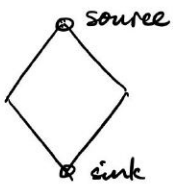
# Lec 4 More cost model & Array Sequence

## # Nested parallel model ← recursively define
note jumps & exceptions break the model

| e | W | S |
|---|---|---|
| x | 1 | 1 |
| c | 1 | 1 |
| $e_1 + e_2$ | $1 + W(e_1) + W(e_2)$ | $1 + S(e_1) + S(e_2)$ |
| $e_1 ; e_2$ | $c + W(e_1) + W(e_2)$ | $c + S(e_1) + S(e_2)$ |
| $e_1 \parallel e_2$ | $c' + W(e_1) + W(e_2)$ | $c' + \max(S(e_1), S(e_2))$ |
| if $e_1$ then $e_2$ else $e_3$ | $W(e_1) + \begin{cases} W(e_2) \\ W(e_3) \end{cases}$ | $S(e_1) + \begin{cases} S(e_2) \\ S(e_3) \end{cases}$ |

This models Sync P-RAM pretty well

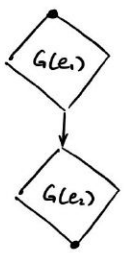
## # Dependence graph representation
└ isomorphic to above representation




$G(x)$, $G(1)$

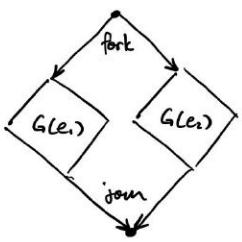
• source = sink

$G(e_1 ; e_2)$




W = # nodes
S = longest path

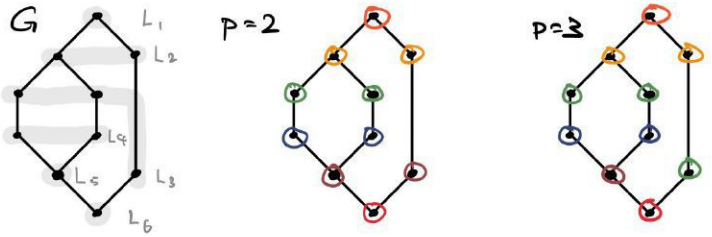
$G(e_1 \parallel e_2)$




## # Mapping nested par model to hardware

pebble game — given depence graph G, put up to p (# processors) pebble on nodes at each step s.t. all prerequisites have a pebble, with the goal of minimising # steps

Greedy strategy — always put down $\min(r, p)$ pebbles, where r = number of ready pebbles
there's always an optimal sol that's greedy (but greedy not always optimal)




Let $n$ = # nodes
$d$ = len of path

**Claims**

1. It requires at least $\max(\lceil \frac{n}{p} \rceil, d)$ steps
2. Finding optimal is NP-hard ← But we can approximate quickly
3. Any greedy strategy will take at most $\frac{n}{p} + d$ steps, so always within factor of 2 to optimal, usually better

(aka greedy scheduling theorem)

**Mapping**

Nested model
$W, S$

PRAM ($T$ = # of steps, $p$ = # of processors)

$$\max\left(\frac{W}{p}, S\right) \leq T \leq \frac{W}{p} + S$$

We want this to dominate... this happens if:

$$p < \frac{W}{S}$$

$$\text{parallelism} = \frac{W}{S}$$

**Proof for greedy scheduling theorem**

Def: node is at level $l$ if its longest path to root is $l$.

Lemma: on every step, either:
1. put down $p$ pebbels
2. finish a level

Proof AFSOC let $L_j$ be longest level that all nodes are covered
Then at $L_{j+1}$ all nodes are either done or ready.
Then if we put less than $p$ pebbels and not finish the level, we're not greedy

# Array Sequences, bottom up

Data structure : array
(other impl could use list, function, trees, ...)

Primitives for array

| | | W | S |
|---|---|---|---|
| a[i] | get i-th elem | O(1) | O(1) |
| \|a\| | get length | O(1) | O(1) |
| alloc (n) | allocate array of length n | O(1) | O(1) |
| parallelFor (pfor) | i = x to y, evaluate e(i) in parallel | $\sum_{i=x}^{y} W(e(i)) + 1$ | $\max_{i=x}^{y} S(e(i))$ |

↑ Has unavoidable side effect

Race condition : both write or one read one write
↑ Avoid this

Implementations

```
map f A =
  R = alloc |A|
  pFor i = 0..(|A|-1)
    R[i] = f A[i]
  ret R

tabulate f n =
  R = alloc n
  pFor i = 0..(n-1)
    R[i] = f i
  ret R
```