$\boxed{\text{Lecture 16}}$

Parallel

Sequences

	List	Tree	Array Sequence
access position i	$O(n)$ work $O(n)$ span	$O(\log n)$ W $O(\log n)$ S	$O(1)$ work $O(1)$ span
add 1 to each (map w/constant-time f)	$O(n)$ work $O(1)$ span	$O(n)$ work $O(\log n)$ span	$O(n)$ work $O(1)$ span
sum (reduce w/constant-time fn)	$O(n)$ work $O(n)$ span	$O(n)$ work $O(\log n)$ span	$O(n)$ work $O(\log n)$ span
add to front	$O(1)$ work span x::xs	$O(\log n)$ work $O(\log n)$ span	$O(n)$ work $O(1)$ span

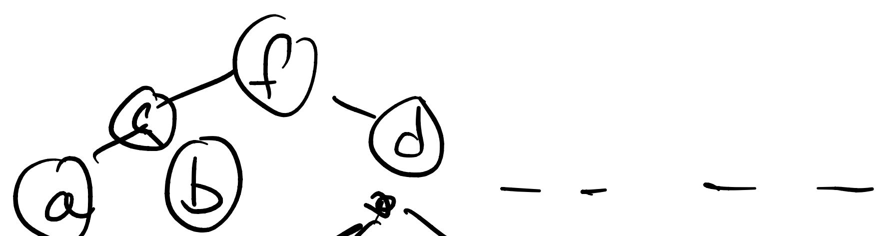
Ordered collections of elements

$\langle a, c, b, f, d \ldots \rangle$

$\quad 0 \quad 1 \quad 2 \quad 3 \quad 4 \ldots$

① Lists [a, c, b, f, d, ...]

② Tree



②a Tree

3. Array-backed sequence

w/ parallel operations

List

(* Get element at position i
assuming $0 \le i < \text{length } l$ *)

```
fun nth(l: 'a list, i: int): 'a =
  case (l, i) of
       (x::xs, 0) => x
     | (x::xs, i) => nth(xs, i-1)
     | ([], _) => raise some error
```

only happen off spec

(* E.g. nth([4, 7, 2, 6], 1) = 7 *)

n length of list

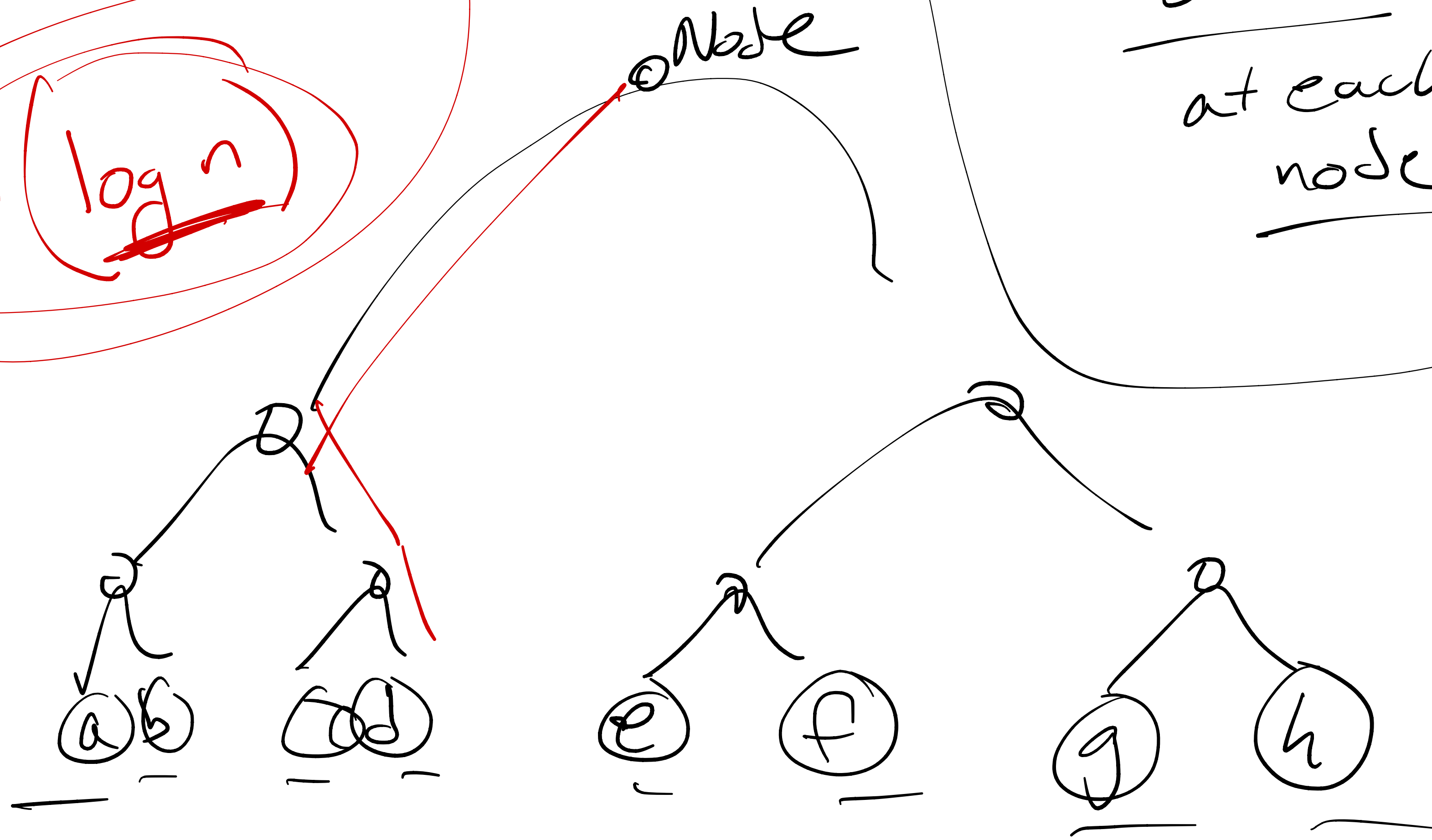
Work $O(n)$

Span $O(n)$

If tree is balanced:

O(log n)

If you store size at each node

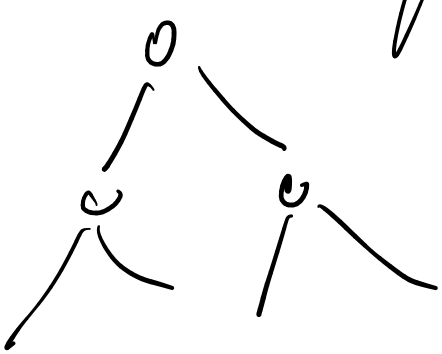


work $O(n)$

Span $O(\frac{\log_2}{})$

If balanced

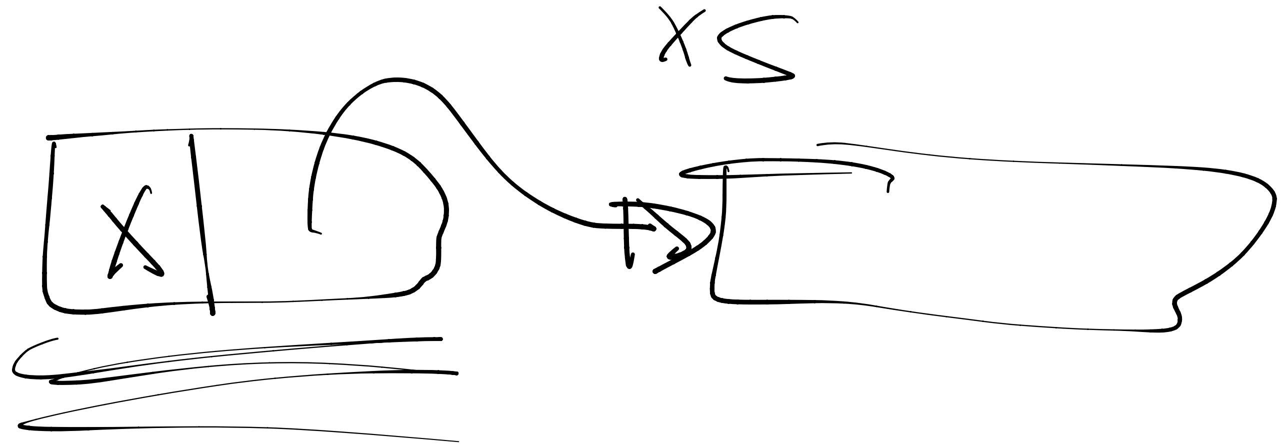
```
fun add1(t) =
  case t of
    Empty => Empty
    Leaf x => Leaf(x+1)
    Node(l,r) => Node(add1 l,
                      add1 r)
```

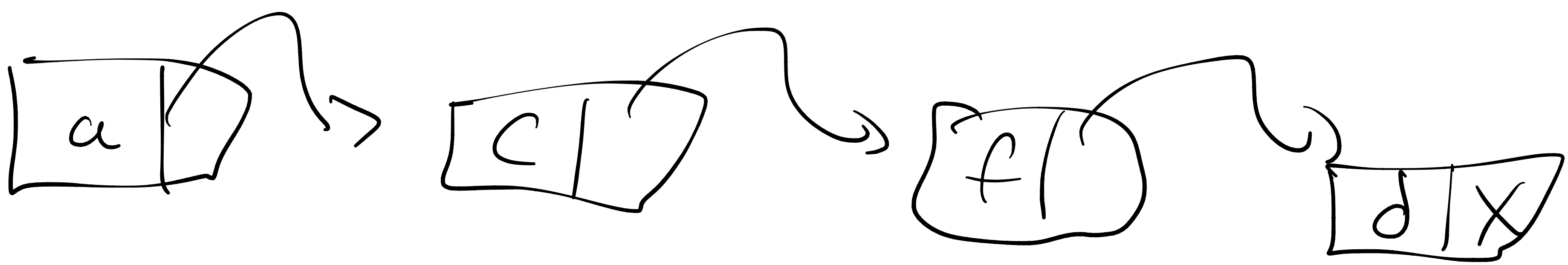


```
fun sum(t) =
  case t of
    Empty => 0
  | Leaf x => x
  | Node(l,r) => (sum l + sum r)
```

x :: xs

list

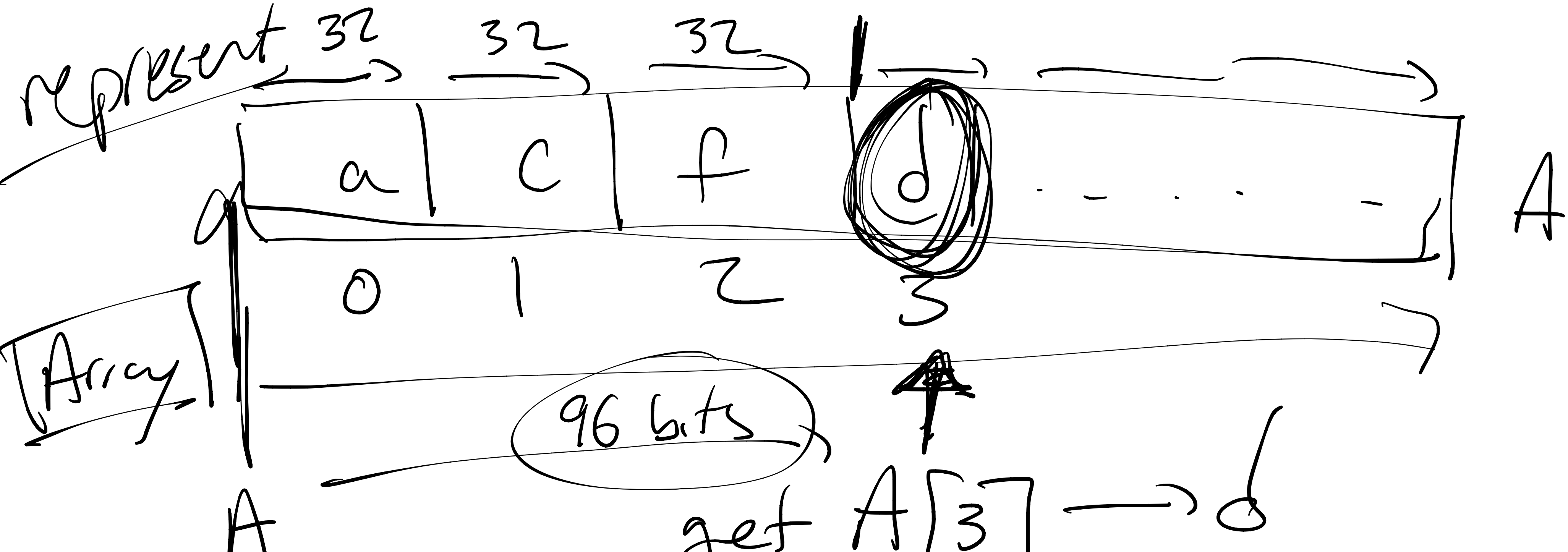




Array sequence

<u>idea:</u>

$\langle \overset{32\text{ bits}}{a}, c, f, d \ldots \rangle$



get $A[3] \longrightarrow d$

Idea:

lists support HOFs
map
reduce
⋮

trees supports HOFs
map
reduce
⋮

array
sequences support HOFs
map
reduce
⋮

Parallelize those!

type 'a Seq.seq

"module"

"sequence"

'a list
'a tree
'a Seq.seq

val Seq.length : 'a Seq.seq -> int

val Seq.nth : int * 'a Seq.seq -> 'a

val Seq.map : ('a -> 'b) * 'a Seq.seq -> 'b Seq.seq

val Seq.reduce : ('a * 'a -> 'a)

* 'a

* 'a Seq.seq

-> 'a

combine

base case

Abstract description of

both the

behavior + cost

of each function

$\boxed{\text{Length}}$ Model of sequence

$\langle x_0, x_1, x_2 \dots x_{n-1} \rangle$

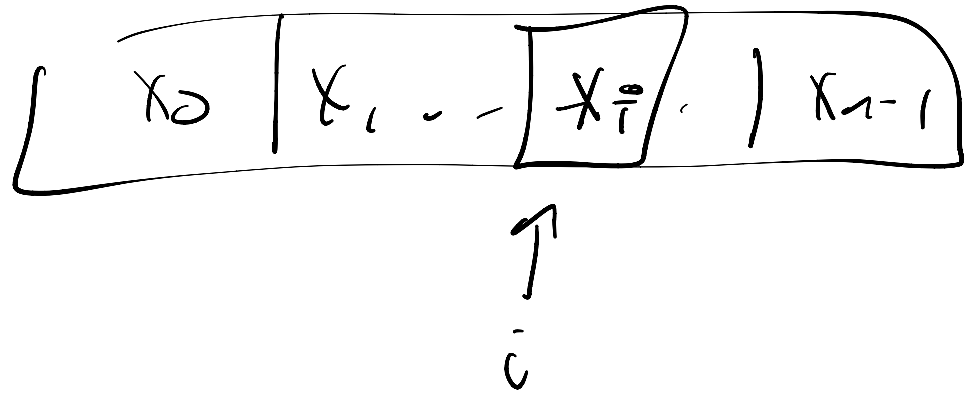
Seq.length $\underline{\langle \underline{x_0}, \underline{x_1}, \underline{x_2} \dots \underline{x_{n-1}} \rangle} = \underline{n}$

Work

Span

$\boxed{O(1)}$

$\boxed{\text{Nth}}$



$$\text{Seq.nth}(i, \langle x_0, \ldots, x_{n-1} \rangle) = x_i$$

<u>work</u>

$O(1)$

<u>span</u>

$\boxed{\text{Map}}$

If f is constant-time:

Work $O(1)$

Span $O(1)$

Behavior

$$map(f, \langle x_0, x_1, \ldots, x_{n-1} \rangle)$$

$$= \langle f\,x_0,\ f\,x_1,\ f\,x_2, \ldots, f\,x_{n-1} \rangle$$

$\boxed{\text{work}}$ general: sum of the work of $f(x_i)$ for all i

$\boxed{\text{span}}$ maximum of the span of $f(x_i)$ for all i

Reduce

$$\text{reduce}(\oplus, \odot, \langle x_0, x_1, \ldots, x_{n-1} \rangle)$$

$$= x_0 \oplus x_1 \oplus x_2 \oplus x_3 \oplus \cdots \oplus x_{n-1}$$

$$\text{or} = \odot$$

$$\text{reduce}(+, 0, \langle 1, 2, 3, 4 \rangle)$$

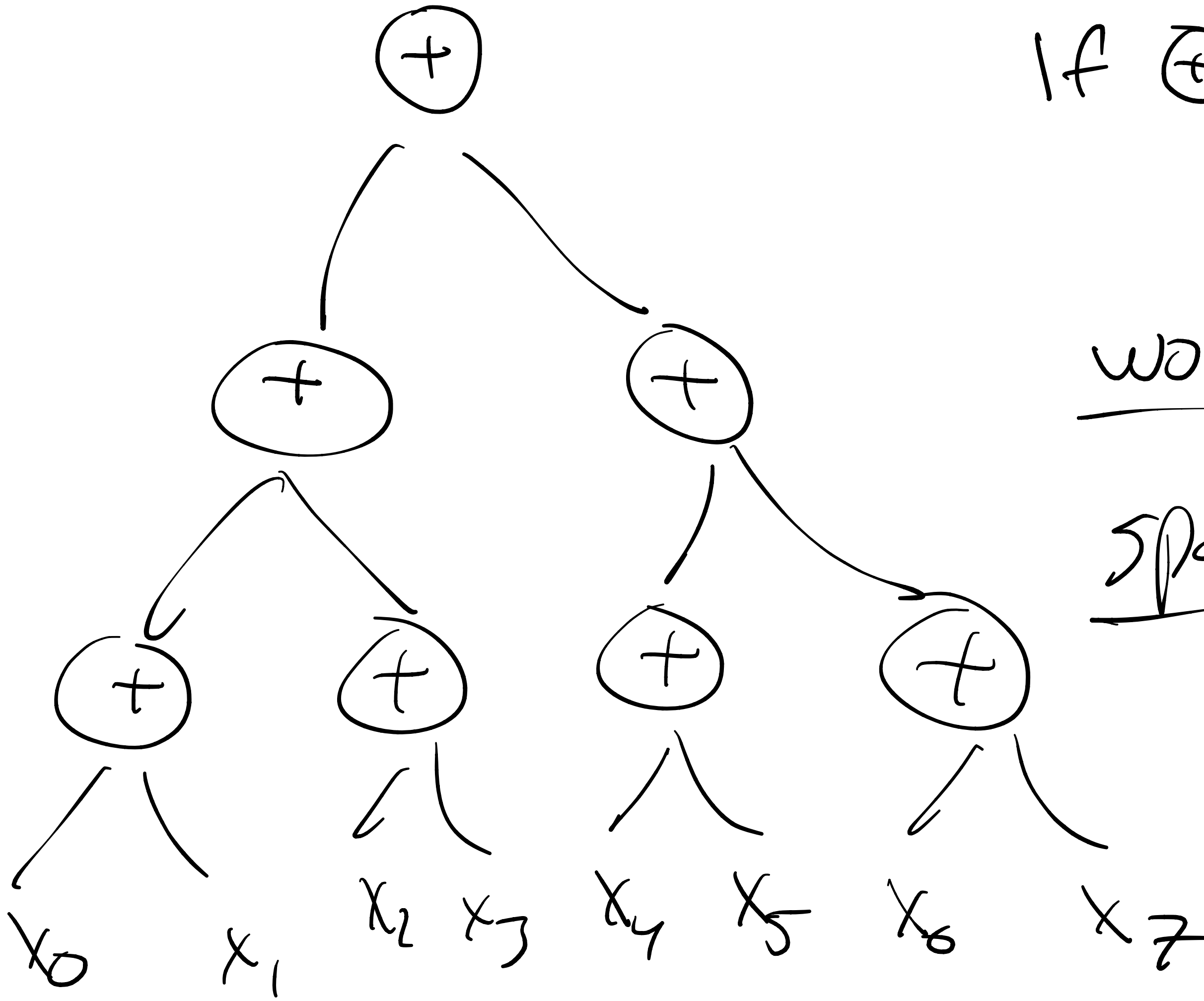
$$1 + (2 + (3 + 4))$$

$$((1 + 2) + 3) + 4$$

$$(1 + 2) + (3 + 4)$$

1) "associative"
grouping doesn't matter

2) fix a parenthesization
→ balanced



If $\oplus$ is $O(1)$:

work $O(n)$

span $O(\log n)$

If $\oplus$ is not constant:

"fun reduce($\oplus$, 0, s) =
  case s of
    <> => 0
    | <x> => x
    | s => reduce($\oplus$, 0, first half)
      $\oplus$ reduce($\oplus$, 0, second half)"

Write recurrence + solve using tree method

Problem: Add all #s in an (int Seq.seq) Seq.seq

row1 $\longrightarrow$

row2 $\longrightarrow$

row3 $\longrightarrow$

row4 $\longrightarrow$

1.
```
fun sum(s: int Seq.seq): int =
  Seq.reduce(fn (x,y) => x+y,
    0,
    s)
```

2.
```
fun count(c: (int Seq.seq) Seq.seq): int =
  sum(Seq.map(sum, c))
```

int Seq.seq

$n \times n$ total $= n^2$ numbers

$$
\begin{aligned}
\langle\langle 1, 0, 1, 0\rangle, &\longrightarrow \langle 2, \\
\langle 0, 1, 1, 0\rangle, &\longrightarrow 2, \\
\langle 1, 0, 0, 1\rangle, &\longrightarrow 2, \\
\langle 1, 1, 1, 1\rangle\rangle &\longrightarrow 4\rangle
\end{aligned}
$$

$\parallel$

10

count $\longrightarrow$ 10

	input size	work	span
inner sum	n	$O(n)$	$O(\log n)$
map	$n \times n$ grid	$O(n^2)$	$O(\log n)$
outer sum	n	$O(n)$	$O(\log n)$
overall: add them!	$n \times n$ grid	$O(n^2) + O(n) = O(n^2)$	$O(\log n)$