"How to have a bad research career"

- 3 homework - 30%
- project - 40%
- midterm - 25%
- 5% participation

Why study parallel algorithms and d.s.?
- Moore's law ending "Dennard scaling"
- parallelism & concurrency fundamental ideas for how to do work (e.g., producing a complex object)
- superset of serial algorithms; understand new aspects of a problem by studying in parallel.
- Amount of data + analyses not slowing down
- genomic data
- video/image data

(1) Multicore machines exist!

- p cores, \( p \times \) faster
  \[
  \frac{t_1}{t_p}
  \]
  \( t_p \) best seq. running time

\[ O(n^2) \text { operations } \quad O(n \log n) \text { operations } \]

- Early apps were large-scale simulation

1961

IBM Stretch:

1MB memory

\[
\]

1972

Iliiac IV: 4 CPUs, 256 FPUs
512K memory 1 CPU 64 FPUs

\( \rightarrow \) SIMD (single-instruction, multiple-data)
1973: Eispach: matrix eigenvalues
        \[ \text{LAPACK (Jack Dongarra)} \]

1976: CRAY-1
        - vector processing capability
        \[ \begin{array}{c|c|c|c}
        \hline
        0 & 1 & \theta & 0 \\
        \hline
        \end{array} \]

1912: MSF algorithm (Böndke)

1968: Batcher Sort / Bitonic Sort
        \[ \begin{array}{c|c|c|c|c}
        \hline
        \emptyset & 1 & \emptyset & \emptyset \\
        \hline
        \end{array} \]
        \[ O(n \log^2 n) \text{ operations} \]
        \[ O(\log^2 n) \text{ delay (13 steps) rounds} \]

1975: Valiant: Max, Merging and Sorting
        \[ O(\log \log n) \text{ rounds} \]
        \[ \Omega(\log \log n) \text{ round} \]
        \[ \Omega(\log n) \text{ n processors} \]

1978: PRAM (Wylie)
1979: Circuit models (P-completeness, NC)

1980s: Golden age.

90-95: Other model (Asynch PRAM)
(BSP - Valiant)
↑
predecessor of MapReduce

1995

Parallelism Winter

2010

Resurgence - Lot of interesting work.

Present

RAM: Random Access Machine:
- how does an algorithm behave when data grows?
  - \( O(n^2) \) time vs \( O(n \log n) \) time.
  - +, x, ÷ unit time
  - load/store unit time - \( O(1) \) time
  - load: compare algorithms asymptotically

- \( O(\log n) \) bit words

Main Memory
Parallel Models

**PRAM**

Parallel Random Access Machine

- Synchronous
  - works in lockstep
- not particularly realistic
  - fixed # processors
  - schedule tasks to processors?

**Cost** = **Running Time** (p processors)

- Processors

Total # instructions = PT = work

Sorting in $O\left(\frac{n \log n}{p} + \log^2 n\right)$ time

PRAM models:
- **ER**: can't read same location at the same time
- **EW**: "write"
- **CR**
- **CW**

**ER**, **EW**, **CR**, **CW**
Vector Models

Connection Machine
Guy Blelloch's thesis

- instructions operate on vectors
  - merge, prefix-sum
vector
memory

Cost: \[ \text{Work (element complexity)} = \sum_{i} \text{length}(v_i) \]

Step complexity = # steps executed by the program

E.g., sorting in \( O(\log^2 n) \) steps
\( O(n \log n) \) work

"Connection Machine" Book

NESL

Circuit Model 1979 (Pippenger)
- view parallel comp. as a circuit (DAC)
  \( \wedge \) (and) and \( \vee \) (or) gates

Input
\[ \text{Cost: } \text{Size} = \# \text{ gates} \]
\[ \text{Size, } O(\log n) \text{ depth} \]

\[ \text{Sorting: } (\text{AKS network}) \]
\[ O(n \log n) \text{ size, } O(\log n) \text{ depth} \]

\[ \text{CRCW PRAM } O(\log n) \text{ depth} \rightarrow \text{AC circuits in same depth} \]
\[ \text{EREW PRAM } O(\log^2 n) \text{ depth} \rightarrow \text{NC circuits} \]

\[ \text{NC}^k : \text{Circuits with } O(\log^k n) \text{ depth} \]

\[ \text{NC} = \bigcup_k \text{NC}^k \]