1. log(n)(base two) = maximum comparisons for searching in a sorted array.

log(1,000,000,000)(base two) ~ 30

Almost 30 comparisons

1. Moves = 2^n - 1

n = number of disks

= 2^30 - 1

= 1073741823 moves

* 1. 10n = nlogn

10 = log n

10^10 = n

Algorithm B will perform better than A until roughly n = 1000

* 1. 2^n = 10n^2

2 = 10n

⅕ = n

Algorithm A will perform better than Algorithm B until roughly n = 1/5

* 1. Equation = 2^(h+1) - 1

Height of 30 =

2^(30+1) - 1

2^31 - 1 =

2147483647 nodes

* 1. 2^(h+1) - 1 = 1, 000, 000

2^(h+1) = 1, 000, 001

(h+1) log2 = log(1,000,001)

h+ 1 = 19.931

h = 18.931

h = 19

Height is at least 19

* 1. First loop executes 3 times

O(n)

* 1. if loop gets executed n^3 times

If statement is true n times

Triggered n^5

n^5 \* n = n^6

* 1. O(n^2) since in any case, it will be that order and the list is averagely sorted.
  2. O(n)

1. a = 2

binary of 1000 = 1111101000 = 5 + 9

14 multiplications

1. t(n) = 3t(n-1); t(0) = 1

= 3^2t(n-2)

= 3^2t(n-3)

3^k(n-k)

T(n) = 3^(n-1)

Base case: 1 = 3t(0-1)

1 = -3t

t = -⅓

n/2^k = -⅓

n = -2k/3

T(n) = 3^(n-1)\*c

* 1. T(n) = 2T(n/2) + n

a = 2

b = 2

k = 1

2 = 2^1

T(n) ∈ ϴ(n^k log n) since a= b^k

* 1. T(n) = 4T(n/2) + n

a = 4

b = 2

k = 1

4 > 2^1

T(n) ∈ ϴ[n^(log\_b(a))] since a > b^k

* 1. T(n) = T(n/4) + 1

a = 1

b = 4

k = 0

1 = 4^0

T(n) ∈ ϴ(n^k log n) since a= b^k