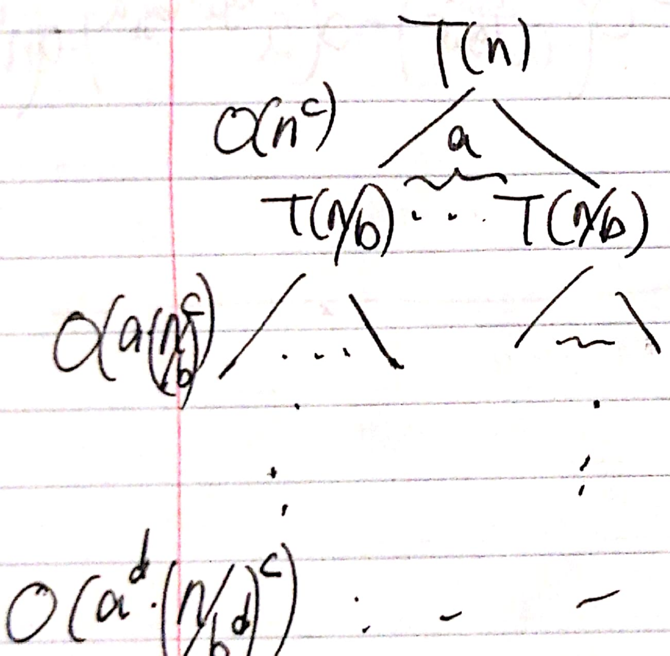
Discussion 2C Week 7 notes Yiyu Chen

1. Complexity analysis. Big-O notation.

`for (int i=1; i≤n; ++i)`	$O(n)$
`for (int i=1; i≤n; i+=t)`	$O(n)$ if t is a constant
`for (int i=1; i≤n; i*=2)`	$O(\log n)$
`for (int i=1; i≤n; n/=2)`	$O(\log n)$.

Recursive Relations.

$$T(n) = a(T(n/b)) + O(n^c)$$



ends when $n/b^d \approx 1$.

$d \approx O(\log n)$

$$T(n) = O\left(n^c + a(n/b)^c + \cdots + a^d (n/b^d)^c\right)$$

$$= O\left(n^c \left(1 + \frac{a}{b^c} + \left(\frac{a}{b^c}\right)^2 + \cdots + \left(\frac{a}{b^c}\right)^d\right)\right).$$

Case 1: $\frac{a}{b^c} < 1$. $T(n) = O(n^c)$

Case 2: $\frac{a}{b^c} = 1$ $T(n) = O(n^c \log_b n)$.

Case 3: $\frac{a}{b^c} > 1$ $T(n) = O\left(n^c \cdot \left(\frac{a}{b^c}\right)^{\log_b n}\right)$.

$$= O\left(n^c \cdot a^{\log_b n} \cdot b^{-c \log_b n}\right)$$

$$= O\left(n^c \left(b^{\log_b n}\right)^{-c} \cdot a^{\log_b n}\right)$$

$$= O\left(\cancel{n^c \cdot n^{-c}} \cdot a^{\log_b n}\right) = O\left(a^{\log_b n}\right)$$

$$= O\left(a^{\frac{\log_a n}{\log_a b}}\right) = O\left(a^{\log_a n \log_b a}\right) = O\left(n^{\log_b a}\right)..$$

Master Theorem

2. Sorting algorithms.

- Merge Sort. $T(n) = 2T(n/2) + O(n)$.

  $O(n\log n)$

- Insertion Sort: $O(n^2)$. Locality helps $O(cn)$

- Bubble Sort: $O(n^2)$. Best $O(n)$

- Selection Sort: $O(n^2)$

- Quick Sort: Average $O(n\log n)$ worst $O(n^2)$.

  (Quick Sort with Insertion Sort $O(n\log n)$)

  binary search $O(\log n)$

Insertion Sort with binary search $O(n^2)$.