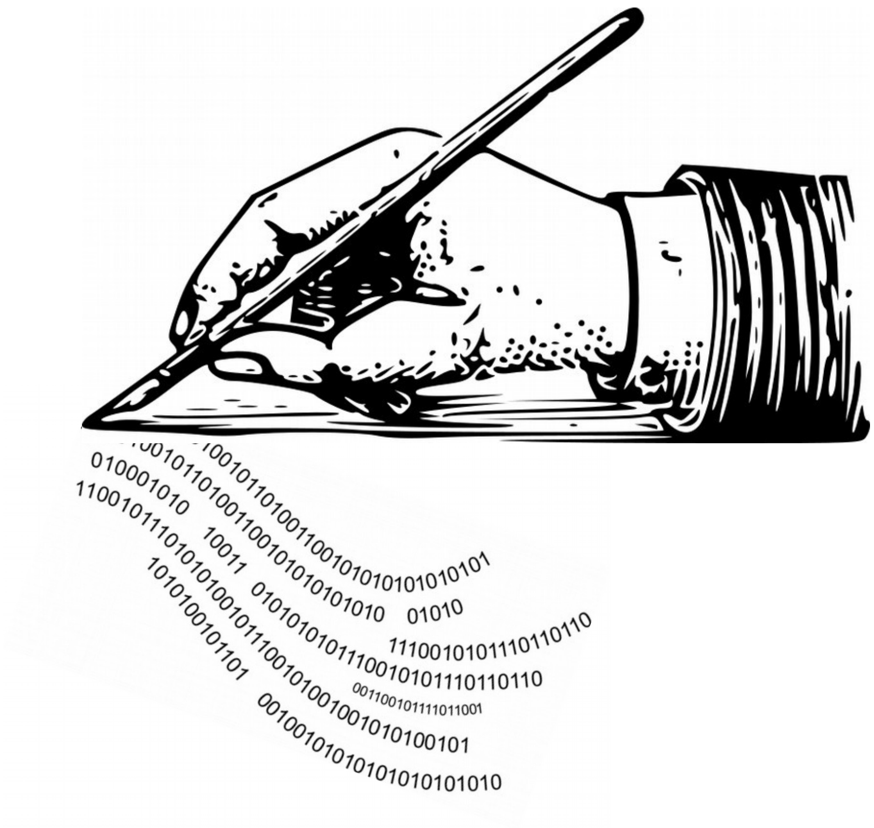
# CS 470
# Spring 2017

Mike Lam, Professor



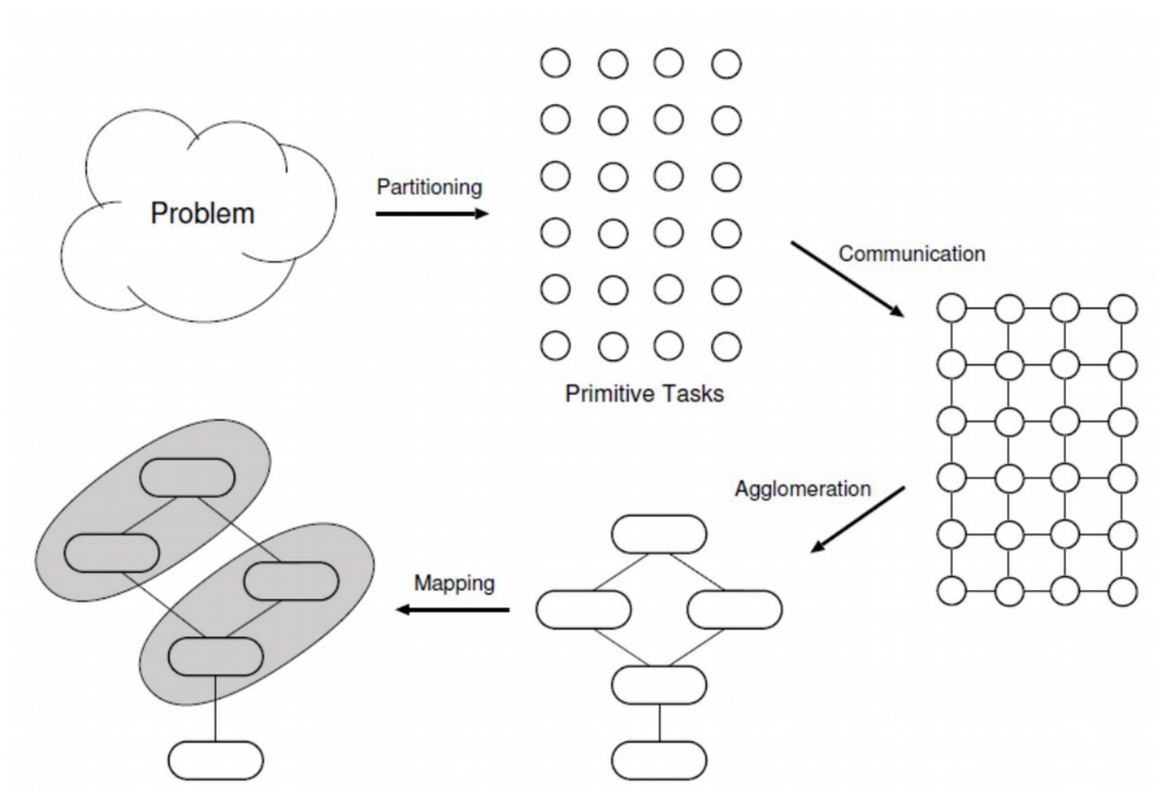
# Foster’s Methodology Examples

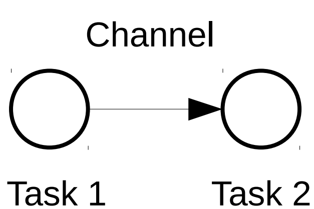
Graphics and content taken from IPP section 2.7 and the following:

http://www.mcs.anl.gov/~itf/dbpp/text/book.html
http://compsci.hunter.cuny.edu/~sweiss/course_materials/csci493.65/lecture_notes/chapter03.pdf
https://fenix.tecnico.ulisboa.pt/downloadFile/3779577334688/cpd-11.pdf

# Foster's methodology

- Task: executable unit along with local memory and I/O ports
- Channel: message queue connecting tasks' input and output ports
- Drawn as a graph, tasks are vertices and channels are edges
- Steps:
  1) Partitioning
  2) Communication
  3) Agglomeration
  4) Mapping




Foster's textbook is online:
`http://www.mcs.anl.gov/~itf/dbpp/text/book.html`

# Boundary Value Problem

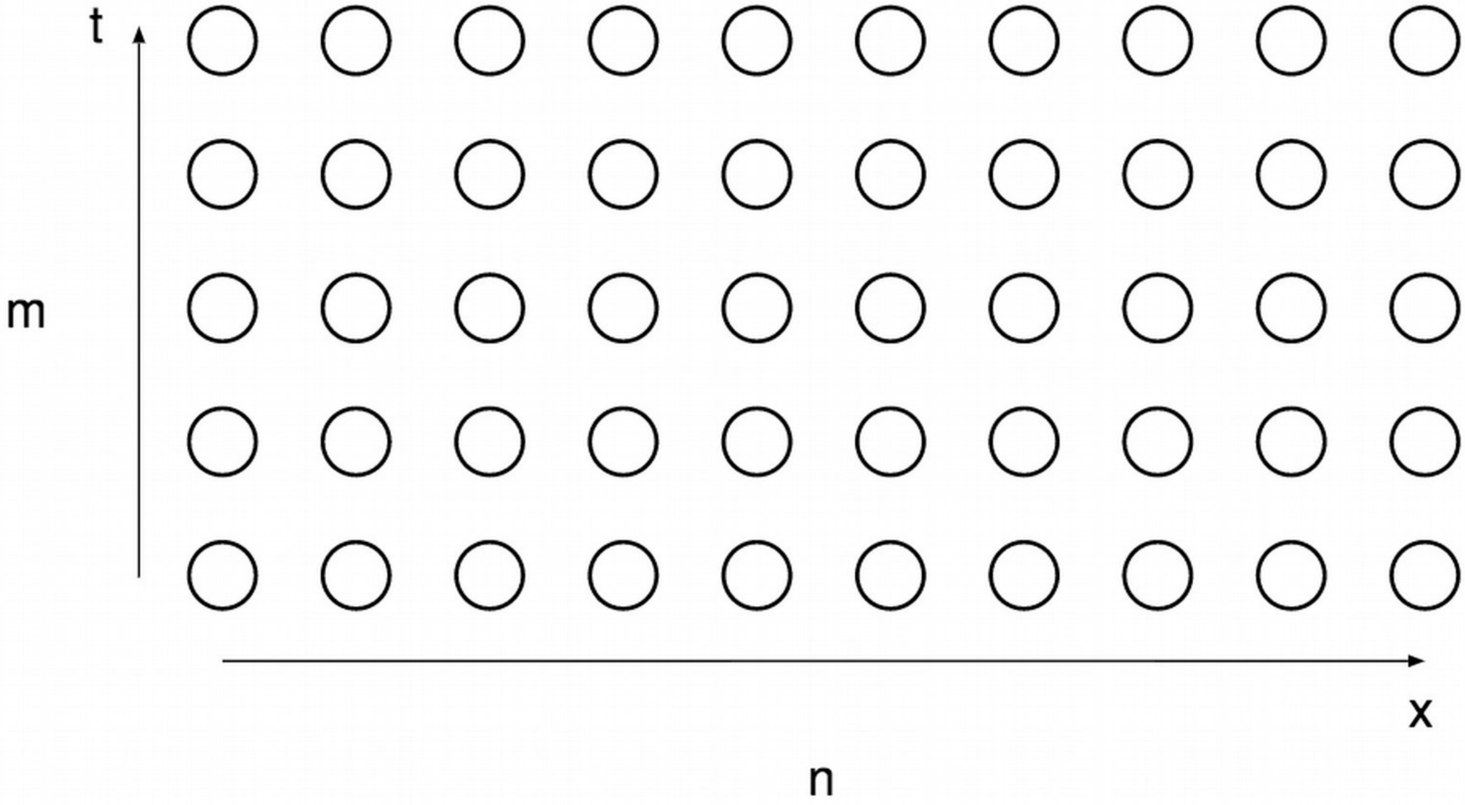
- Problem: Determine the temperature changes in a thin bar of uniform material with constant-temperature boundary caps over a given time period, given the length of the bar and its initial temperature
  - General solution: solve partial differential equation
    - Usually too expensive!
  - Approximate solution: finite difference method
    - Discretize space (1d grid) and time (ms)

- Goal: Parallelize this solution, using Foster's methodology as a guide

# Boundary Value Problem

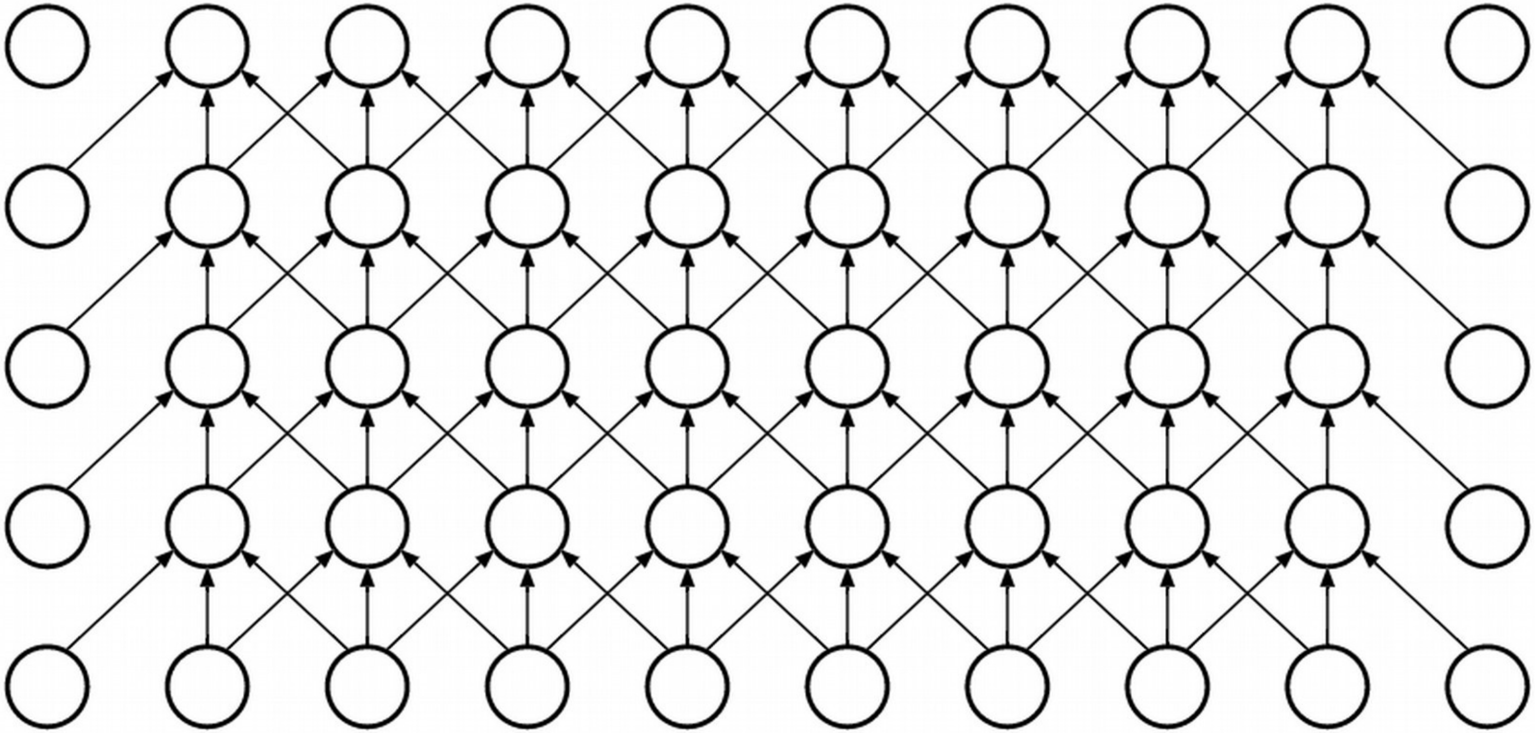
Partitioning:

Make each $T(x, t)$ computation a primitive task.
$\Rightarrow$ 2-dimensional domain decomposition



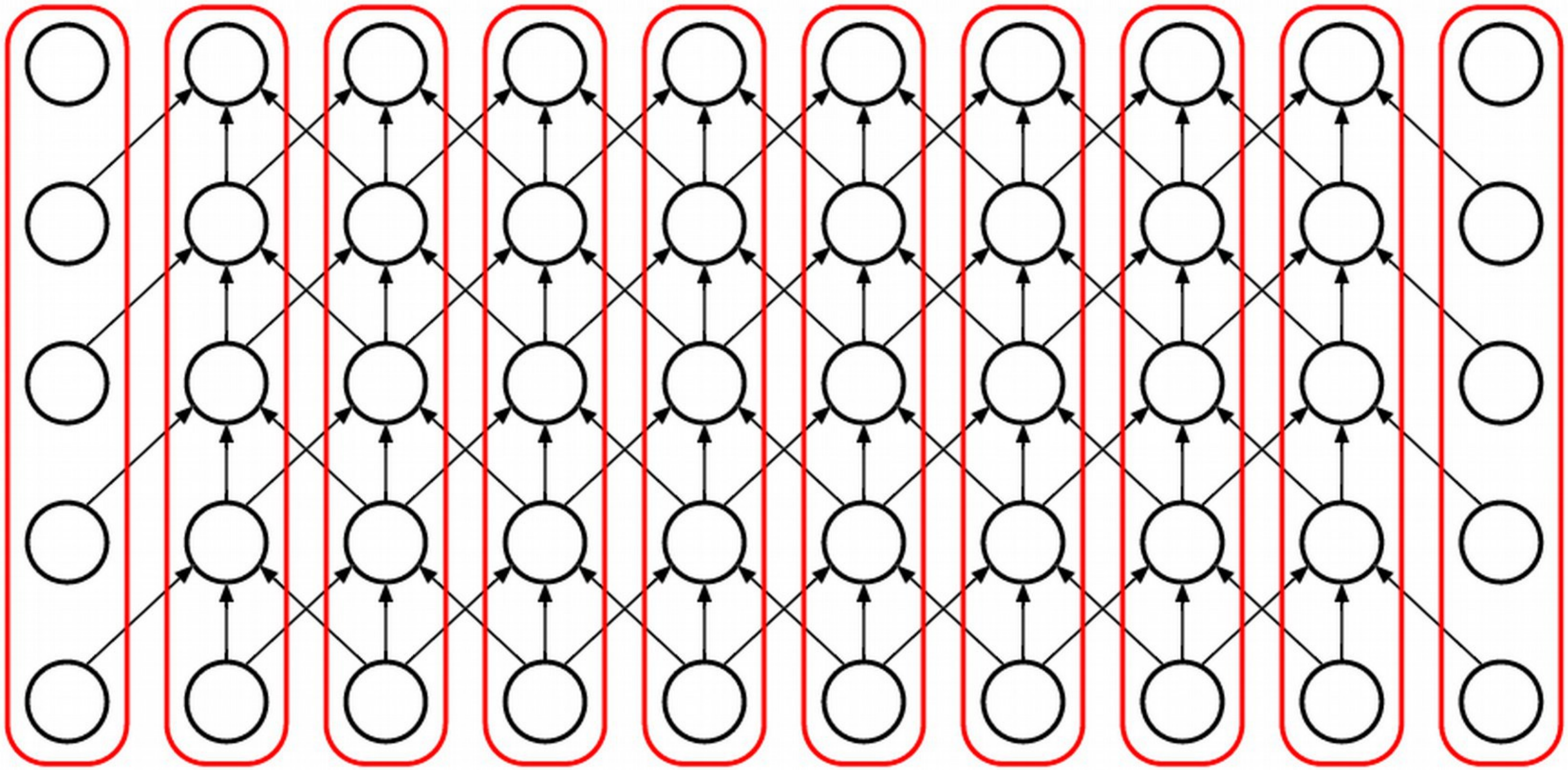
# Boundary Value Problem

Communication:



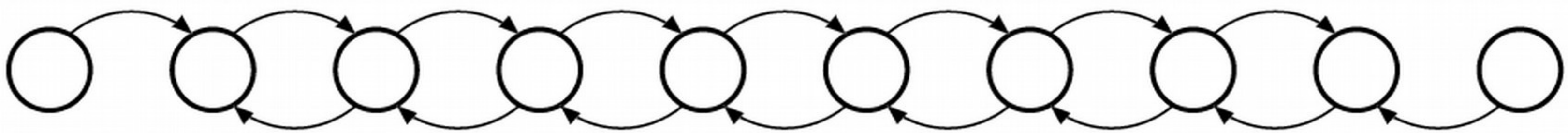
# Boundary Value Problem

Agglomeration:

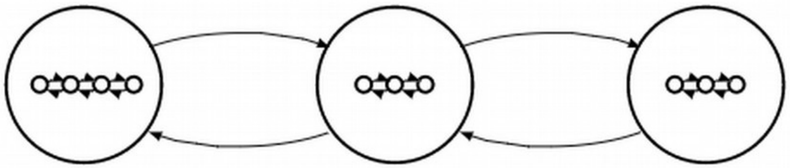


# Boundary Value Problem

Agglomeration:



Mapping:



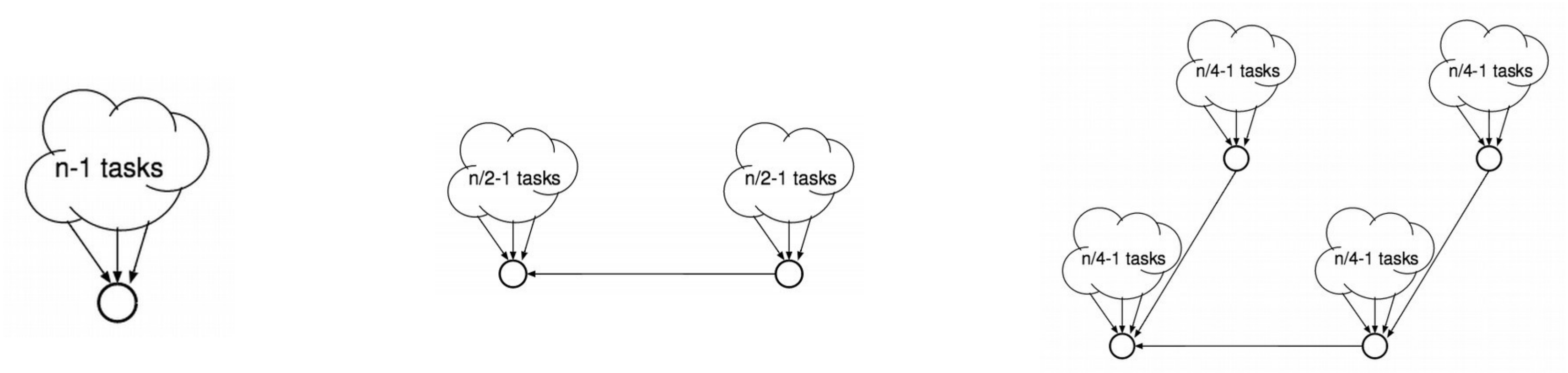
# Finding a maximum

- Problem: Determine the maximum value among some large set of given values
  - Special case of a reduction

- Goal: Parallelize this solution, using Foster's methodology as a guide

# Finding a maximum

- Partitioning: each value is a primitive task
  - (1d domain decomposition)
  - One task (root) will compute final solution
- Communication: divide-and-conquer
  - Root task needs to compute max after n-1 tasks
  - Keep splitting the input space in half

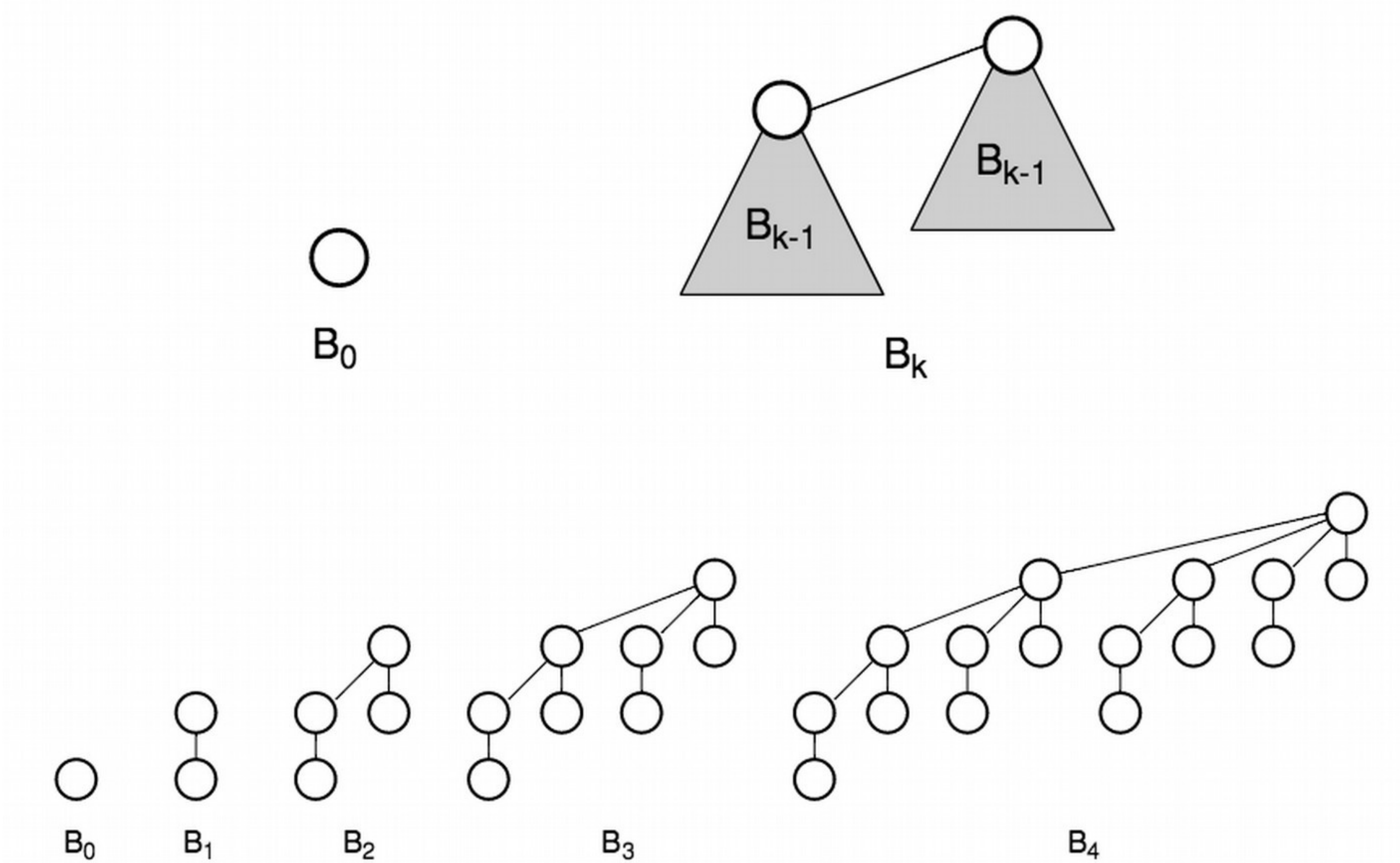


# Finding a maximum

- Binomial tree with n = $2^k$ nodes
  - (remember merge sort in P2?)

Recursive definition:

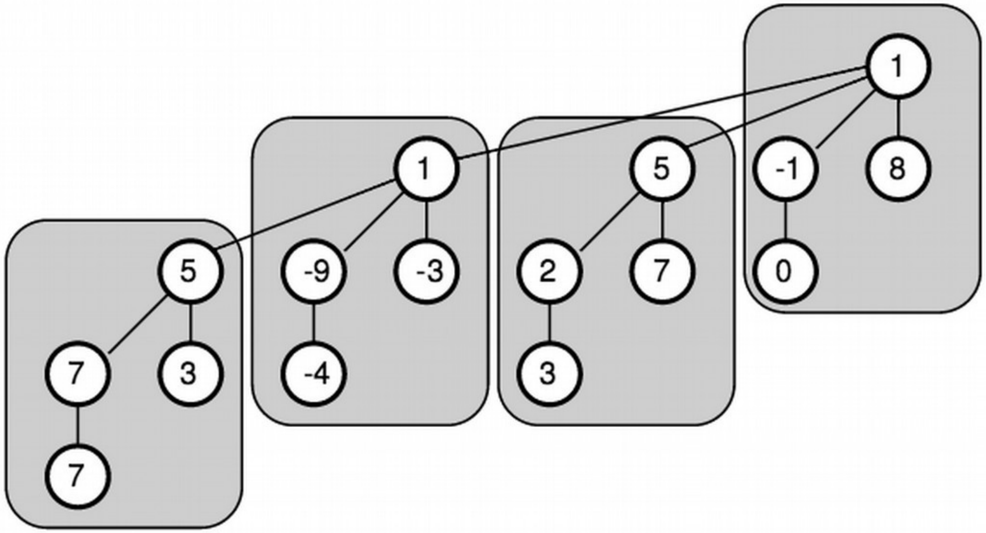
Examples:



# Finding a maximum

Agglomeration:

Group $n$ leafs of the tree:




Mapping:

The same (actually, in the agglomeration phase, use $n$ such that you end up with $p$ tasks).

# Random number generation

- Goal: Generate uniform psuedo-random numbers in a distributed way
- Problem: We wish to retain some notion of reproducibility
  - In other words: results should be deterministic, given the RNG seed
  - This means we can't depend on the ordering of distributed communications
- Problem: We wish to avoid duplicated series of generated numbers
  - This means we can't just use the same generator in all processes
- Naive solution:
  - Generate all numbers on one node and scatter them (a la P2)
  - Too slow!
- Can we do better?  (Foster's)
  - Generating each random number is a task
  - Channels between subsequent numbers from the same seed
  - Tweak communication & agglomeration
  - Minimize dependencies

# Random number generation

**Goal:**
Uniform randomness and reproducibility

$$L_{k+1} = a_L L_k \bmod m$$
$$R_{k+1} = a_R R_k \bmod m$$

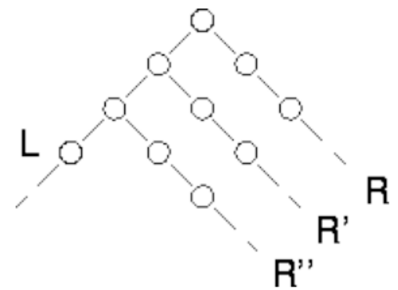



**Figure 10.1:** *The random tree method. Two generators are used to construct a tree of random numbers. The right generator is applied to elements of the sequence L generated by the left generator to generate new sequences R, R′, R″, etc.*

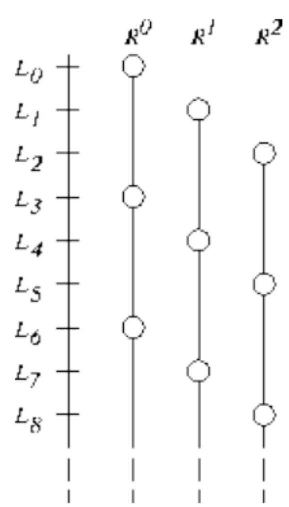



**Figure 10.2:** *The leapfrog method with n=3. Each of the three right generators selects a disjoint subsequence of the sequence constructed by the left generator's sequence.*

More info in Chapter 10 of
`http://www.mcs.anl.gov/~itf/dbpp/text/book.html`