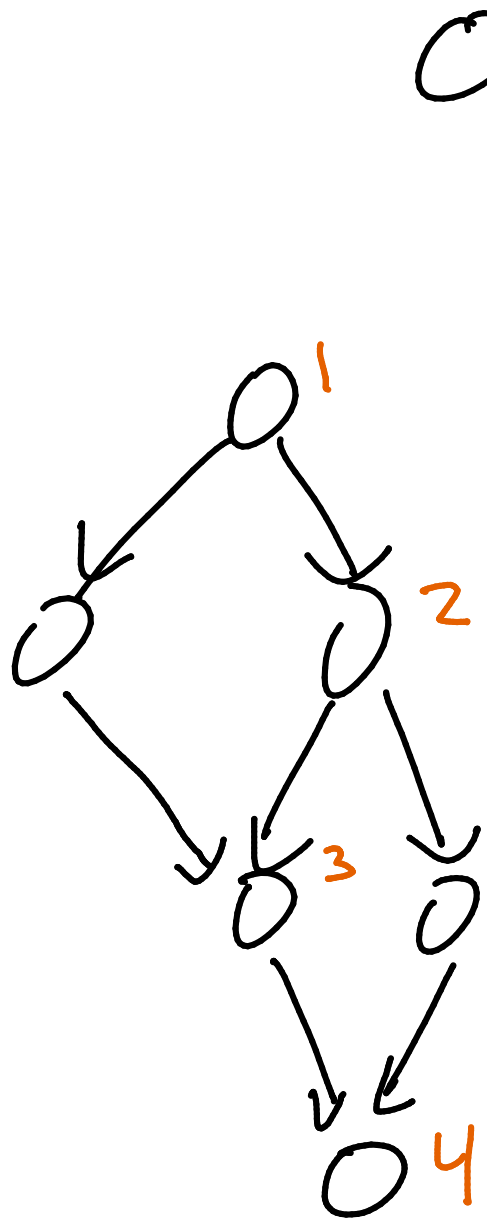
Speedup analysis - how do we think about quantifying how well we are doing?

Review on setup — fixed-time

- algorithm is collection of fixed-size steps

Create a graph w/ dependencies of one step on another



: this task depends on finishing

directed acyclic graph (DAG)

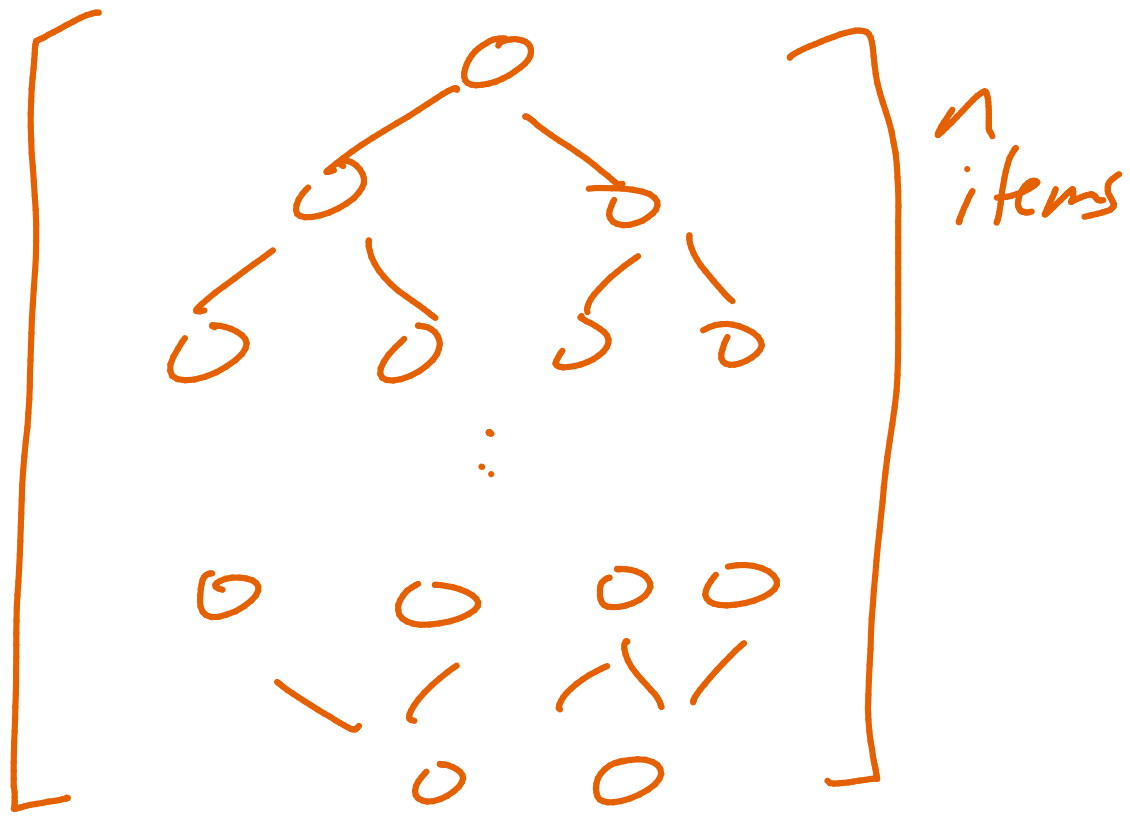
sum list

T_1 = time to run on one processor
= work = # of nodes $O(n)$

T_∞ = time to run on infinite processors
= span = length of longest path in graph $O(\log n)$

fork join sum

height of this tree $O(\log n)$



Speedup on P processors
T_1 / T_P

On infinite processors
Speedup = T_1 / T_∞ = parallelism

For sum fork join,

parallelism $= O\left(\frac{n}{\log n}\right)$

called <u>exponential speedup</u>

Conceptually, speedup $= \frac{n}{\log n}$

n is much bigger than $\log n$!

$2^{(\log_2 n)} = n$

n is exponentially bigger than $\log n$

Brent's Law / Thm - basic expectations

Amdahl's Law - depressing

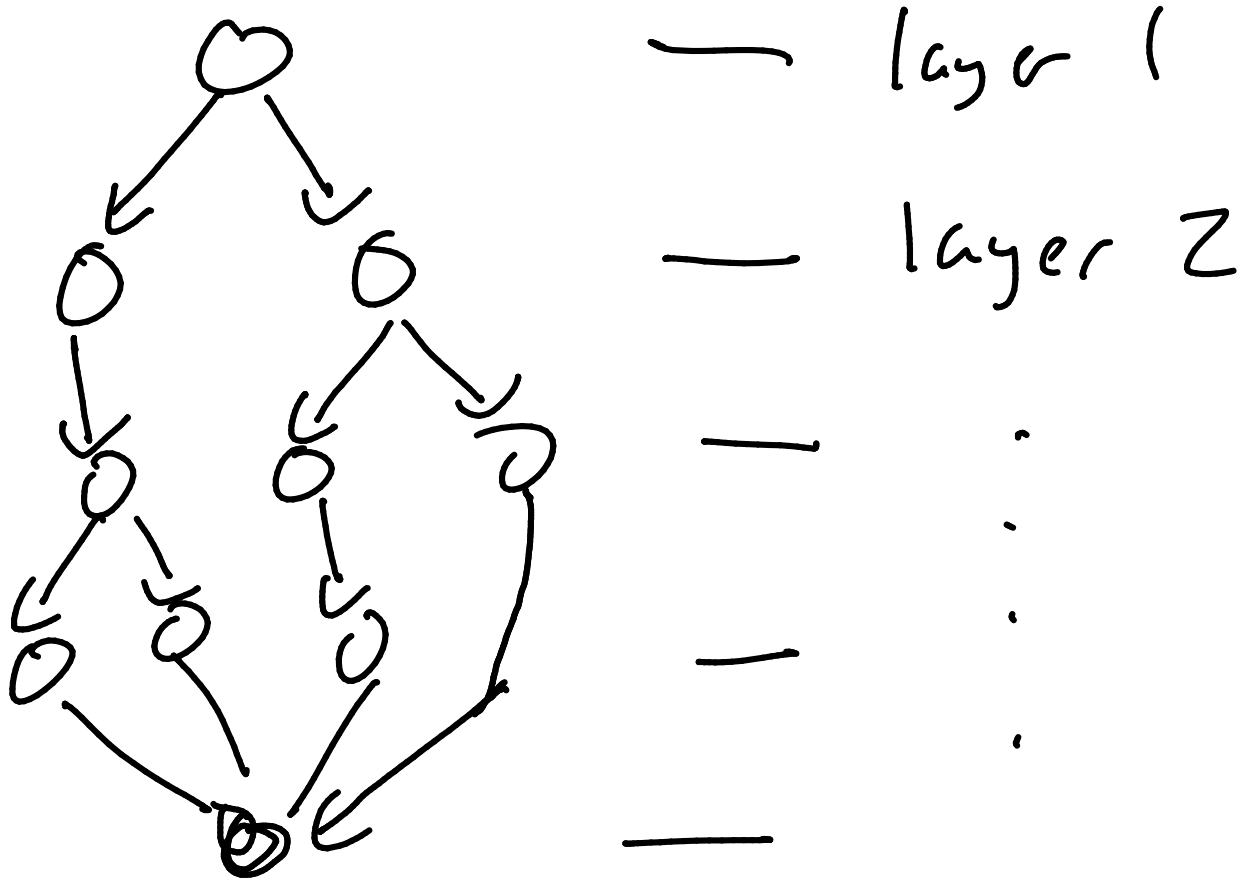
Gustafson's Law - nah, we can do it

Brent's Thm - goal was to figure out reasonable lower and upper bounds for possible speedup

Lower bound: if you take the work (T_1), and it evenly among P processors, that's as fast as you're going to get [w/ P processors]

$$T_P \geq \frac{T_1}{P}$$

Upper bound: Think of the DAG in layers.



How much time on P processors at each layer?

layer i

# of nodes at layer i

$$T_P^i = \left\lceil \frac{m_i}{P} \right\rceil$$

divide your m_i nodes evenly across P processors

$$\leq \frac{m_i}{P} + 1$$

So total time is

$$\sum_{i=1}^{\text{layers}} T_P^i \leq \sum_{i=1}^{\text{layer}} \left(\frac{m_i}{P} + 1 \right)$$

$$= \sum \frac{m_i}{P} + \sum 1$$

$$= \frac{1}{P} \sum m_i + \#\text{ of layers}$$

$$= \frac{\text{total work}}{P} + \#\text{ of layers}$$

$$= \frac{T_1}{P} + T_\infty$$

Put both bounds together
(one version of Brent's Thm)

$$\frac{T_1}{P} \leq T_P \leq \frac{T_1}{P} + T_\infty$$

never do better than even distribution

never do worse (assuming you divide evenly when you can) then dividing evenly plus the time to do the portion that must be done sequentially

1967, Gene Amdahl (IBM)

Look folks.

We've got an algorithm. T_1 = work

Break into two pieces
- piece that must be sequential
- piece that can be parallelized

S = fraction that can't be parallelized

$(0 \leq S \leq 1)$

$$T_1 = \underbrace{S T_1}_{\text{sequential}} + \underbrace{(1-S) T_1}_{\text{parallel}}$$

Let's assume with P processors we do the smart thing, and divide evenly

$$T_P = S T_1 + \frac{(1-S) T_1}{P}$$

Speedup

$$\frac{T_1}{T_P} = \frac{T_1}{S T_1 + \frac{(1-S) T_1}{P}} = \frac{1}{S + \frac{(1-S)}{P}}$$

Amdahl's Law

As $P \Rightarrow \infty$

$\frac{T_1}{T_\infty}$ = parallelism = $\frac{1}{S}$

Suppose that 1/4 of a program must be sequential.

$S = 1/4$

w/ infinite processors, speedup = 4

Suppose $S = \frac{1}{20} = .05$. 95% of program can be parallelized.

max parallelism is 20.

In 1988, Gustafson → Sandia Nat'l Labs

- with 1024 processors, speedups of around 1010–1020 with $S = 0.4$????

Gustafson says that Amdahl's assumption that S is fixed is wrong.

In his reality, not true.

- the more processors you have, the more work you choose to do

OR

- you've got a fixed amount of time, and you run what you can in that time.

He says

assume you intend to spend a fixed amount of time. Call that T

S = fraction that is sequential

$$T = ST + (1-S)T$$

what is speedup over P processors?

Defines speedup

$\frac{T_{1p}}{T}$ T_{1p} = time to run on one processor all the work I do if I have P processors"

$T_{1p} = ST + \underbrace{(1-S)T}_{\text{parallel work}} * P$ ← P to do all sequentially

$$\frac{T_{1p}}{T} = \frac{ST + (1-S)TP}{ST + (1-S)T}$$

$$= \frac{S + (1-S)P}{S + 1 - S} \qquad = S + (1-S)P$$

$$= S + P - SP$$

speedup $= P + (1-P)S$

Gustafson's Law

$$\text{Speedup} = P + (1 - P)\,S$$

Suppose $P = 1000$

$$\text{speedup} = 1000 - 999\,S$$

If $S = 0.1$

$$\text{speedup} = 1000 - 99.9$$

$$\approx 900$$