Lecture 3

Review / Sequence data structure
replacement of lists in sequential prog.
(lists but parallel)

aggregation
iterate - work, span are same: $O(n)$
(not desirable)
reduce $O(n)$ work, $O(\log n)$ span
↳ requires $\boxed{\text{assoc}}$ function
scan - access history
impl technique (one afternoon)
(requires $\boxed{\text{assoc}}$ also)
tabulate
(map, append)) $O(1)$ span

update $O(n)$ work, $O(1)$ span
inject (bulk update) $O(m+n)$ W, $O(\log n)$ Span

impl filter using tabulate, inject

How to prove assoc

$$f(f(x,y),z) = f(x, f(y,z))$$

Straightforward case analysis

+ Maximum contiguous subsequence sum problem
MCSS
1979 Swedish statician

given a sequence
ex: (1, 2, 0, -1, 5, 3, 4, -7)
subsequences:
14 sum

Shamus Bentley - divide & conquer solution $O(n \log n)$
not work efficient
Jay Kadane - gave $O(n)$ alg. in class

lower bound - linear work
span: not constant span
$(\lg n)$

Brute force: all possible subsequences

fn bf a =
let b = $(a[i..j] : 0 \le i \le j < |a|)$ - tabulate

c = map $(\lambda x.$ reduce "+" $x)$ b $\leftarrow$ compute sum for each one

in reduce $-\infty$ max $\cancel{b}$ c

cost - in tabulate ("b") n^2 i,j pairs
$O(n^2)$ work
$O(1)$ span
"c": $\boxed{O(n^3) \text{ work}, \quad O(\lg n) \mid \text{span}}$ (reduce)
final reduce $O(n^2)$ w, $O(\lg n)$ span

final

brute force alg:
not taking adv. of overlap of subsequences

MCSS but $\boxed{\text{starting at given index}}$

MCSS_B

i

linear work $O(n)$ W

parallel $O(1)$ or $O(\lg n)$ span

ex:

```
fun mcss_b a i =
   let b = a [i... |a|-1]               W: O(n)
       (c,s)= scan "+" 0 b              S: O(lg n)
   in
   max(reduce max -∞ c, s)
```

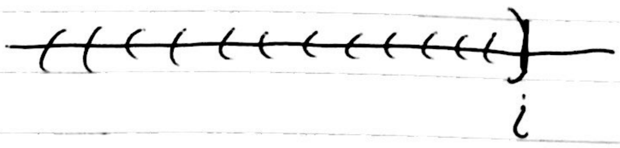
~~λ((m, j, s), n)~~
~~let s' = s + n~~
~~= if s' > m then (m, s') else~~

```
fun mcss_reduction a =
   let b = ((mcss_b a i) : 0 ≤ i < |a|)
   in reduce max -∞ b            ← tabulate (λi. mcss_b a i) |a|
   end
```

$O(n^2)$ W

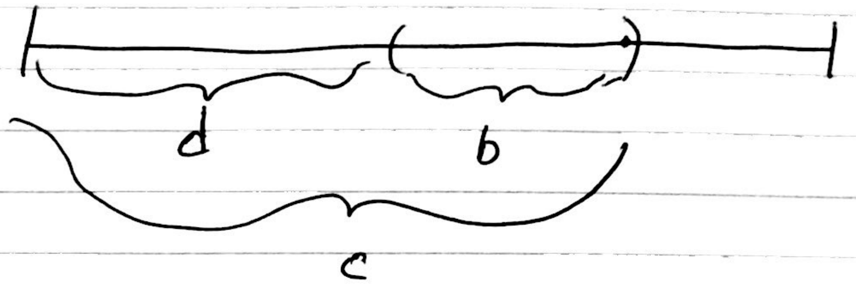
$O(\lg n)$ S

mcss_e   <u>ends</u> in particular position



observation

min d's = max b's



$$\sum b = \sum c - \sum d$$

```
fun mcss_e a i =
  let b = a[0...i]

      (c, s) = scan + 0 b

      m = reduce min ∞ c
  in
      s - m
  end
```

$O(n)$ W
$O(\lg n)$ S

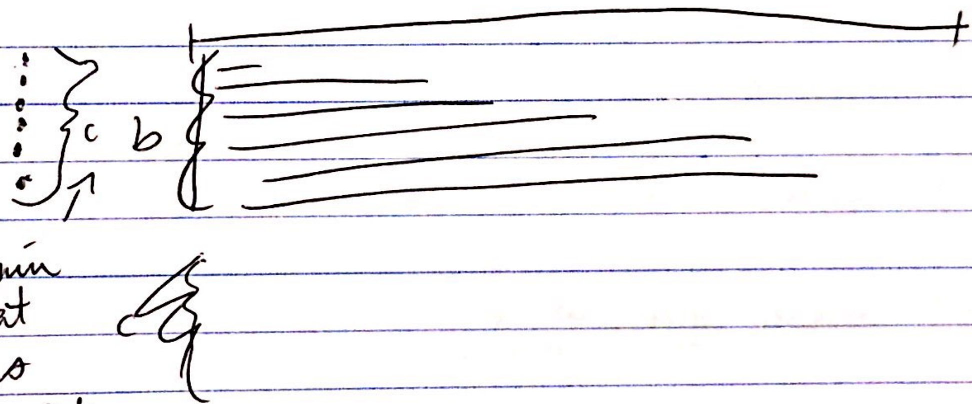
almost have done work for all ending positions

```
fun mcss a =
  let b, s = scan + 0 a
      c, _ = scan min ∞ b
      d    = ⟨ b[i] - c[i] : 0 ≤ i < |a| ⟩
  in
      reduce max -∞ d
  end
```

optimal

$O(n)$ W

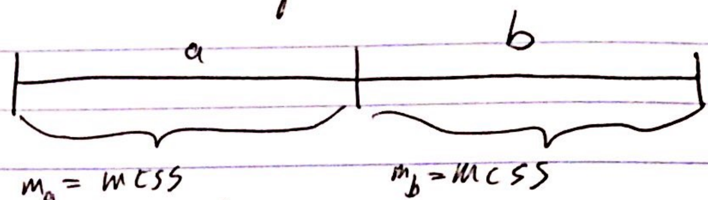
$O(\lg n)$ S



min that comes before it

d largest mcss-e[i]

divide and conquer



~~m_{ab} = best~~

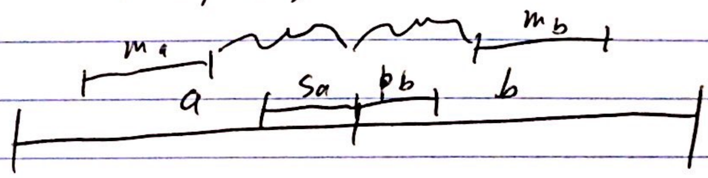
~~if $m_a > m_b$~~

~~then m_a~~

~~else m_b~~

m_{ab} = best_across (a, b)

ret max (m_a, m_b, m_{ab})



$$W(n) = 2W\left(\frac{n}{2}\right) + O(n)$$
$$S(n) = S\left(\frac{n}{2}\right) + O(1)$$

$$W(n) = O(n \lg n)$$
$$S(n) = O(\lg n)$$

"strengthening" -

here, also return sum to mid point

ret max, prefix & ~~max~~ suffix

(m_a, p_a, s_a, t_a) = mcss a

(m_b, p_b, s_b, t_b) = mcss b

return (max (m_a, m_b, $s_a + p_b$),

max (p_a, $t_a + p_b$),

max (s_b, $s_a + t_b$),

$t_a + t_b$)