Cpts 515. 8/28/2020

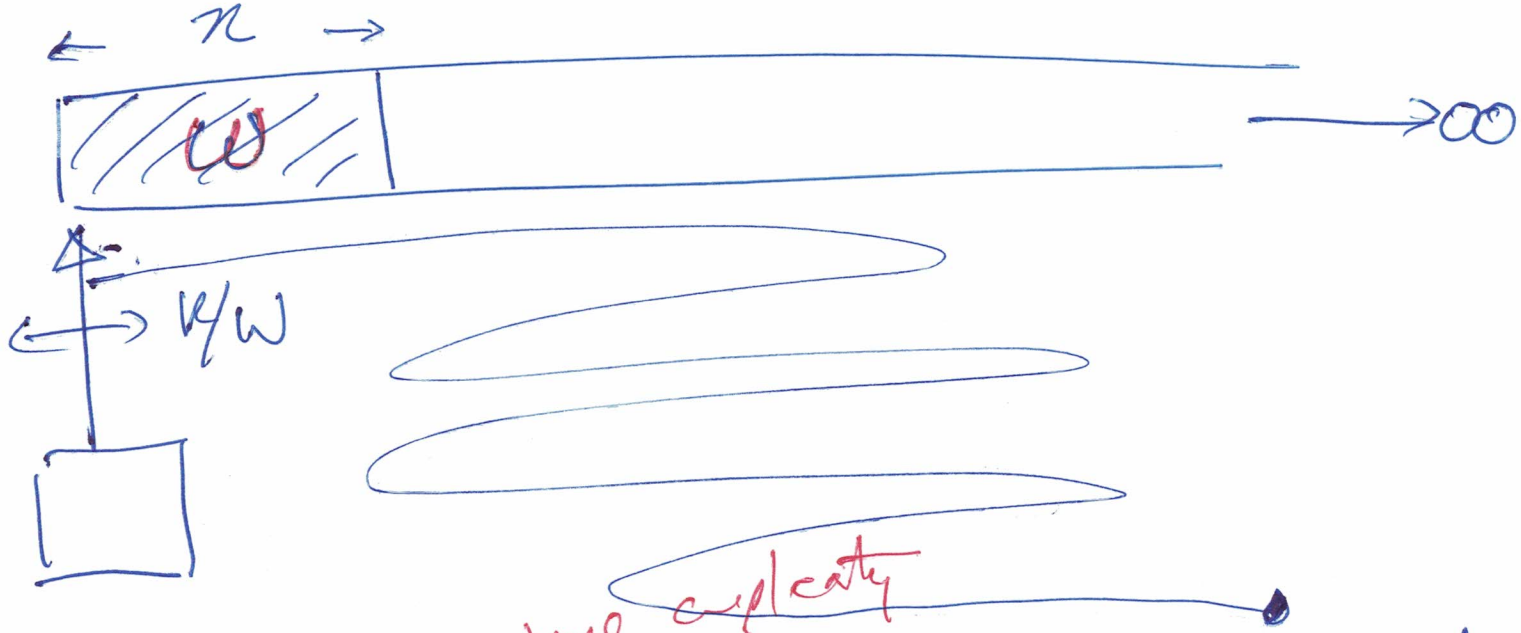
Today: mathematical def of time & space Complexities.

We use Turing machines.

Recall TM that halts on all input

||

Algorithm.



worst-case time complexity

$T(w)$ = the # of moves of the TM on w before it halts.

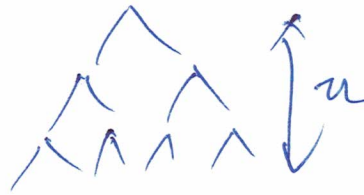
$$T(n) = \max_{|w|=n} T(w)$$

n is size of input

average-case time complexity

$$T(n) = \operatorname{avg}_{|w|=n} T(w)$$

```
int Fib(n){
  if n ≤ 1      ret 1 ;
  ret Fib(n-1) + Fib(n-2);
}
```



Input size = $\log_2 n = N$.

time = $O(2^n) = O(2^{2^N})$.

↳ Fib runs in double exp time!

(It's sup bad alg).

In reality, nobody measure Fib in this way!

Software Engg's: Let $N = n$.

Time = $O(2^n) = O(2^N)$

↳ Exp time!

$$S(n) = \max_{|w|=n} S(w) \quad \leftarrow \text{worst-case}$$

$$S(n) = \operatorname{avg}_{|w|=n} S(w) \quad \leftarrow \text{avg-case.}$$

---

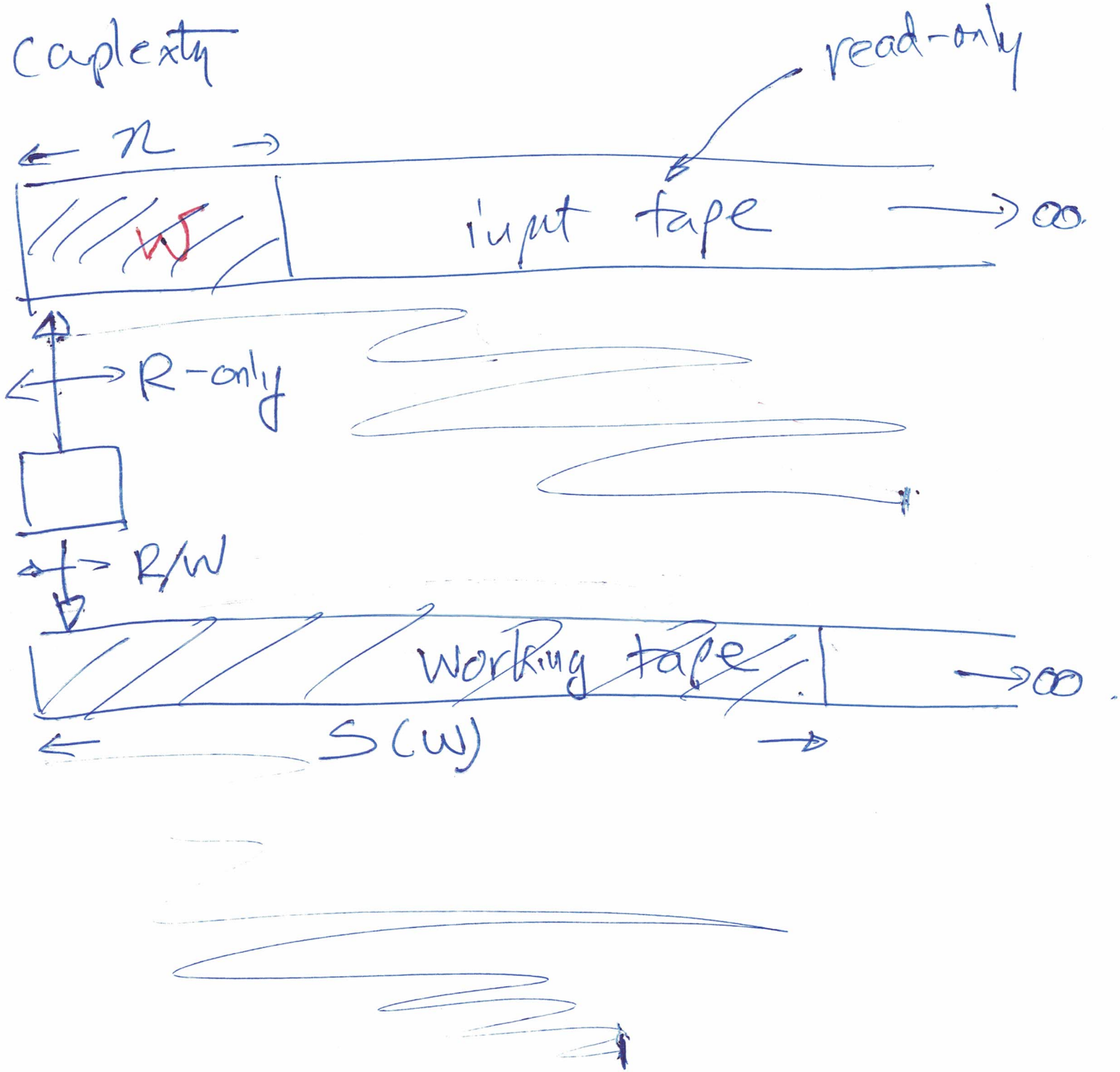
n

$\overleftarrow{abbaa}$ # $\overrightarrow{abbaa}$ | string cmp.

↑ p ↑ q

space = $O(\log n)$

Space complexity



Pascal —— Writh.

program = dg + data Stroture.

---

abbaa
abbaa

= your input.

space = O(1)

In real-word,

space = O(log n).

Big-O. $T(n) = O(n^2)$: There is $c > 0$

s.t. for almost all n, we have

$$T(n) \le c \cdot n^2.$$

for each except for finitely many.

All complexity functions are "nice". —— monotonic.

$O(\log n)$, $O(n)$, $O(n \log n)$, ...

Thm. $f(n) = O(g(n))$ iff $\exists\, c > 0$ s.t.

$$\limsup_{n \to \infty} \frac{f(n)}{c \cdot g(n)} < 1.$$

replaced by lim, b.c. complexity functions are super nice.

$$O(12 \cdot n^2 + 3n + 30) = O(n^2).$$

Proof. Take $c = 13$. Then.

$$\lim_{n \to \infty} \frac{12n^2 + 3n + 30}{13 \cdot n^2} = \frac{12}{13} < 1.$$

$$O(12n^2 + 3n + 30) = O(n^3).$$

Proof. Take $c = 1$. Then

$$\lim_{n\to\infty} \frac{12n^2 + 3n + 30}{n^3} = 0 < 1.$$

$$O(12n^2 + 3n + 30) \neq O(n).$$

Proof. Assume it is. Then, for each $c > 0$,

$$\lim_{n\to\infty} \frac{12n^2 + 3n + 30}{c \cdot n} = +\infty < 1$$

contradiction.

$$4 \cdot 2^{3n^2} = 2^{O(n^2)}$$

Proof. Take $c = 4$. Then

$$\lim_{n \to \infty} \frac{4 \cdot 2^{3n^2}}{2^{4 \cdot n^2}} = 0 < 1.$$

All complexity functions are written in Big-O notation!

We never write $T(n) = 2n^2 + 12n + 30$.

Instead: $T(n) = O(n^2)$.

Suppose that I have two algs solving the same problem:

	A	B
time:	$3n^2+12$	$3000n^2+12$.

Bob's alg is roughly 1000 times slower than Alice

Big-O:	$O(n^2)$	$O(n^2)$.

Bob's alg and Alice's alg are running ~~the same thing~~

$\Rightarrow$ "1000 times faster" is NOT faster!

Blum's speed-up theorem:

All algorithms that run more than linear-time

almost all algorithm.

Can be made any times faster!

⇒ Never try to improve an alg by making it 10 times faster!

What's an improvement?

$O(n^2)$ ⟶ $O(n \log n)$.

Analysis of Alg = to figure out Time & space complexity.

through solving recurrence.

?

$$\begin{cases} f(n+1) = f(n) + 1 \\ f(0) = 0 \end{cases}.$$

$$\Rightarrow \quad f(n) = n.$$

$$\begin{cases} f(n+2) = f(n+1) + f(n) \\ f(0) = f(1) = 1. \end{cases}$$

$$\Rightarrow \text{Fibonacci Julia}.$$

$$\begin{cases} f(n+1) = (n+1) \cdot f(n) \\ f(1) = 1 \end{cases}$$

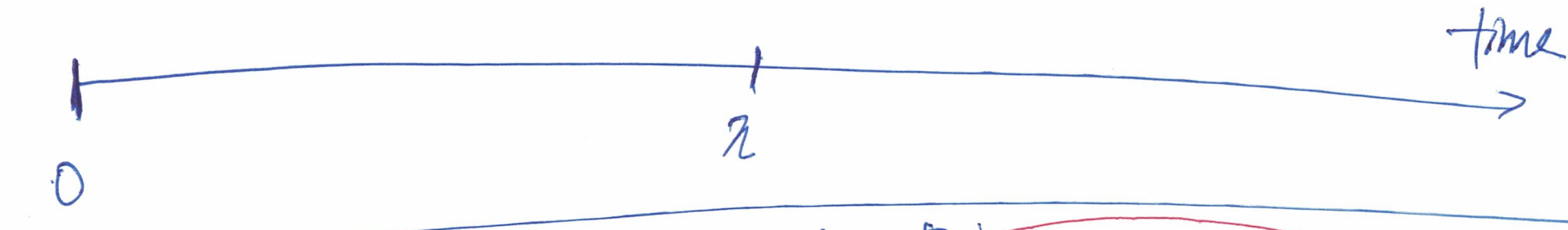
$$\Rightarrow \quad f(n) = n!$$

---

how to give estimations of recurrence functions?

ex: can you prove $f(n) \sim O(2^n)$.

general technique : use induction.

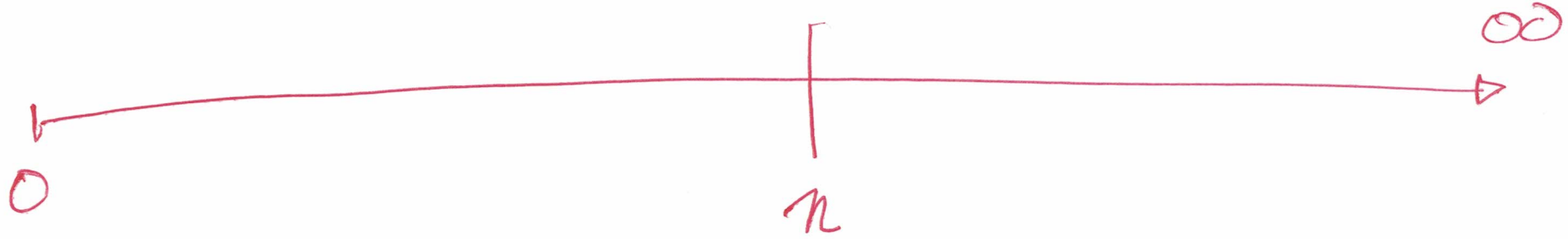


In Math, how we do this :

for each n, "the sun rises from the east at day n."

"$P(n)$"

Mathematical Induction.



Two steps:

(1). $P(0)$.

(2). For each day n, if $P(n)$, then $P(n+1)$;

$\Rightarrow$ We can conclude $\forall n.\ P(n)$.

↑ mathematical induction!

Why? nobody knows.

math-induction is NOT a theorem!

is Axiom.

why? it comes with def. of

natural numbers.