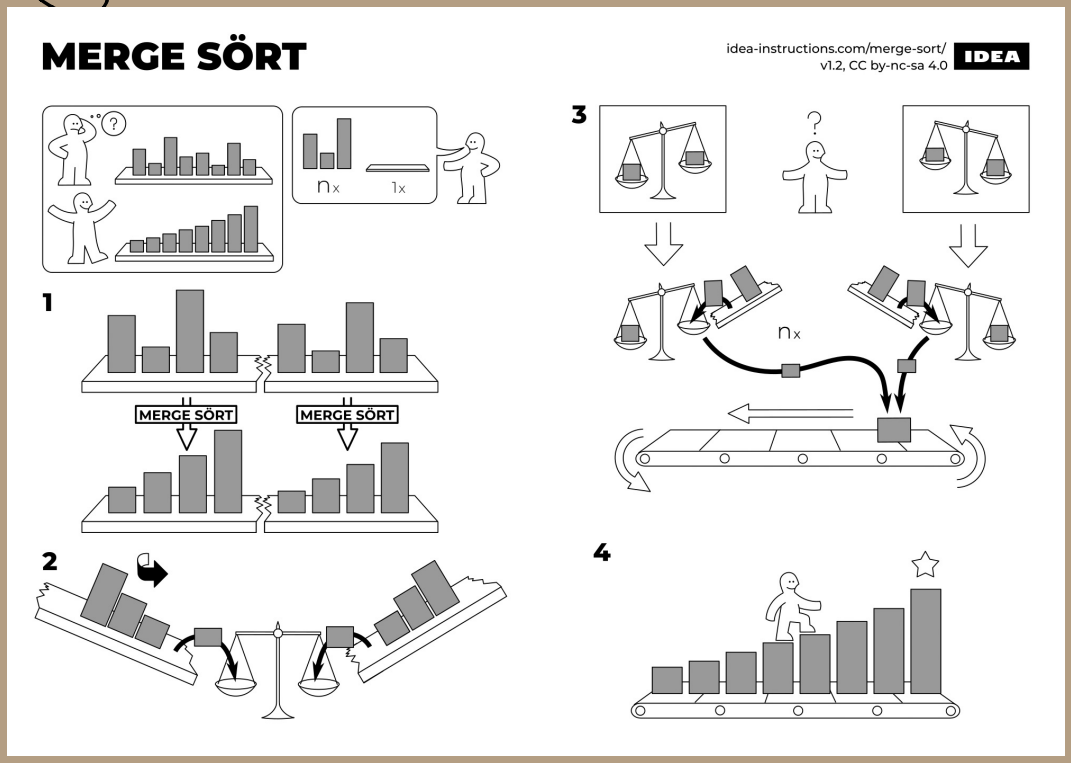
Algorithms



Recursion (day 3)

Recap:

- HW0: all should be in except for 1 extension (I'll start grading once they are all in.)
- HW1 - due Wednesday
- Office hours:
  - Monday at 1pm
  - Tuesday at 2pm
  - } discord & move to zoom if needed
  - Friday at 10am (in class zoom)

A note on studying:
Many strategies!

Recursion Trees: $T(k) = 3T(\frac{k}{2}) + k^2$

Let's start with an example.

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

n^2: nested for loops

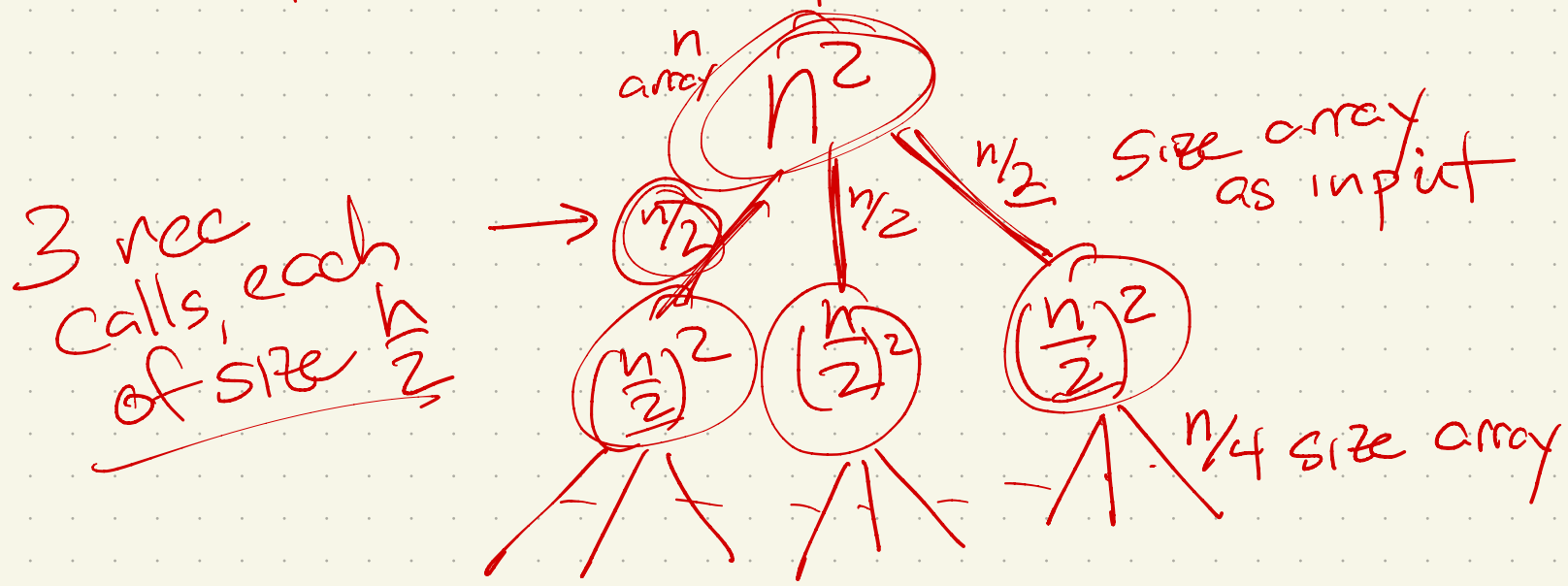
How can I "visualize" the time spent?

→ array of size n

divide into 3 of size $\frac{n}{2}$, $\frac{n}{2}$, $\frac{n}{2}$

& make 3 recursive calls

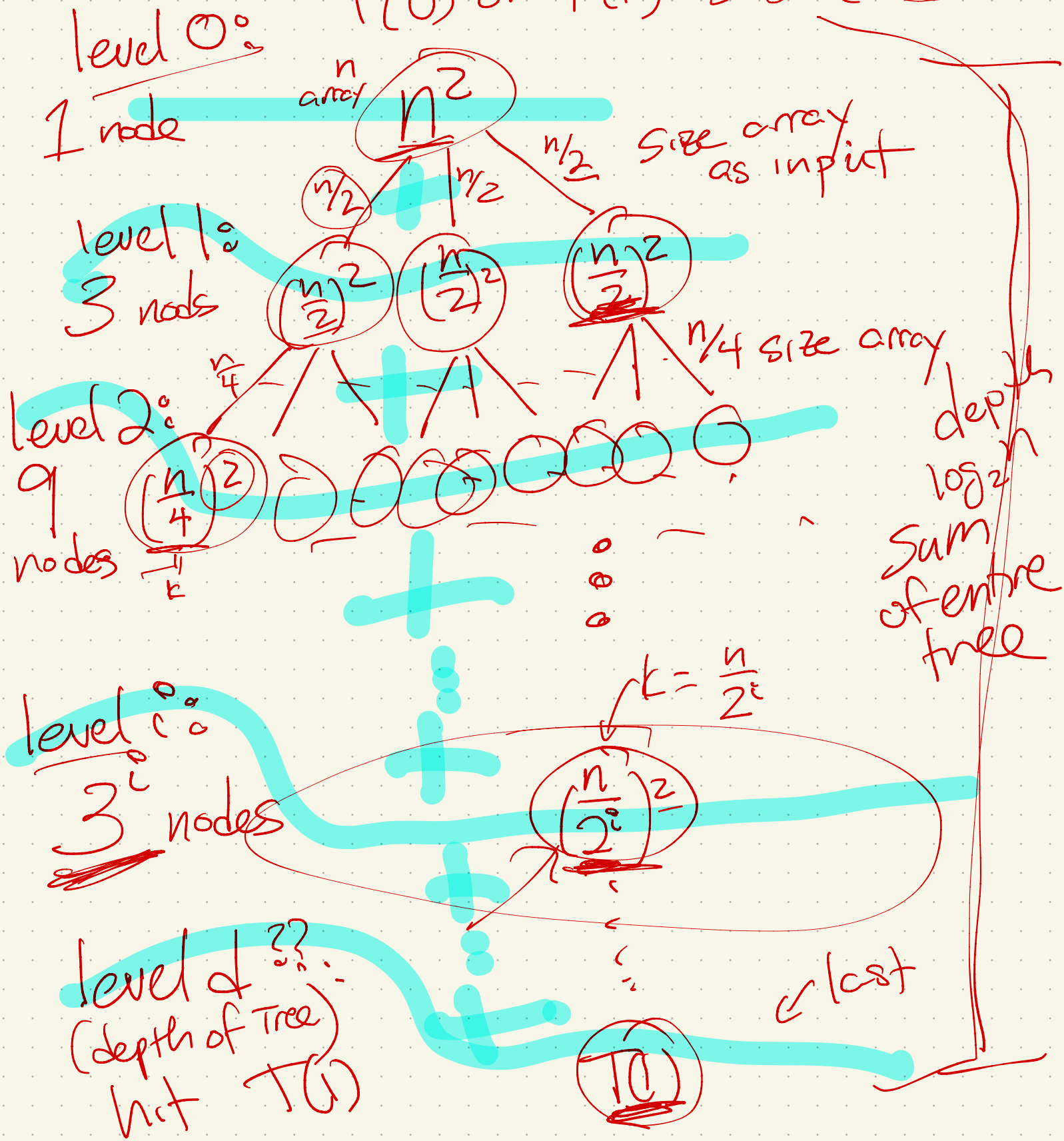
think about "top level"



(cont): Solve $T(n)$, where

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

$T(0)$ or $T(1)$ base case



Sum of all work in tree

$$= \sum_{i=0}^{d} (\#\text{ nodes on level } i)(\text{amt of work per node})$$

$$= \sum_{i=0}^{\log n} (3^i)\left(\frac{n}{2^i}\right)^2$$

Solve for d: $1 = \frac{n}{2^d}$

$2^d = n$

& take log: $\log(2^d) = \log n$

$d \log 2 = d = \log_2 n$

(& use algebra & sums)

$$= n^2 \left[\sum_{i=0}^{\log n} (3^i)\left(\frac{1}{2^i}\right)^2\right] = n^2 \sum_{i=0}^{\log n} \left(\frac{3}{4}\right)^i$$

factor out non-i terms; simplify

Finally:

$$n^2 \sum_{i=0}^{\log_2 n} 3^i \cdot \left(\frac{1}{2^i}\right)^2$$

simplify

$$\left(\frac{1}{2^i}\right)^2 = \left(2^{-i}\right)^2 = 2^{-2i} = \left(\frac{1}{2^2}\right)^i = \frac{1}{4^i}$$

$$= n^2 \sum_{i=0}^{\log_2 n} \left(\frac{3}{4}\right)^i$$

"geometric series"

$-1 < C = \frac{3}{4} < 1$

descending series

$$= \left(\frac{3}{4}\right)^0 + \left(\frac{3}{4}\right)^1 + \left(\frac{3}{4}\right)^2 + \cdots$$

$$\sum_{i=0}^{\infty} C = \frac{1}{1-C}$$

constant!

$$n^2 \sum_{i=0}^{\log_2 n} \left(\frac{3}{4}\right)^i$$

$$< n^2 \sum_{i=0}^{\infty} \left(\frac{3}{4}\right)^i$$

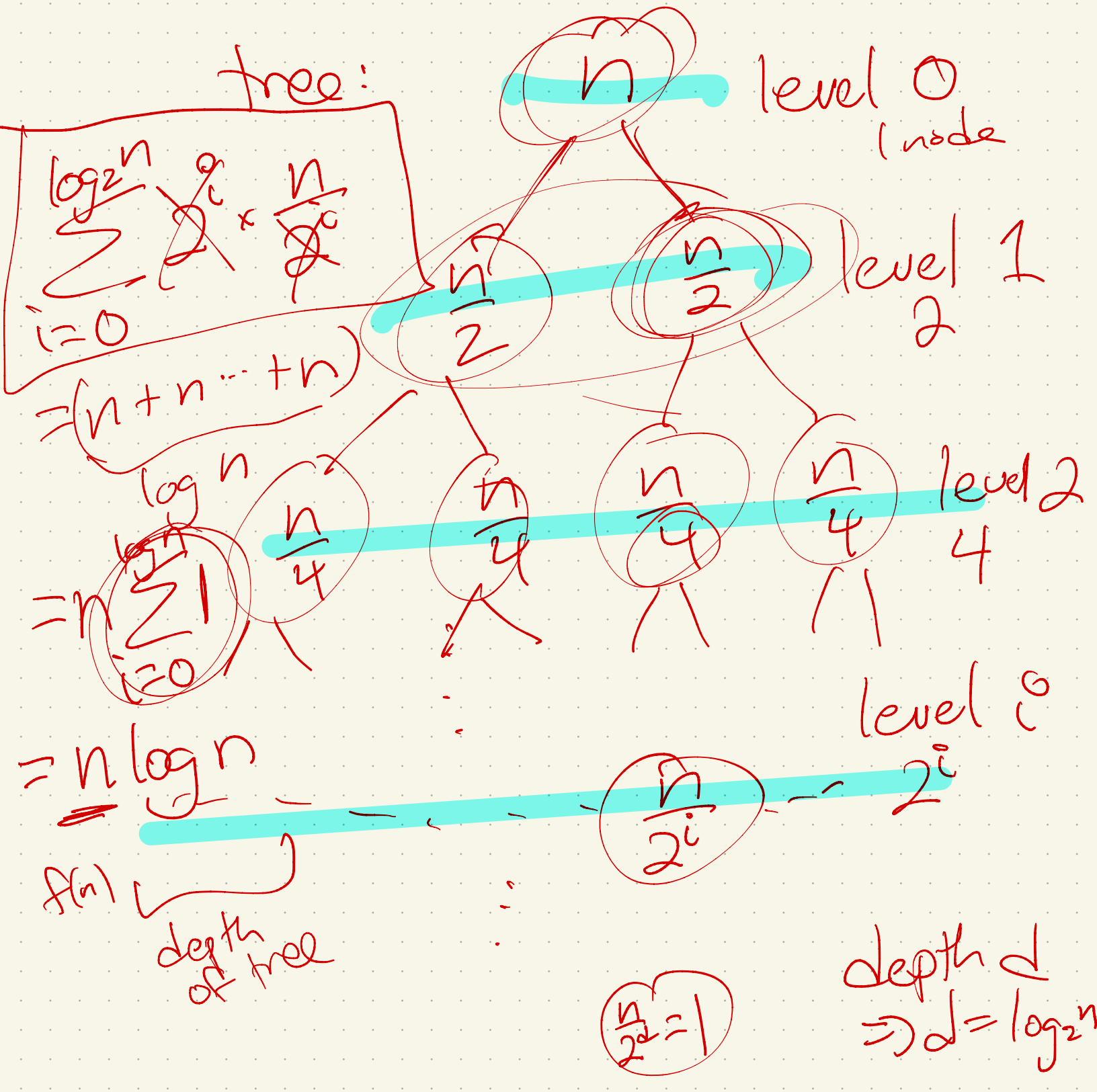
$$= n^2 \left(\frac{1}{1-\frac{3}{4}}\right)$$

$$= 4?$$

$$= O(n^2)$$

Another: merge sort

$$M(n) = 2M\left(\frac{n}{2}\right) + n$$



Note on reading:

If you don't follow the bit on ignoring floors & ceilings — don't stress!

I need you to know you can do this, but won't ask you to prove it.

Ex: discussion on domain transformation

$$T(n) = T(\lceil \tfrac{n}{2} \rceil) + T(\lfloor \tfrac{n}{2} \rfloor) + O(n)$$

$$= 2T(\tfrac{n}{2}) + O(n)$$

Next: how to generalize?

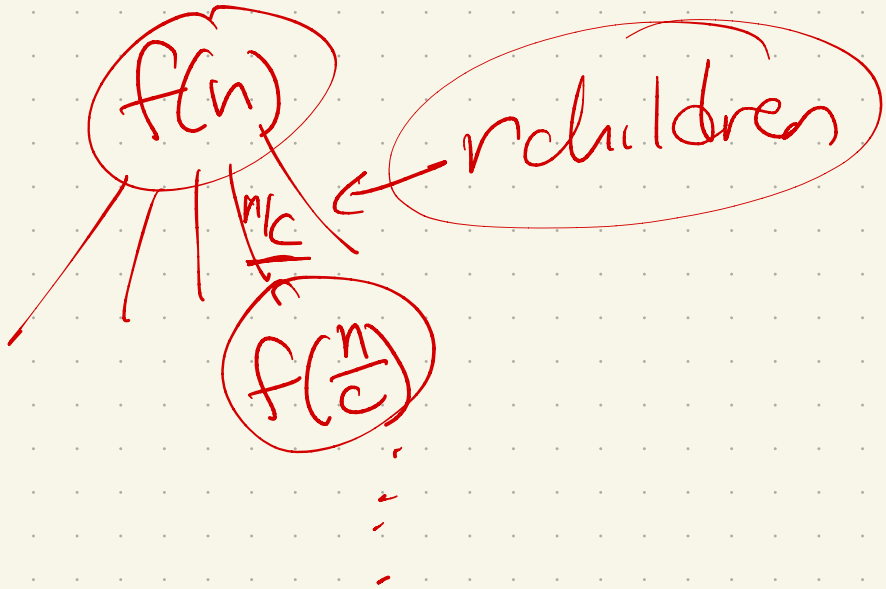
$$T(n) = r\,T\left(\frac{n}{c}\right) + f(n)$$

↑ master thm

What it means:

```
Algorithm(n):
    // code
    for i ← 1 to r
        Algorithm(n/c)
    // more code
```

total f(n) ops



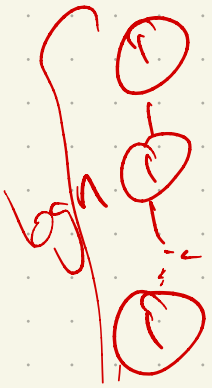
Solving those trees!

$$\sum_{i=0}^{\text{depth}} \left(r^i\right)\left(f\left(\frac{n}{c^i}\right)\right)$$

- r^i: # nodes on level i
- $f\left(\frac{n}{c^i}\right)$: work in each node
  - Ex: $\frac{n}{2^i}$ in MS
  - $\left(\frac{n}{2^i}\right)^2$ in other one

3 possibilities:

- decreasing geom. series (like my example) — $f(n)$ will dominate
- increasing: # nodes dominates — $n^{\log_c r}$
  - ie: binary search! $B(n) = B\left(\frac{n}{2}\right) + a($
- balanced — like mergesort



Other examples

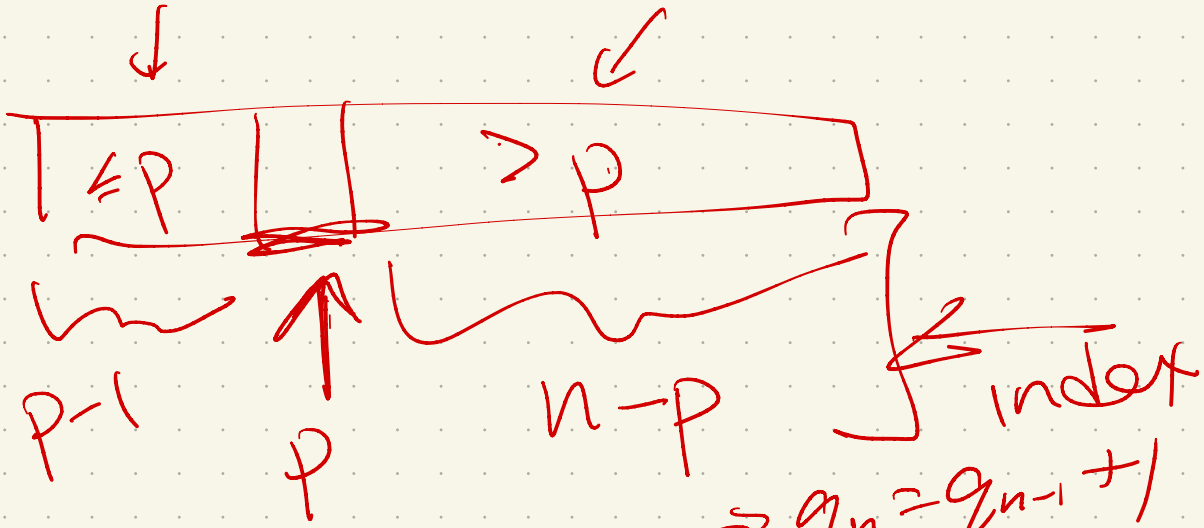
Medians: find "middle" element. $k = \frac{n}{2}$

Two were covered:

looking for any k

```
QUICKSELECT(A[1..n], k):
  if n = 1
      return A[1]
  else
      Choose a pivot element A[p]
      r ← PARTITION(A[1..n], p)
      if k < r
          return QUICKSELECT(A[1..r−1], k)
      else if k > r
          return QUICKSELECT(A[r+1..n], k−r)
      else
          return A[r]
```

Figure 1.12. Quickselect, or one-armed quicksort



$k < p-1$

$k > p$

$q_n = q_{n-1} + 1$

$Q(n) \leq Q(n-1) + n$

"Faster" version:

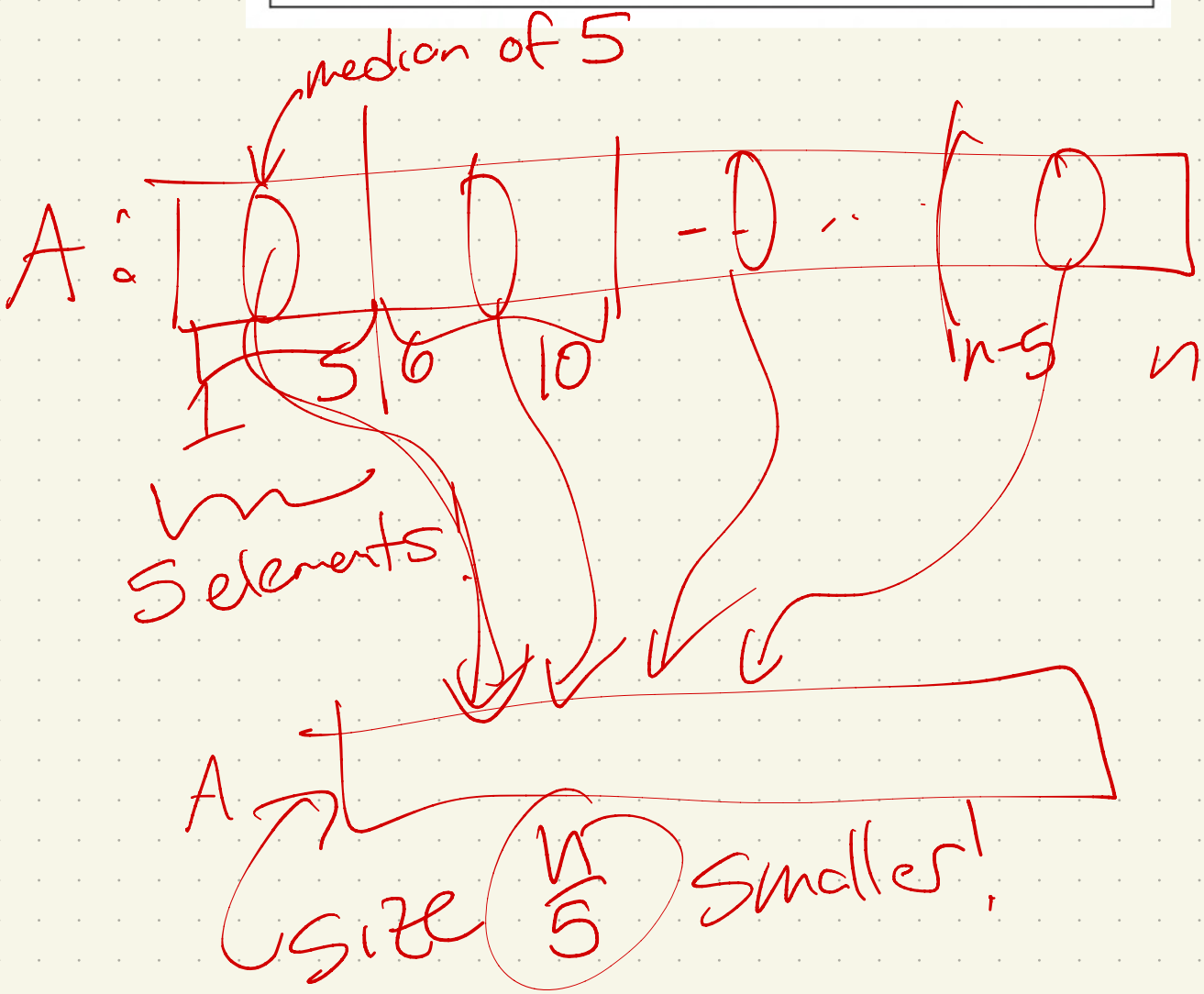
Q: why 5?

```
MomSelect(A[1..n], k):
  if n ≤ 25   ⟨⟨or whatever⟩⟩
      use brute force
  else
      m ← ⌈n/5⌉
      for i ← 1 to m
          M[i] ← MedianOfFive(A[5i−4..5i])   ⟨⟨Brute force!⟩⟩
      mom ← MomSelect(M[1..m], ⌊m/2⌋)        ⟨⟨Recursion!⟩⟩
      r ← Partition(A[1..n], mom)
      if k < r
          return MomSelect(A[1..r−1], k)      ⟨⟨Recursion!⟩⟩
      else if k > r
          return MomSelect(A[r+1..n], k−r)    ⟨⟨Recursion!⟩⟩
      else
          return mom
```

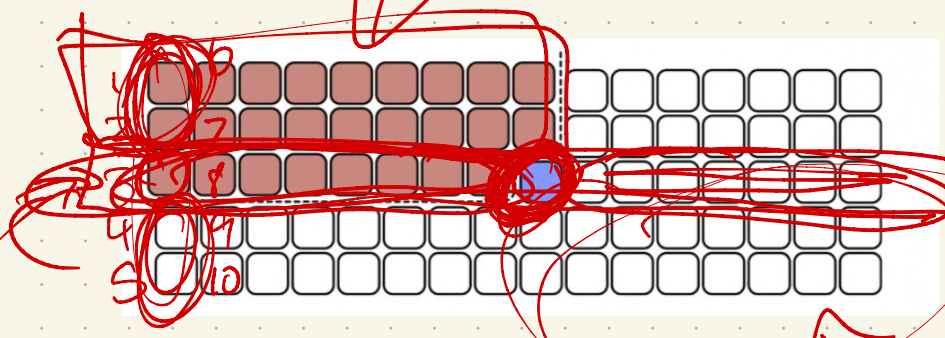
O(n)

$\frac{n}{5}$

median of 5



Its picture:



≤ blue cell

bigger than blue

that array of n/5 medians

:)

Multiplication:

```
SplitMultiply(x, y, n):
  if n = 1
      return x · y
  else
      m ← ⌈n/2⌉
      a ← ⌊x/10^m⌋;  b ← x mod 10^m        ⟨⟨x = 10^m a + b⟩⟩
      c ← ⌊y/10^m⌋;  d ← y mod 10^m        ⟨⟨y = 10^m c + d⟩⟩
      e ← SplitMultiply(a, c, m)
      f ← SplitMultiply(b, d, m)
      g ← SplitMultiply(b, c, m)
      h ← SplitMultiply(a, d, m)
      return 10^(2m) e + 10^m (g + h) + f
```

Runtime:

VS.

```
FastMultiply(x, y, n):
  if n = 1
      return x · y
  else
      m ← ⌈n/2⌉
      a ← ⌊x/10^m⌋;  b ← x mod 10^m        ⟨⟨x = 10^m a + b⟩⟩
      c ← ⌊y/10^m⌋;  d ← y mod 10^m        ⟨⟨y = 10^m c + d⟩⟩
      e ← FastMultiply(a, c, m)
      f ← FastMultiply(b, d, m)
      g ← FastMultiply(a − b, c − d, m)
      return 10^(2m) e + 10^m (e + f − g) + f
```

Runtime:

Exponentiation:

Still open!

(Amazing, right??)

The algorithms do very well:

- to compute a^n, need $O(\log n)$ multiplications

However, doesn't achieve lowest possible for every value - it's just with a constant!