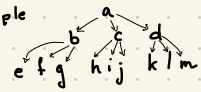
Pablo César Ruíz Hernández A01197044
Carlos Javier García Treviño

1 → Appended at the end of the document

2 → 3ary heap

a) By level, from left to right.

For example



Would be stored as

arr[size] = {a, b, c, d, e, f, g, h, i, j, k, l, m}

b) If the heap is full: $n = 3^0 + 3^1 + 3^2 + 3^3 + \dots + 3^k$, for k levels.

If the heap has only one element in its last level, $n = 3^0 + 3^1 + \dots + 3^{k-1} + 1$

$\Rightarrow (\sum_{i=1}^{k+1} 3^i) + 2 \leq n \leq (\sum_{i=1}^{k} 3^i) + 1$

$\frac{3^k - 3}{2} + 2 \leq n \leq \frac{3^{k+1} - 3}{2} + 1$

$3^k + 1 \leq 2n \leq 3^{k+1} - 1$

$3^k \leq 2n - 1 \leq 3^{k+1}$

It is easy to see that a full heap of k levels has $\frac{1 - 3^k}{3 - 1}$ nodes

$\Rightarrow n = \frac{1 - 3^k}{1 - 3} \Rightarrow -n(1-3) + 1 = 3^k$

$\Rightarrow k = \log_3(1 + n(3-1))$

But since we need to take care of the heap being not full, a general formula for the height given the amount of nodes, would be

$k = \lceil \log_3(1 + 2n) \rceil$

c) Parent node would be at position $\left\lceil \frac{(\text{current} - 1)}{3} \right\rceil$ which will always round to the parent node.

Sibbilings (not cousins) would then be between

$\left\lceil \frac{\text{current} - 1}{3} \right\rceil + 1$, $\left\lceil \frac{\text{current} - 1}{3} \right\rceil + 2$, $\left\lceil \frac{\text{current} - 1}{3} \right\rceil + 3$,

Where current would be one of those.

d) As commented in class, a k-ary heap can be built in linear time.

3 Merging two sorted lists of n elements

A decision tree can be used to represent the divisions of the arrays into sorted arrays and comparing that count with the number of leaves

So, the number of ways to divide 2n numbers into two n size sorted lists is $\binom{2n}{n} = \frac{(2n)!}{(n!)^2}$

$\Rightarrow 2^h \geq \frac{(2n)!}{(n!)^2} \Rightarrow h \geq \log_2\left[\frac{(2n)!}{(n!)^2}\right]$

$\Rightarrow 2^h \geq \log_2(2n!) - 2\log_2[n!]$

$\Rightarrow h = \Theta(2n \log_2(2n)) - 2\Theta(n \log_2(n))$

$h = \Theta(2n)$

So the lower bound is linear

4-

Knuth states that merge insertion is optimal in the number of comparisons for $n = 15$.

Chapter 3 describes a formula to count the number of comparisons, which is simplified as:

$$F(n) = \sum_{1 \leq k \leq n} \lceil \log(\tfrac{3}{4}k) \rceil$$

Which results in the following table of values for F(n)

$n =$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$\lceil \lg n! \rceil =$	0	1	3	5	7	10	13	16	19	22	26	29	33	37	41	45	49
$F(n) =$	0	1	3	5	7	10	13	16	19	22	26	30	34	38	42	46	50

Which indeed has 42 comparisons when $n = 15$

EXAMPLE: