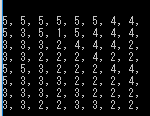
CS324\_Homework4

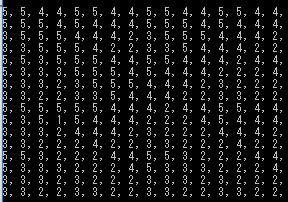
1. Done, 1 will be the randomly placed space,
2. 4x4



8x8



16x16



**3.**  Attached as an image, on the last page. but the general concept was that every k+1 the whole system grows by 4

**4.** Difference problem: running time

I believe that the algorithm for this recursive algorithm would result in looking something like:

t(n) = 4T(n-1) - 1

Where C is some constant

λ = 4

h(x) = a(1) \* 4^x

v(n) = 4v(n-1)-1

k=4k -1

-3k = -1

k=1/3

v(n) = 4(⅓) -1

Xn = a\*4^n + 4/3 -1

X2 = 3 = a\*4^(3) + 4/3 -1

3 = a\*64 +4/3

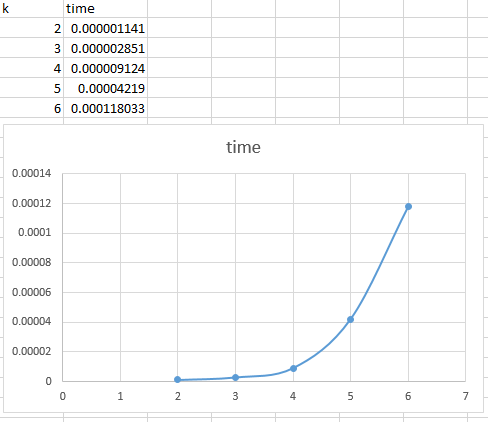
1-⅓ = 64a

⅔ = 64a

2/192 = a

\*\* I don't think I did this quite correctly,

5.



X axis = k value

Y value = time taken to fill board

The run times are able to agree with my run-time analysis from number 4 if we include a constant bigo that will basically represent the processing power my PC used to solve the problem.

Problem 2: Coins.

1. Ok, file is coin.cpp
2. Always a power of 3 makes this interesting, I made a divide and conquer algorithm that would slice everything into thirds, with this it can narrow down to the third sector, then the ninth sector, etc, by applying this strategy.
3. The number of comparisons difference equation for my solution would look something like:

x=6 because we will make 6 total comparisons with our algorithms

t(n) = 2t(n-1)

λ=2

h(n) = a1\*2^n

v(n) = 2v(n-1)

C + 1=2c+1

0=c

v(n) = 2(0)

x(n) = a1\*2^n + 2(0)

6 = a1\*2^3

6 = a1 \* 8

6/8 = a1

¾=a1

\*\* our time equation should be appx the same, so we should just solve for the big o order.

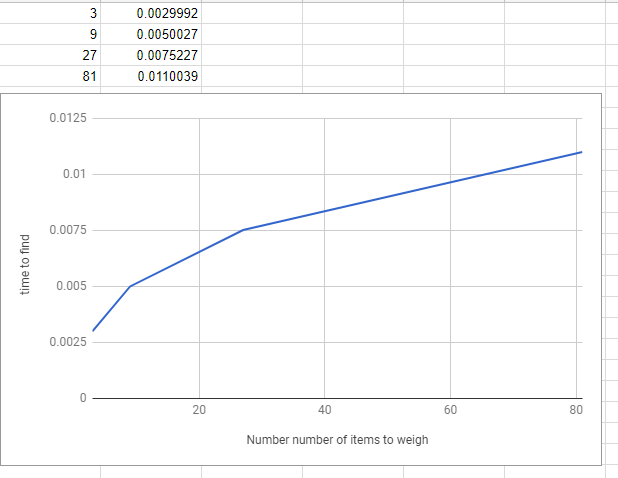
t(n) = 2t(n-1)

This will result in a logarithmic looking graph.

This will be a homogenous equation as, g(n) is =0

So we will have:

4) ok

5) 

6) As long as n is a consistent number, we will always make the same total number of comparisons of weight (because my program will continue to weigh all groups within range to make its check accurately) I would say my equation was correct for the most part, because it seems to scale logarithmically.