**Hash Table vs AVL Tree**

**Hash Table Analysis**

**Load Factor**

λ =

where: n = number of items inserted into hash table = 512

m = hash table size = 512

λ = load factor

λ = = 512/512 = 1.0

**Expected Operations Derivation**

Poisson distribution: P ( X = k ) =

Where: x = random variable for chain length

K = specific chain length value

λ = 1

e = Euler

Expected chain length: E [ X ] = \* = λ = 1.0

Search Cost in Chain: For chain of length K

Average position = = = =

Expected Search Cost: E [search] = 1 + = 1 + = 1 + = 1.5 operations

**AVL Tree Analysis**

**Height Bounds**

For AVL tree with n nodes: h =

Where n = 26 ( 26 letters in the alphabet )

Fibonacci relation:

N ( h ) = N ( h – 1) + N ( h – 2) +1

Where N ( h ) is the minimum number of nodes in an AVL tree of height h

N(0) = 1

N(1) = 2

N(2) = N(1) + N(0) + 1 = 2 + 1 + 1 = 4

N(3) = N(2) + N(1) + 1 = 4 + 2 + 1 = 7

N(4) = N(3) + N(2) + 1 = 7 + 4 + 1 = 12

N(5) = N(4) + N(3) + 1 = 12 + 7 + 1 = 20

N(6) = N(5) + N(4) + 1 = 20 + 12 + 1 = 33

**Height determination:**

For n = 26 nodes:

N(5) = 20 < 26 < 33 = N(6)

Therefore: h = 6 is the worst height case

**Total Operations**

Fibonacci bound: h = 6 (worst case)

Logarithmic approximation: h = = 4.754 = 5

Total = 3 trees \* 5 operations each = 15 operations

3 comes from having **3-letter initials** requiring **3 separate tree searches**.

**Performance ratio**

E [ Hash ] = 1 + = 1 + = 1.5 operations

E [ Tree ] = 3 \* = 15 operations

Hash : tree = 1.5 : 15

hash is roughly 10 times faster on average