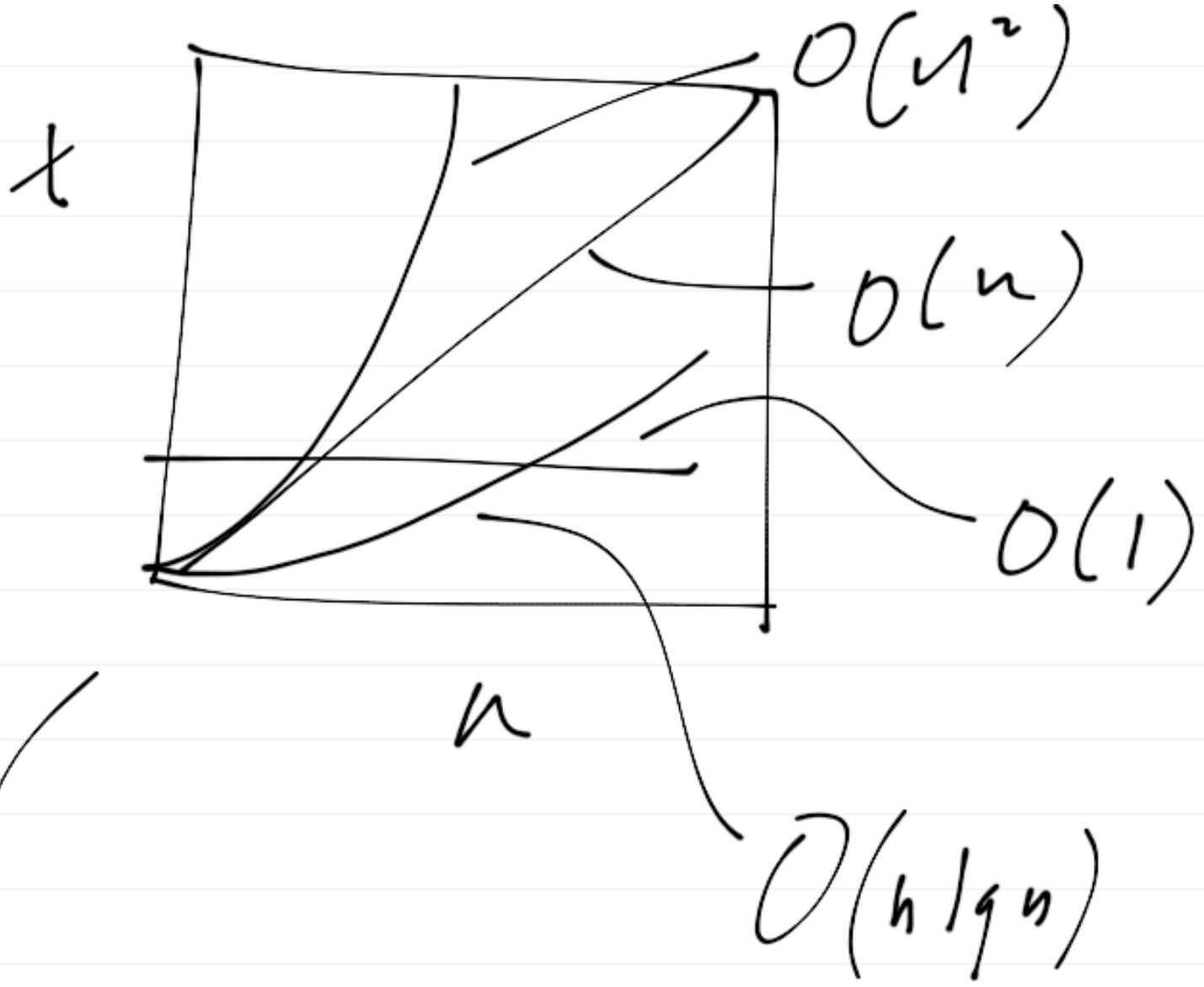
Alg. Analysis (Chp. 2)

- goal: compare & predict alg. performance (run time, given n inputs)
  - analytic approach
- empirical approach: Use time.t class to time execution



empirical graph & report

Still relevant today

e.g. graphics papers

- problems with empirical approach:
  - machine-specific
  - unreliable for few runs; run several times
    (e.g. due to Unix multi-tasking OS)

Comparisons:

$O(1) < O(\lg n) <$

Constant logarithmic

$O(\lg^2 n) < O(n) <$

iterated linear

logarithmic

$O(n \log n) < O(n^2) <$

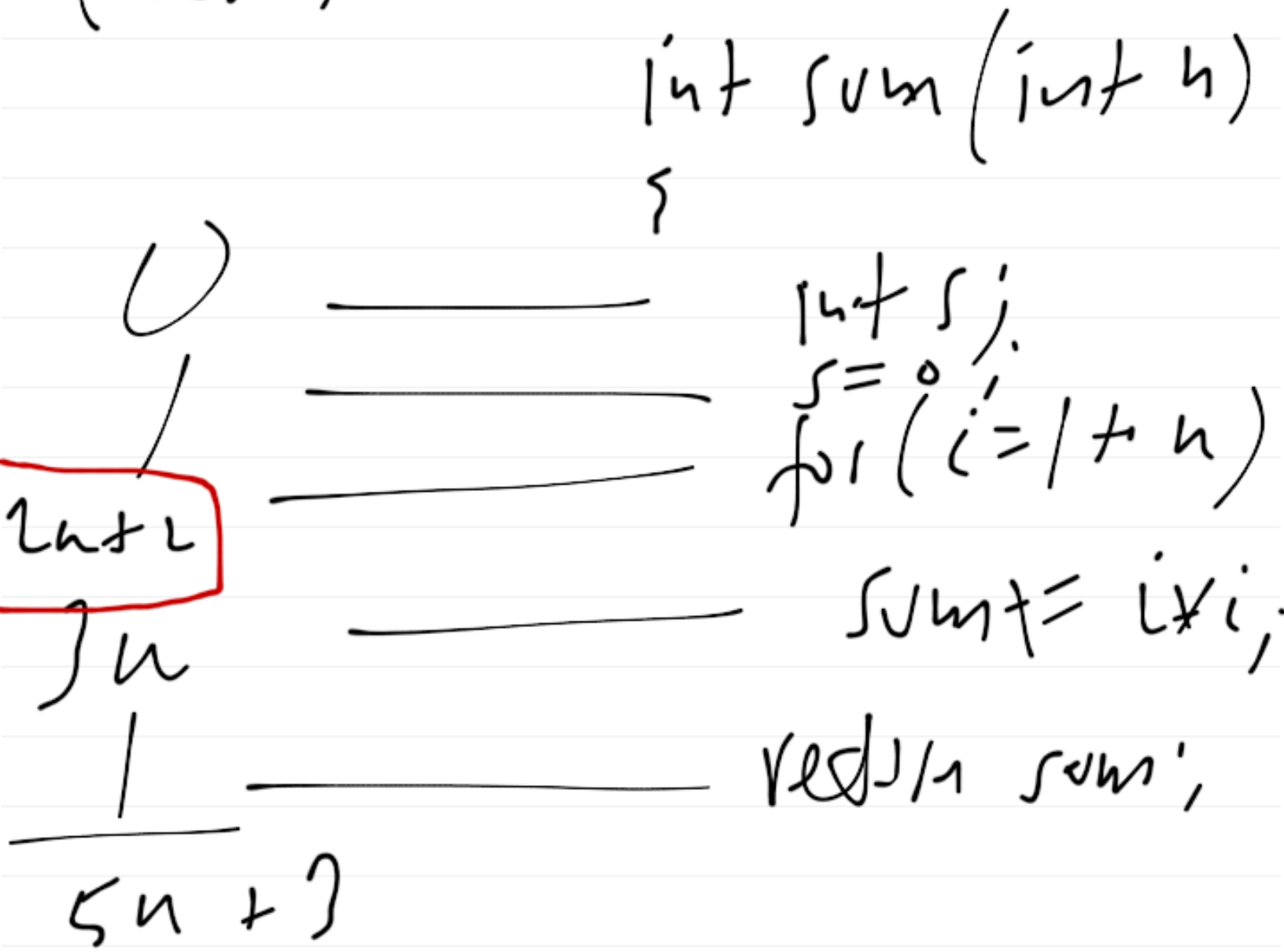
log linear quadratic

$O(n^3) < O(2^n)$

cubic exponential

- What to count?

operation (cost)



- $5n+3$:
  - reduce all constants to 1
    $\Rightarrow n+1$
  - apply $O(f(n))$ definition

$T(n) = O(f(n))$ if

$\exists\, c, n_0 \mid T(n) \leq c f(n)$

when $n \geq n_0$

$n+1 \in O(n)$ ✓

$T(n)$ $O(f(n))$

$f(n) = n$

$n+1 \leq cn$ when $n \geq n_0$

(let $n_0 = 100$, $c = 2$

$100+1 \leq 2(100)$?

✓

- in practice, drop lower order terms

$$T(n) = O(2n^2 + n)$$

$$\in O(2n^2)$$

$$\in O(n^2)$$

WORST CASE

- big-oh specifies upper bound : alg can't be any worse than $O(f(n))$

- $O(f(n))$: worst-case

- $\Omega(g(n))$: best case

$T(n) = \Omega(g(n))$ if

$\exists\, c, n_0 \mid T(n) \geq c g(n)$

- lower bound

- alg. is at least as fast (slow) as $\Omega(g(n))$

- $T(n) = \Theta(h(n))$ iff

  $T(n) = O(h(n))$ &

  $T(n) = \Omega(h(n))$

- $\Theta(h(n))$ denotes

  average running

  time

- if $\Theta = \Omega$ optimal (?)

look this up

$-T(n) = o(p(n))$ if

$\forall c \; \exists n_0$ |

$T(n) < cp(n)$ when

$n > n_0$

strict inequality

($O(f(n))$ allows $\leq$, little-oh does not)

— program analysis
(rules of thumb)

1. for loops:
   - no. statements X
     no. iterations

2. nested loops:
   - count inside-out

O(n) for(i=0; i<n; i++)

O(n) for(j=0; j<n; j++)

k++;

O(n² = n × n)

3. Consecutive statements:
just add up (Σ)

4. Conditionals:

Use larger of the bounds

(overestimate rather than underestimate)

5. Recursion : the

tricky one

— need to solve

recurrence relations

e.g. fib(int n)

{

if (n ≤ 1) return 1;

else

return fib(n-1) +

fib(n-2)

}

$$T(0) = T(1) = 1$$

$$T(n) = T(n-1) + T(n-2) + 2$$

book says for $n > 4$

$$T(n) \geq \left(\frac{3}{2}\right)^n$$

inductive proof:

next time