$T(n) = T(n-2q-1) + T(3q/2) + T(q/2) + \Theta(1) \quad T(n) > 2$

$T(n) = 1 \quad$ when $T(n) \leq 2$

Depends on $q \in [0, n/4]$

$T(n-2q-1) = T(n-2q-1-2q-1) + T(3q/2) + T(q/2) + \Theta(1) + \dots$

$= T(n-4q-2) + 2T(3q/2) + 2T(q/2) + \Theta(1)$

$T(n-4q-2) = T(n-8q-4) + 3T(3q/2) + 3T(q/3) + \Theta(1)$

$\Downarrow$

$$\sum_{i=1}^{2n} i \cdot x$$

$O(n)$ runtime

(b) $T(n) = T(n-q-1) + T(n/2-q) + \Theta(n)$

$T(n-q-1) = T(n-2q-2) + T(n/2 - q/2 - 1/2 - q) + \Theta(n)$

$= T(n-2q-2) + T(n/2 + q/2 - 1/2) + \Theta(n)$

$+ T(n/2 - q)$

$T(n-2q-2) = T(n-4q-4) + T(n/2 - q/2 - 1/2) + \Theta(n)$

$+ T(n/2 + q/2 - 1/2) + T(n/2 - q)$

$+ \Theta(n)$

...

$O(n \log n)$

(c.) $T(n) = T(n-q-1) + T(3q) + \Theta(n)$

$\Rightarrow \quad T(n-1) + T(0) + \Theta(n)$

$\Rightarrow \quad T(n-1) + \Theta(n)$

$\Rightarrow \quad O(n)$ worst case

Given a sorted array A, the expected runtime of the silly randomized variant of quicksort is $O(\log n)$.

We can state this as we know that on the first occurrence of the algorithm, we have a $1/n$ of finding the v.

On the second $2/n$

Third $4/n$

and so on and so forth when dealing with randomization.

we can derive $\frac{2^{x-1}}{n}$, thus the average number

of recursive calls is:

$$\frac{1}{n}\sum_{i=1}^{\log(n)} i\,2^{i-1} \Rightarrow i2^{i-1} < \log(n)\cdot 2^{i-1}$$

$$\Rightarrow \frac{1-2^{\log(n)}}{1-2} = 2^{\log n} - 1 = n-1$$

$$\frac{\log(n)}{n} \cdot (n-1) \Rightarrow \log(n)$$

average case run time is $\log(n)$
worst case is $O(n)$ if it has v as the A[0] position and A[length-1] as the random choice for comparison on every iteration, but that's highly unlikely

5. $T(n) = 4T(n/2) + n^2 \lg n$

$a=4 \quad b=2 \Rightarrow n^{\log_2(4)} = n^2$

$f(n)/n^2 \lg n = \lg n$ which shows that $f(n)$ is asymptotically larger, not polynomially. Thus, the master theorem can't be applied. Using the recursion tree method:

$T(n) = 4T(n/2) + n^2 \lg n \ldots$

$T(n/2) = 4T(n/4) + (n/4)\lg(n/4) + n^2 \lg n \ldots$

$T(n/4) = 4T(n/16) + (n/16)\lg(n/16) + (n/4)\lg(n/4) + n^2 \lg n \ldots$

$T(n) \leq n^2 \lg n(\lg n - 1) + n^2 \Rightarrow n^2(\lg n)^2$

$O(n^2(\lg n)^2)$

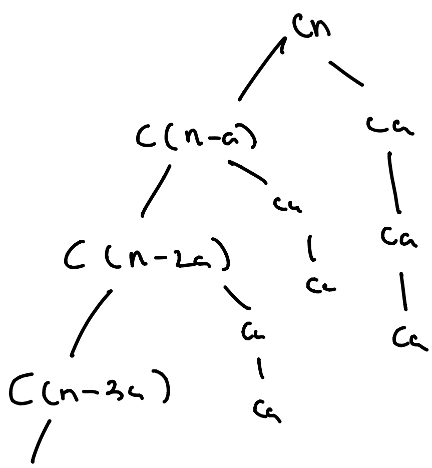
(a) $T(n) = T(n-a) + T(a) + cn$, where $a \geq 1$ and $c > 0$ are constants

$T(n-a) = T(n-2a) + 2T(a) + c(n-a) + cn$

$T(n-2a) = T(n-4a) + 3T(a) + c(n-3a) + c(n-a) + cn$

$T(n-4a) = T(n-8a) + 4T(c) + c(n-7a) + c(n-6a) \cdots \quad c(n)$

. . . .



$T(1) \quad T(n) = (n/a)\,cn = O(n^2)$

b. $T(n) = T(an) + T((1-a)n) + cn$ where $a \in (0,1)$, and $c > 0$ are constants.

an	$(1-a)n$	cn
$\vert$	$\vert$	$\downarrow$
$a(n+1)$	$(1-a)(n+1)$	$c(n+1)$

$O(n)$ constant run time

a. $T(n) = 4T(n/3) + n$

$a = 4 \quad b = 3$

$n^{\log_3(4)}$

$T(n) = \Theta(n^{\log_3 4})$ by case 1

$T(n) \leq 4\left(cn^{\log_3 4}/3\right) + n$

$= \frac{4}{3} cn^{\log_3 4} + n$

this holds as long as $c > 1$ ✓

b. $T(n) = 4T(n/2) + n$

$a = 4 \quad b = 2$

$n^{\log_2(4)} = n^2 \quad f(n) = O(n^{2-1})$ by

case 1 ✓

$\boxed{\Theta(n^2)}$

$T(n) = 4(c \cdot n^2/2) + n$

$= \frac{4}{2}(c \cdot n^2) + n$

$= 2cn^2 + n$

where $c > 1$ ✓

$$T(n) = 3T(\sqrt{n}) + \log n$$

$$x = \log(n)$$

$$T(n) = 3T(\sqrt{n}) + x$$

$$T(2^x) = 3T(2^{x/2}) + x$$

$$A(x) < cx^{\log 3} + dx$$

$$A(x) \leq 3\left(c(x/2)^{\log 3} + d(x/2)\right) + x \quad \text{when} \quad d \geq -2$$

$$\leq cx^{\log 3} + (3/2\, d + 1)x$$

$$\leq cx^{\log 3} + dx$$

$$A(x) \geq cx^{\log 3} + dx \quad \text{when} \quad d \leq -2$$

$$\cdots$$

$$\leq cx^{\log 3} + dx$$

$$\Rightarrow A(x) = \Theta(x^{\log 3}) \Rightarrow T(n) = \Theta(\log^{\log 3} n)$$

Knowing that for any $n > 0$, the number of leaves of an almost complete binary tree is $n/2$

Base case: We know that $h=0$ is true from the previous statement.

Proof by induction: Lets assume the statement is true for $h-1$. Let N be the # of nodes at height h. Looking at a tree T, it will have $n' = n - \lceil n/2 \rceil$ nodes thus

$$N = n'/2^h$$
$$= n/2 \; / 2^h$$
$$= (n/2)/2^h$$
$$= \boxed{\frac{n}{2^{h+1}}}$$