are the same.

## Q3:a-f

A. $\frac{n(n+1)}{2000n^2} \rightarrow \frac{n^2+n}{2000n^2}$ $(divide\ both\ by\ n^2) \rightarrow \frac{1}{2000}$ $as\ n\ approaches\ \infty$. The limit is a constant, so the first (top) function is Θ (has the same growth rate) of the bottom function

B. $\frac{100n^2}{.01n^3} \rightarrow (divide\ both\ by\ n^3) \rightarrow \frac{\frac{100}{n}}{.01} \rightarrow 0\ as\ n\ approaches\ \infty$. Thus, $100n^2$ is $O(.01n^3)$

C. $\frac{\log_2 n}{\ln(n)} \rightarrow Both\ logs\ go\ to\ infinity\ as\ n\ increases, so\ use\ L'Hospital's\ rule \rightarrow \frac{\frac{1}{xln(2)}}{\frac{1}{x}} \rightarrow$ $\frac{1}{\ln(2)} \rightarrow Constant$. Thus, the first function is Θ of the bottom function

D. $\frac{(\log_2 n)^2}{(\log_2 n^2)} \rightarrow Both\ go\ to\ infinity\ as\ n\ increases, so\ use\ L'Hospital's\ rule \rightarrow \frac{\frac{2\log_2 x}{\ln(2)x}}{\frac{2}{\ln(2)x}} \rightarrow$ $\log_2 x \rightarrow\ \infty$ The top divided by the bottom goes to infinity as n approaches infinity. Thus, the top function is Ω of the bottom function.

E. $\frac{2^{n-1}}{2^n} \rightarrow \frac{2^n * 2^{-1}}{2^n} \rightarrow 2^{-1} = \frac{1}{2}$ The top divided by the bottom approaches a constant for the first / top function is Θ of the second / bottom function.

F. $\frac{(n-1)!}{n!} \rightarrow \frac{(n-1)*(n-2)*(n-3)\dots}{n*(n-1)*(n-2)*(n-3)\dots} \rightarrow \frac{1}{n} \rightarrow 0$ The first / top function is order O of the second / bottom function.

# Q4

Order of the functions from least to greatest in terms of growth rates:

$n^{\frac{1}{3}}, 5\log(\mathrm{n} + 100)^{10}, \ln^2 n, \quad .001n^4 + 3n^3 + 1, 2^{2n}, 3^n, (n-2)!$