- Region Sort (SPAA'19)

Today $\boxed{2/2/23}$

- parallel prefix sum (scan)
- mergesort : <u>merge</u>

Scan

Input: - sequence A,

- associative function f
- left identity element $\perp$ (I)

$A = [1, 0, 0, \boxed{1}, 1, 100]$

$f = +$ $\quad 0 \quad 1 \quad 1 \quad 1 \quad 2 \quad 3 \quad , \quad 103$

$\perp = 0$

Output: $\left(\overset{R}{[r_0, r_1, \ldots, r_{|A|-1}]}, r_{|A|} \right)$

where

$$r_i = \begin{cases} \perp & i = 0 \\ f(r_{i-1}, A[i-1]) & 0 < i \leq |A| \end{cases}$$

$\Theta(n)$ work (and depth)

$a + (b + c)$

$=$

$(a + b) + c$

$n = 2^k$

$[\dot{0}, 1, \dot{1}, 1, \dot{2}, 2, \dot{3}, 4], 4$

$[1, 0, 0, 1, 0, 1, 1, 0]$ // length n

$[1, 1, 1, 1]$ // length $n/2$

↓ recurse

$(\underline{[\dot{0}, \dot{1}, \dot{2}, \dot{3}]}, 4)$

$[0, \overset{0+A[0]}{\_}, 1, \overset{1+A[2]}{\_}, 2, \overset{2+A[4]}{\_}, 3, \overset{3+A[6]}{\_}]$ ← R

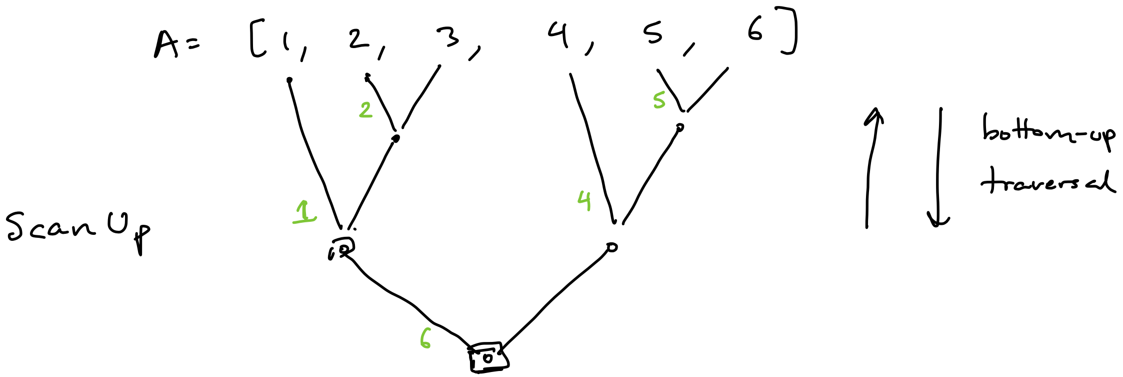
```
scan(A, f, ⊥):
    pairs ← ...                          (parallel_for)
    (pairs_rec, tot) = scan(pairs, f, ⊥)
    R = map even/odd elms as above       (parallel_for)
    return R
```

$$W_{scan}(n) = W_{scan}(n/2) + O(n) \in \Theta(n) \text{ work}$$

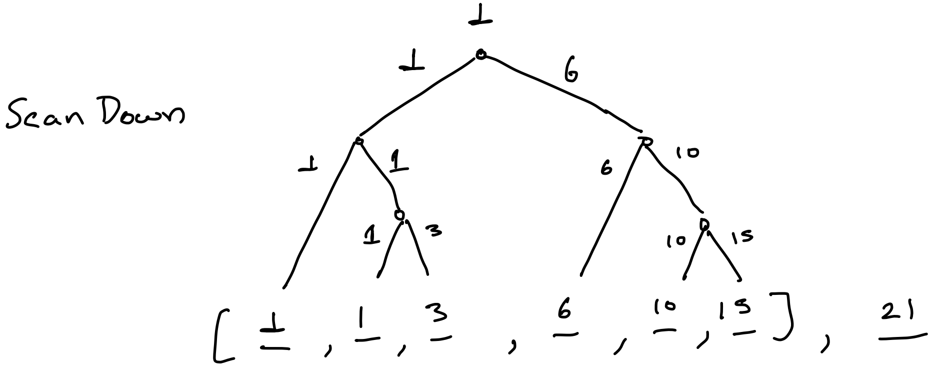
$$D_{scan}(n) = D_{scan}(n/2) + O(\log n) \in \boxed{\Theta(\log^2 n) \text{ depth}}$$

is this optimal?

We can solve it in O(n) work and O(log n) depth!



L = [1, 2, 6, 4, 5] // length n-1



Scan Up: - computes partial sums of left subtrees and stores in L

- evenly split A in the middle (m)
- split L into ranges [0; m-1], [m: n]

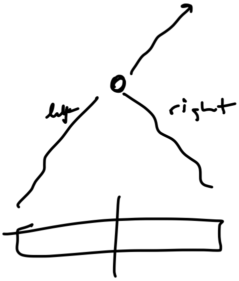
save L[m-1] for itself

Scan Down: (top-down)

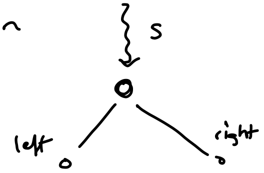
- get a value s from the parent (root uses ⊥)
- pass s to the left child
- pass $f(s, L[m-1])$ to right child

```
Scan(A, f, ⊥)
  L ← array(|A| - 1)
  Res ← array(|A|)
  total ← ScanUp(A, L, f)      ← O(log n) depth
  ScanDown(Res, L, f, ⊥)       ← "
  return (Res, total)
// ScanUp returns the sum of its range
ScanUp(A, L, f) =
  if (|A| = 1) return A[0]
  else
    n ← |A|
    m ← n/2
  (l, r) ← [ ScanUp(A[0 : m], L[0 : m-1], f) ||
             ScanUp(A[m : n], L[m : n], f) ]

  L[m-1] ← l
  return f(l, r)
```



```
ScanDown (Res, L, f, s) =
  if (|Res| = 1) then Res[0] = s        return
  else
    n ← |Res|
    m ← n/2
                                            , f
    ScanDown (Res[0 : m], L[0 : m-1], s) ||
    ScanDown (Res[m : n], L[m : n], f(s, L[m-1]))
                                            , f
    return
```

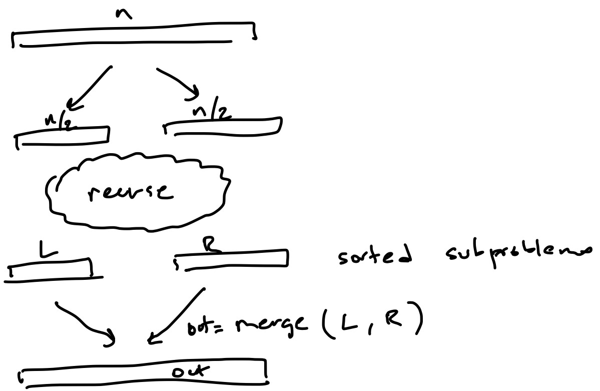


$O(n)$ work, $O(\log n)$ depth

$\downarrow$

$\underline{O(\log n \log\log n)}$

Merge Sort



$$W_{merge}(n/2, n/2) \to O(n)$$

$$W(n) = 2W(n/2) + O(n) \in \Theta(n \log n)$$

$$D(n) = D(n/2) + \underbrace{\boxed{D_{merge}(n/2, n/2)}}_{O(n)} \in \Theta(n)$$

depth

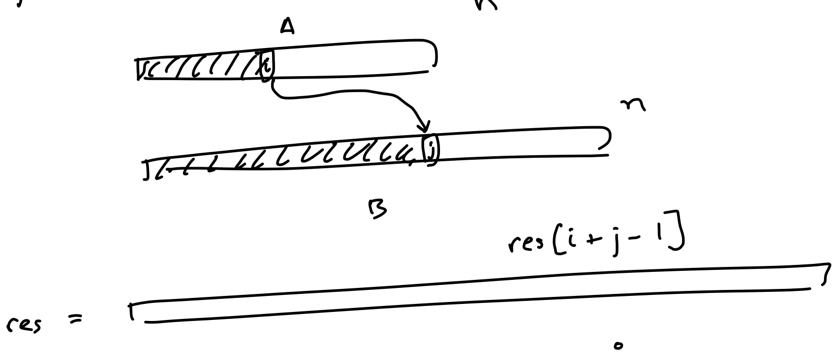
$O(n)$

$O(n/2)$

⋮

"two-finger merge"

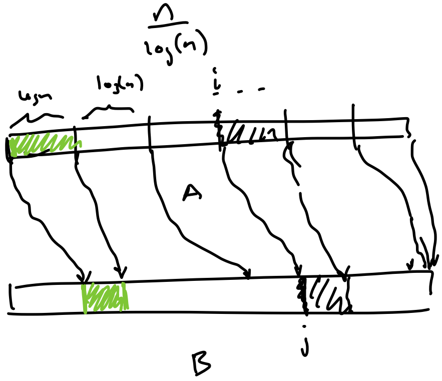
merge(A, B):



$O(n \log n)$ work

$O(\log n)$ depth

$\frac{n}{\log(n)}$ subproblems



$$O\left(\frac{n}{\log(n)} \cdot \log(n)\right) = O(n) \text{ work}$$

res[i+j]

$n + O(n^{\varepsilon})$