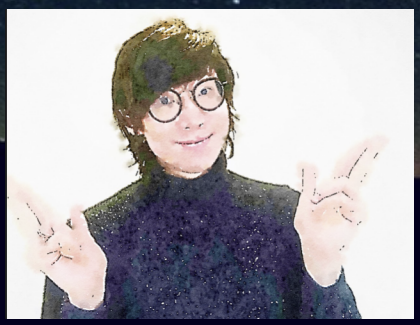
CSC240

Lecture 11

Author: Joseph Siu

Email: joseph.siu@mail.utoronto.ca

Date: March 14, 2024



Midterm EX310 Wednesday Mar. 6.

Friday Mar 8 lecture instead of tutorial.

Offices hours will start ≈ 6pm today.

Asymptotic Notation

Let $\mathcal{F}$ denote the set of all fractions from $\mathbb{N}$ to $\mathbb{R}^+ \cup \{0\}$.
For all $f \in \mathcal{F}$, let

$$O(f) = \{g \in \mathcal{F} \mid \exists c \in \mathbb{R}^+ . \exists b \in \mathbb{N} . \forall n \in \mathbb{N} . [(n \geq b) \text{ IMPLIES } (g(n) \leq cf(n))]\}$$

For all n sufficiently large, $g(n)$ is at most a constant factor times $f(n)$.

$$\Omega(f) = \{g \in \mathcal{F} \mid \exists c \in \mathbb{R}^+ . \exists b \in \mathbb{N} . \forall n \in \mathbb{N} [(n \geq b) \text{ IMPLIES } (g(n) \geq cf(n))]\}$$

For all n sufficiently large, $g(n)$ is at least a constant factor times $f(n)$.

Example 1. $6n + 4 \in O(3n)$ Since $6n + 4 \leq 3 \times 3n$ for all $n \geq 2$.

Remark 1. We use $O(n^2)$ instead of $O(f(n))$ where $f(n) = n^2$ for all $n \in \mathbb{N}$.

$$(g \in \Omega(f)) \text{ IFF } (f \in O(g))$$

$$\Theta(f) = \{g \in \mathcal{F} \mid \exists c \in \mathbb{R}^+ . \exists c' \in \mathbb{R}^+ . \exists b \in \mathbb{N} . \forall n \in \mathbb{N} . (n \geq b) \text{ IMPLIES } (cf(n) \leq g(n) \leq c'f(n))$$

1 Analysis of Algorithms

For a particular algorithm A,
let $t_A(I)$ be the number of steps the algorithm A performs on input I.

Example 2. Linear Search (L, x):

```
1 i <- 1
2 while i <= length(L) do
3     if L[i] = x then return i
4     i <- i+1
5 return 0
```

Steps:

- $+1$
- comparison with x **(we will focus on this one)**
- array access $L(i)$
- return **(this is a bad one)**
- omparison of i and length(L)

on input $L = [2, 4, 6, 8]$,

	# of steps
$x = 2$	1
$x = 4$	2
$x = 8$	4
$x = 1$	4

Want b express running time as a function of input size.

1.1 Worst case time complexity

Let S_n denote the set of all input of size n to the algorithm A.

Then $T_A : \mathbb{N} \to \mathbb{N}$ is the worst case time complexity where $T_A(n) = \max\{t_A(I) \mid I \in S_n\}$.

$T_{\text{linear search}}(n) = n$

1.2 Average case time complexity

$T'_A : \mathbb{N} \to \mathbb{R}^+ \cup \{0\}$ where $T'_A(n) = \mathrm{E}[t_a]$ where the expectation is taken over propability space on S_n.

$$T'_A(n) = \frac{\sum_{I \in S_n} t_A(I)}{|S_n|}$$

if all elements in S_n are equally likely.

2 Wost Case Time Complexity

Let $u \in \mathbb{N} \to \mathbb{N}$

Algorithm A has worst case time complexity at most u means $T_A \leq u$.

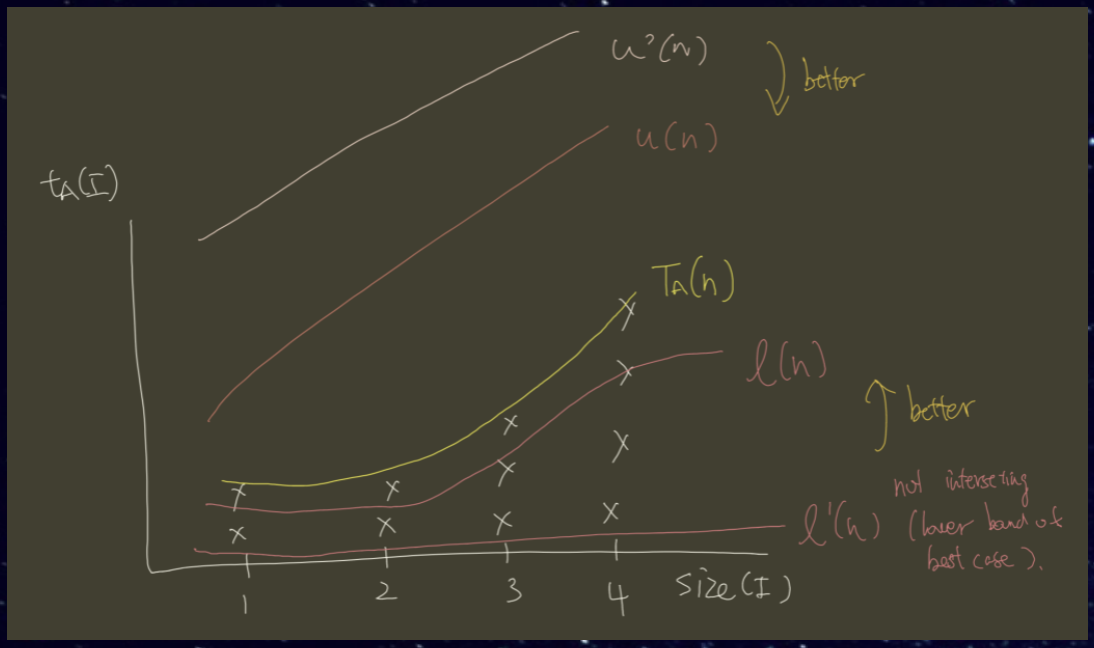
That is,

$$\forall n \in \mathbb{N}. \max\{t_A(I) \mid I \in S_n\} \leq u(n)$$

Or,

$$\forall n \in \mathbb{N}. \forall I \in S_n. [t_A(I) \leq u(n)]$$

To prove $T_A \leq u$, you must show for all $n \in \mathbb{N}$ and **for all inputs** $I \in S_n$, algorithm A on input I takes at most $u(n)$ steps.



$\ell : \mathbb{N} \to \mathbb{N}$

Algorithm A has worst case time complexity at least ℓ means $I_A \geq \ell$

$$\forall n \in \mathbb{N}. \, [\max\{t_A(I) \mid I \in S_n\} \geq \ell(n)]$$

Or,

$$\forall n \in \mathbb{N}. \exists I \in S_n. \, [t_A(I) \geq \ell(n)]$$

Remark 2. This is wrong: In other words, there exists an input I such that $t_A(I) \geq \ell(\text{size}(I))$.
We have to show for all n.

To prove $T_A \geq l$ for all $n \in \mathbb{N}$, you must **find an input** $I \in S_n$ such that A takes at least $\ell(n)$ steps on input I.

$T_A \in O(u)$:

$$\exists c \in \mathbb{R}^+.\exists b \in \mathbb{N}.\forall n \in \mathbb{N}.\,[(n \geq b) \text{ IMPLIES } (T_A(n) \leq c \cdot u(n))]$$

Or,

$$\exists c \in \mathbb{R}^+.\exists b \in \mathbb{N}.\forall n \in \mathbb{N}.\,[(n \geq b) \text{ IMPLIES } \forall I \in S_n.t_A(I) \leq c \cdot u(n)]$$

$T_A \in \Omega(\ell)$:

$$\exists c \in \mathbb{R}^+.\exists b \in \mathbb{N}.\forall n \in \mathbb{N}.(n \geq b) \text{ IMPLIES } (T_A(n) \geq c \cdot \ell(n))$$

Or,

$$\exists c \in \mathbb{R}^+.\exists b \in \mathbb{N}.\forall n \in \mathbb{N}.(n \geq b) \text{ IMPLIES } [\exists I \in S_n.(t_A(I) \geq c \cdot \ell(n))]$$

3 Recursion

```
1 function square(n)
2 if n=1
3 then return n
4 else return 2*n - 1 + square(n-1)
```

input size: n

step: # of recursive calls $\times 4$= # of arithmetic operations.

Let $T_{\text{SQ}}(n) : \mathbb{Z}^+ \to \mathbb{N}$ denote the # of arithmetic operations performed by square(n).

$$T_{\text{SQ}}(n) = \begin{cases} 0 & \text{if } n = 1 \\ 4 + T_{\text{SQ}}(n-1) & \text{if } n > 1 \end{cases}.$$

Definition 1

A recurrence is a recursively defined function. Solving a recurrence means finding a non-recursive description for the function. 𝍐

Methods

0. Look it up online or in a book.
1. Guess and Verify.
   - Generate a table of values
   - Look for a pattern
   - Guess a solution
   - Check solution for small values of n
   - Prove it is correct using induction

n	$T(n)$
1	0
2	4
3	8
4	12

Guess $T(n) = 4(n-1)$

Let $P(n)$ = "$T(n) = 4(n-1)$"

Prove $\forall n \in \mathbb{N}.\ P(n)$.

2. Homogeneous Linear Recurrences using Characteristic Polynomials

$$F(0) = 0$$
$$F(1) = 1$$
$$F(n) = F(n-1) + F(n-2), \text{ for } n \geq 2$$

Constructor Case is a linear combination of a fixed number of preceding terms.

$$f(n) = \sum_{i=1}^{d} a_i f(n-i)$$

guess $f(n) = c \cdot x^n$ where c, x are parameters whose values are chosen letter.

In this case,

$$cx^n = cx^{n-1} + cx^{n-2}$$
$$x^2 = x + 1 \quad \text{This is called the characteristic equation}$$
$$x^2 - x - 1 = 0$$
$$x = \frac{1 \pm \sqrt{5}}{2}$$
$$F(n) = c \cdot \left(\frac{1+\sqrt{5}}{2}\right)^n \text{ or } F(n) = c \cdot \left(\frac{1-\sqrt{5}}{2}\right)^n$$

If $f(n) = y^n$ and $f(n) = x^n$ are two solutions for the self-referential part, then $f(n) = cy^n + c'x^n$ is also a solution.

Since $0 = F(0) = c + c'$ and $1 = F(1) = c\left(\frac{1+\sqrt{5}}{2}\right) + c'\left(\frac{1-\sqrt{5}}{2}\right)$. We can solve for c and c', it turns out that $c = \frac{1}{\sqrt{5}}$ and $c' = -\frac{1}{\sqrt{5}}$.

Hence,

$$F(n) = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^n}{\sqrt{5}} - \frac{\left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}}$$

This can be proved by induction.

Remark 3. $x^d - \sum_{i=1}^{d} a_i x^{d-i} = 0$ is the characteristic equation.

If this equation has d distinct roots $r_1, \ldots, r_d$
$f(n) = \sum_{i=1}^{d} c_i r_i^n$

3. Repeated Substitution and Verify (Plug and Chug)

Example.

For $n \in \mathbb{Z}^+$, let

$$M(n) = \begin{cases} c & \text{if } n = 1 \\ M\left(\lceil \frac{n}{2} \rceil\right) + M\left(\lfloor \frac{n}{2} \rfloor\right) + dn & \text{if } n > 1 \end{cases}.$$

Consider the special case when n is a power of 2:

$$M(n) = \begin{cases} c & \text{if } n = 1 \\ 2M\left(\frac{n}{2}\right) + dn & \text{if } n > 1 \end{cases}.$$

$$\begin{aligned} M(n) &= 2M\left(\frac{n}{2}\right) + dn \\ &= 2\left[2M(\frac{n}{4}) + d\frac{n}{2}\right] + dn \\ &= 4M(\frac{n}{4}) + 2dn \\ &= 4\left[2M(\frac{n}{4}) + d\frac{n}{4}\right] + 2dn \\ &= 8M(\frac{n}{8}) + 3dn \\ &\vdots \\ &= 2^i M(\frac{n}{2^i}) + idn \end{aligned}$$

When $i = \log_2 n$, we get

$$\begin{aligned} M(n) &= n \cdot M(1) + dn \cdot \log_2 n \\ &= cn + dn \cdot \log_2 n \end{aligned}$$

Let $Q(i) =$"$M(2^i) = c2^i + di2^i$"

We prove $\forall i \in \mathbb{N}.Q(i)$ by induction using the thoerem $M(n) \in \Theta(n \log n)$.

Theorem 1 – Master Theorem

Suppose that for $n \in \mathbb{Z}^+$,

$$T(c) = \begin{cases} c & \text{if } n < B \\ a_1 T(\lceil \frac{n}{b} \rceil) + a_2 T(\lfloor \frac{n}{b} \rfloor) + dn^i & \text{if } n \geq B \end{cases},$$

where $a_1, a_2, B, b \in \mathbb{N}$, $a = a_1 + a_2 \geq 1$, $b > 1$, $c, d, i \in \mathbb{R}^+ \cup \{0\}$.
Then,

$$T(n) \in \begin{cases} \Theta(n^i \log n) & \text{if } a = b^i \\ \Theta(n^i) & \text{if } a < b^i \\ \Theta(n^{\log_b a}) & \text{if } a > b^i \end{cases}$$