CE2101/ CZ2101: Algorithm Design and Analysis

Week 8: Review Lecture

Liu Ziwei

Outline

- Complexity Analysis

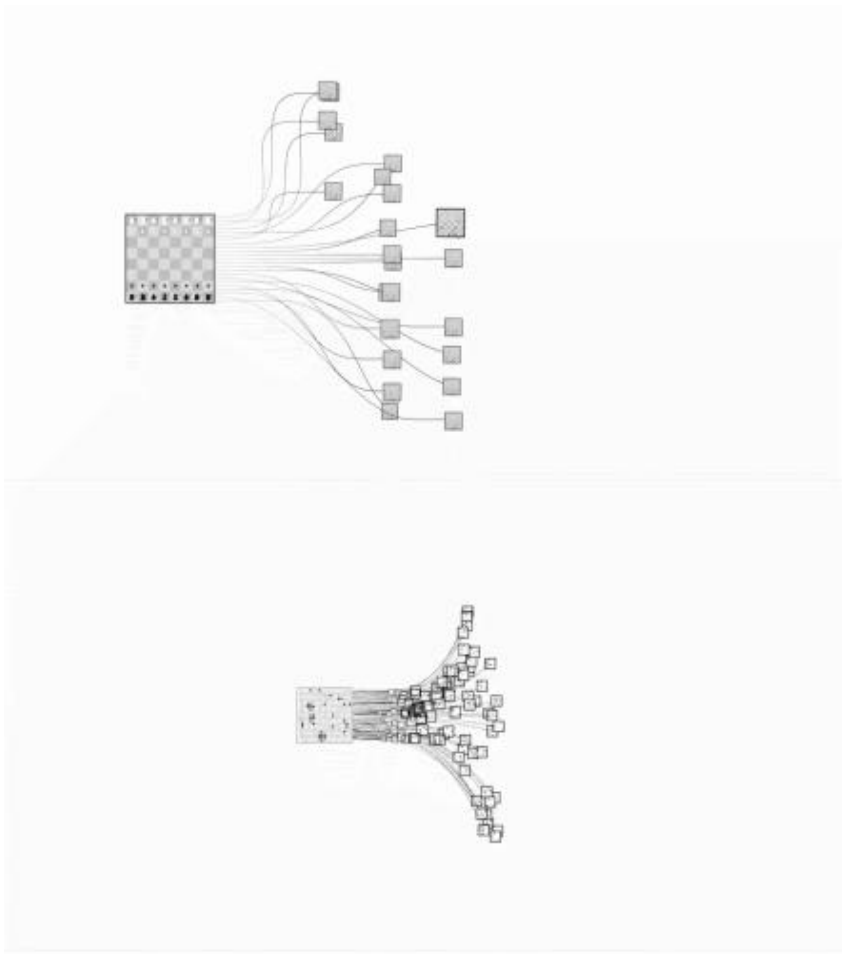
- Solving Recurrences (covered in the exam)
  - Substitution Method
  - Iteration Method
  - Master Method
  - *Linear Homogeneous Recurrence Relation*

- Extended Topics (not covered in the exam)

Please feel free to interrupt me if you have any questions :)

AlphaGo

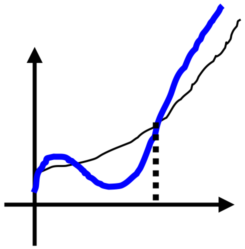
Search Space



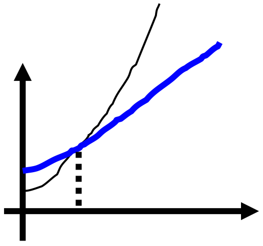
Review of the big oh, big omega, big theta

- The idea of the O, Ω and θ definitions is to establish a relative order among functions.
- We compare the **relative rates of growth**.

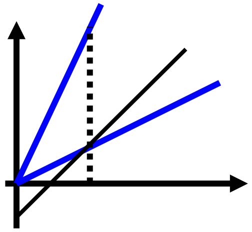
  - If f(n) = O(g(n)), g(n) gives the asymptotic upper bound



  - If f(n) = Ω(g(n)), g(n) gives the asymptotic lower bound



  - If f(n) = θ(g(n)), g(n) gives the asymptotic tight bound



Revision of Complexity Analysis

- Complexity analysis expressed in big-O, big-θ, big-Ω gives the growth rate of a function compared with another function – the order of magnitude of increase in f(n) when n increases.
  E.g $f(n) = O(n^2)$

- Two functions with the same complexity class may have very different running times.

Revision of Complexity Analysis

- Arrange the following functions in increasing order of their big-O time complexity

$2n^2$, logn, nlgn, n!, 2^n, 3^n, lgn, 10n, $100n^{1/2}$, $5n^{2.5}$, $\log(n^2)$, 2^{2n}, 1000, n^n

1000 = O(1)

Logn, lgn,

$\log(n^2)=\theta(\log n)$ $100n^{1/2}$

10n

$\log_{10}(x) = \ln(x) / \ln(10)$

$\ln(x) = \log_{10}(x) / \log_{10}(e)$ That is, $\log_a(x) = c * \log_b(x)$

Revision of Complexity Analysis

$n\lg n$

$2n^2$

$5n^{2.5}$

2^n

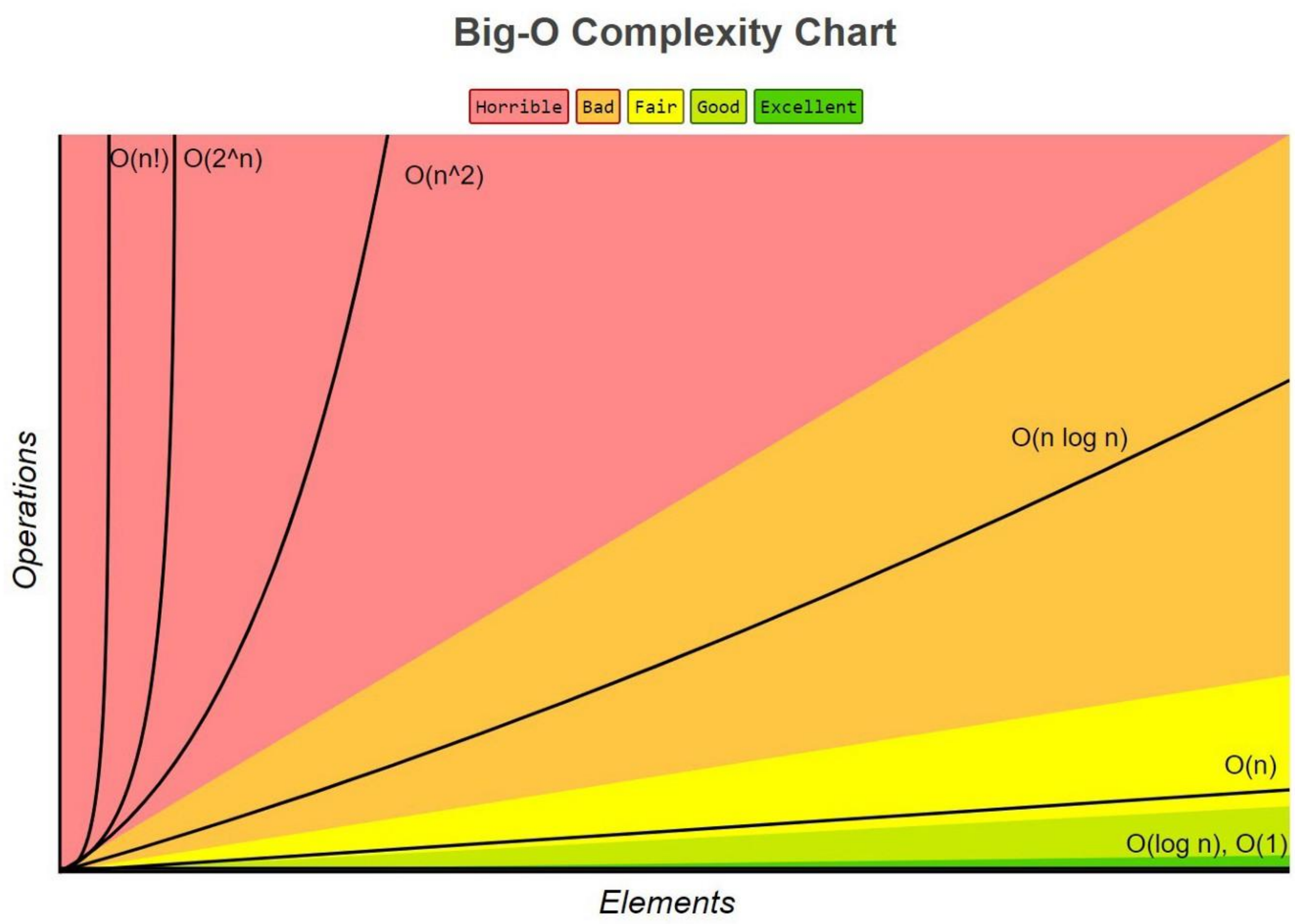
3^n

2^{2n}

$n!$

n^n

Big-O Complexity



Solving Recurrences

1) The substitution method - guess and check

2) The iteration method - expand (iterate) the recurrence

3) The master method – use the manual

1. The substitution method

- It is a "guess and check" strategy. First guess the form of the solution and then use mathematical induction to prove it.
- A powerful method because often it is easier to prove that a certain bound (in the form of the *O* notation) is valid than to compute the bound.

- **Mathematical Induction**: If $p(a)$ is true and, for some integer $k \geq a$, $p(k+1)$ is true whenever $p(k)$ is true, then $p(n)$ is true for all $n \geq a$.
- Example: The worst case for merge sort ($n = 2^k$)

  $W(2) = 1$

  $W(n) = 2\,W(n/2) + n - 1$

Guess $W(n) = O(f(n))$ then prove it.

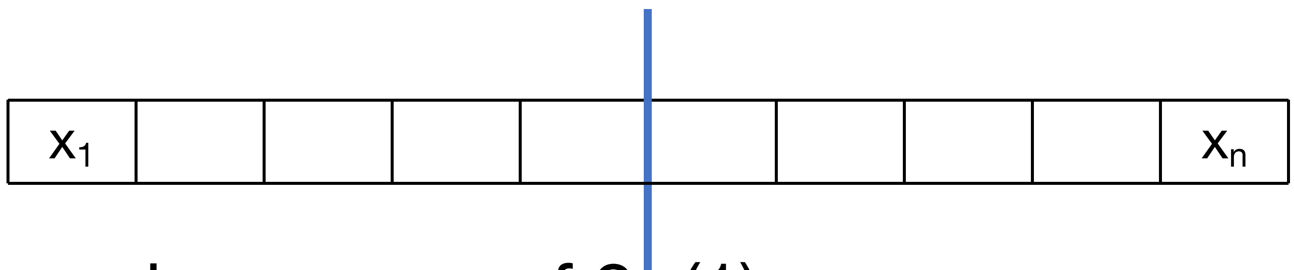
Show (i) $W(2) <= f(2)$ (ii) for some integer $k \geq 2$, assume $W(n) = O(f(n))$ for $n \leq 2^k$, prove $W(2n) <= f(2n)$ then $W(n) = O(f(n))$ for all $n \geq 2$.

Assume k (2^k), Prove k+1 (2^{k+1})

Example of the substitution method

Recurrence for the best case of mergesort: $T(1) = 0$

$T(n) = 2T(n/2) + n/2$ Guess $T(n) = O(n\lg n)$



Proof: consider n is a power of 2. (1)

$T(1) = 0 <= 1*\lg 1$

Example of the substitution method

(2) Assume that $T(2^k) <= k^* 2^k$, prove that $T(2^{k+1}) <= (k+1)^*2^{k+1}$.

$T(2^{k+1}) = 2T(2^k) + 2^k$

$<= 2^* k^* 2^k + 2^k$

$<= k^* 2^{k+1} + 2^k + 2^k$

$= (k+1)^* 2^{k+1}$

Thus $T(n) = O(n\lg n)$

2. The iteration method

- The idea is to expand (iterate) the recurrence and express it as a summation of terms depending only on n and the initial condition.

- Techniques for evaluating summations can then be used to provide bounds on the solution.

- The iteration method usually leads to lots of algebra.

- We should focus on how many times the recurrence needs to be iterated to reach the boundary condition.

Expand, Reach Initial Condition, and Sum

Example of the iteration method

Suppose that, instead of using E[middle] as pivot, QuickSort also can use the median of E[first], E[(first + last)/2] and E[last]. How many key comparisons will QuickSort do in the worst case to sort n elements? (Remember to count the comparisons done in choosing the pivot.)

Before partition: (P is the median)

P									

After partition: (3 comparisons to get the median, n-3 comparisons for partition)

	P								

$T(1) = 0$

$T(2) = 1$

$T(n) = T(n-2) + n$

Example of the iteration method

T(n) = T(n-2) + n

= T(n-4) + n - 2 + n

= T(n-6) + n - 4 + n - 2 + n

= T(n-8) + n - 6 + n - 4 + n - 2 + n

$a_i = a_1 + (i-1)d$

$s_k = \frac{k}{2}(a_1 + a_k)$

If n = 2k (even)

T(n) = T(2) + (4 + 6 + 8 + … + n) // k-1 terms

$\equiv 1 + \frac{k-1}{2}(4 + n)$

$\equiv O(n^2)$

Example of the iteration method

If $n = 2k+1$ (odd)

$T(n) = T(1) + (3 + 5 + 7 + \ldots + n)$ // k terms

$= \frac{k}{2}(3 + n)$

$= O(n^2)$

$$\boxed{\begin{array}{l} a_i = a_1 + (i-1)d \\ s_k = \frac{k}{2}(a_1 + a_k) \end{array}}$$

3. The master method

For **$W(n) = aW(n/b) + f(n)$** $a \geq 1$ and $b > 1$

The manual:

1. If $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$, then $W(n) = \Theta(n^{\log_b a})$.

2. If $f(n) = \Theta(n^{\log_b a})$, then $W(n) = \Theta(n^{\log_b a} \log n)$.

   If $f(n) = \Theta(n^{\log_b a} \log^k n)$, $k \geq 0$,
   then $W(n) = \Theta(n^{\log_b a} \log^{k+1} n)$

3. If $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$, and if $a\, f(n/b) \leq c\, f(n)$ for some constant $c < 1$ and all sufficiently large n, then $W(n) = \Theta(f(n))$.

Compare f(n) and $n^{\log_b a}$, then Choose Condition

Example of the master method

Multiplying two nxn matrices

$W(n) = 7W(n/2) + 15n^2/4$

$n^{\log_b a} = n^{\log_2 7} = n^{\frac{\ln 7}{\ln 2}} = n^{2.8075}$

$$\log_b x = \frac{\log_d x}{\log_d b}$$

$f(n) = 15n^2/4$

$= O(n^{2.8075-0.5})$

$W(n) = \theta(n^{2.8075})$

Thanks!

Q & A