

Time Complexity

Objectives

- In this lecture, we focus on problems that are computable, and investigate the amount of **time** required to solve these problems
 - Later, we will investigate the amount of space, and other resources required to solve a problem
- Before that, we will review the big- O , small- o , big- Ω , and small- ω notations

Big-O and Big- Ω Notations

Definition: Let f and g be functions that maps \mathbb{N} to \mathbb{R}^+ . We say $f(n) = O(g(n))$ if there exists positive integers c and n' such that for every $n \geq n'$, $f(n) \leq cg(n)$.

When $f(n) = O(g(n))$, we say $g(n)$ is an asymptotic upper bound for $f(n)$

Big-O and Big-Ω Notations

We say $g(n) = \Omega(f(n))$ if $f(n) = O(g(n))$

Important: $f(n) = O(g(n))$

is a special notation, so that we will never write $O(g(n)) = f(n)$ instead

Although, we can write something like:

$f(n) = O(g(n)) = O(h(n))$, which means:

$f(n) = O(g(n))$, and $g(n) = O(h(n))$

Small-o and Small- ω Notations

Definition: Let f and g be functions that maps \mathbb{N} to \mathbb{R}^+ . We say $f(n) = o(g(n))$ if

$$\lim_{n \rightarrow \infty} f(n) / g(n) = 0$$

We say $g(n) = \omega(f(n))$ if $f(n) = o(g(n))$

Examples

Is the following true?

1. $5n^2 + 1002n + 17 = O(n^2)$

2. $\log_3 n = O(\log n)$

3. $\log n = O(\log_3 n)$

4. $\log n = O(n^{0.00001})$

5. $\log(n^2 \log n) = O(\log n)$

6. $2^n = O(3^n)$

7. $3^n = O(2^n)$

8. $n^{1/(\log n)} = o((n^{1/(\log n)})^2)$

the Limit Rule

Suppose $L \leftarrow \lim_{n \rightarrow \infty} f(n)/g(n)$ exists (may be ∞)

Then if $L = 0$, then f is $O(g)$

if $0 < L < \infty$, then f is $\Theta(g)$

if $L = \infty$, then f is $\Omega(g)$

To compute the limit, the standard **L'Hopital rule** of calculus is useful: if $\lim_{x \rightarrow \infty} f(x) = \infty = \lim_{x \rightarrow \infty} g(x)$ and f, g are positive differentiable functions for $x > 0$, then $\lim_{x \rightarrow \infty} f(x)/g(x) = \lim_{x \rightarrow \infty} f'(x)/g'(x)$ where $f'(x)$ is the derivative

EXAMPLE 7.6

The following are easy to check.

1. $\sqrt{n} = o(n)$.
2. $n = o(n \log \log n)$.
3. $n \log \log n = o(n \log n)$.
4. $n \log n = o(n^2)$.
5. $n^2 = o(n^3)$.

However, $f(n)$ is never $o(f(n))$.

Analyzing Algorithms

Let A be the language $\{ 0^k 1^k \mid k \geq 0 \}$, and we have seen that A is decidable before.

Below is one such TM that decides A :

M_1 = "On input string w ,

1. Scan across the tape and **reject** if 0 appears on the right of a 1
2. Repeat if both 0s and 1s remain in tape
 - a. Scan the tape, cross off a 0 and a 1
3. If all 0s and 1s are crossed, **accept**.
Otherwise, **reject**."

Analyzing Algorithms (2)

How many steps will M_1 need to decide if w is in A or not? Let n be the length of w

- Step 1 takes at most $O(n)$ steps
- Step 2 will repeat at most $n/2$ times, each time taking $O(n)$ steps
 - In total, Step 2 requires $O(n^2)$ steps
- Step 3 takes $O(n)$ steps

Thus, M_1 needs $O(n^2)$ steps to decide if w is in A or not

Running time of a DTM

- Number of steps the machine will take on input whose size is n .
- To formalize ...

Running Time

Definition: Let M be a deterministic Turing machine that halts on all inputs. The running time of M is the function $f:N \rightarrow N$, where $f(n)$ is the maximum number of steps that M uses on any input of length n

If $f(n)$ is the running time of M , we say M runs in time $f(n)$, and M is an $f(n)$ -time TM

Time Complexity Class

Definition: Let $t: \mathbb{N} \rightarrow \mathbb{R}^+$ be a function. We define the time complexity class, $\text{TIME}(t(n))$, to be the collection of all languages that are decidable by an $O(t(n))$ -time Turing machine

In the previous example, M_1 is an $O(n^2)$ time TM, so that the language $A = \{0^k 1^k \mid k \geq 0\}$ is in $\text{TIME}(n^2)$

Analyzing Algorithms (3)

Can we decide $A = \{ 0^k 1^k \mid k \geq 0 \}$ faster?

Below is another TM that decides A :

M_2 = "On input string w ,

1. If 0 appears on the right of a 1, **reject**
2. Repeat if both 0s and 1s remain in tape
 - (i) If total # of 0s and 1s is odd, **reject**
 - (ii) Scan the tape, cross off every other 0.
Then cross off every other 1.
3. If all 0s and 1s are crossed, **accept**.
Otherwise, **reject**."

Analyzing Algorithms (4)

Question 1: Why M_2 can decide A correctly?

Question 2: What is running time of M_2 ?

- Step 1 and Step 3 takes $O(n)$ steps.
- For each time Step 2 is repeated, # of 0s is halved \rightarrow repeated for $\log n$ times
- Each time Step 2 is run, it takes $O(n)$ steps \rightarrow in total takes $O(n \log n)$ steps

Thus, the running time of M_2 is $O(n \log n)$

Analyzing Algorithms (5)

This implies that A is in $\text{TIME}(n \log n)$

Question 1: Earlier, we show that A is in $\text{TIME}(n^2)$... Is there a contradiction??

Question 2: Can we find a TM that decides A faster? That is, in $o(n \log n)$ time?

- The answer is NO... (if TM just have a single tape)
- In fact, it is shown that if a language can be decided by a single-tape TM in $o(n \log n)$ time, the language is regular

$M_3 =$ “On input string w :

1. Scan across tape 1 and *reject* if a 0 is found to the right of a 1.
2. Scan across the 0s on tape 1 until the first 1. At the same time, copy the 0s onto tape 2.
3. Scan across the 1s on tape 1 until the end of the input. For each 1 read on tape 1, cross off a 0 on tape 2. If all 0s are crossed off before all the 1s are read, *reject*.
4. If all the 0s have now been crossed off, *accept*. If any 0s remain, *reject*.”

Analyzing Algorithms (7)

The running time of **M3** is $O(n)$!

What we have learnt before:

Single-tape and Multi-tape TM have the same power (in terms of computability, I.e., whether a problem can be solved)

What we have learnt now:

Single-tape and Multi-tape does not have the same power (in terms of complexity, I.e., how fast a problem can be solved)

Next Time

- Complexity relationship among models
 - Single-Tape versus Multi-Tape
 - Deterministic versus Non-Deterministic
- P and NP
 - Two important classes of problems in time complexity theory