Technical overview:

Themes:
- models of computation: matter!
- fancy data structures: cool!
- tight lower bounds: hard!

Temporal DS: manipulate time (time travel)
- persistence: fixed past
  - motivation: undo, geometry (time-space)
- partial: linear time, query past
- full: branching time \(\Rightarrow\) tree
  - generally possible with \(O(1)\) overhead
  - confluent: can merge timelines \(\Rightarrow\) DAG
  - lots of results & open problems
    - e.g. confluent files & directories solved in 851!
- retroactivity: change the past
  - motivation: mistake correction, geometry
  - hard in general
Geometric DS: points in $d > 1$ dimensions
- motivation: relational databases
- can preprocess $n$ points in 3D to find all points in query box in $O(lg n)$ time
- kinetic DS: moving points

Dynamic optimality: is there one best BST?
- $O(1)$-competitive against any BST?
- any balanced BST is $O(lg n)$-competit.
- Tango Trees are $O(lg lg n)$-competitive
- conjecture: Greedy is $O(1)$-competitive

Memory hierarchy:
- when you load 1 word of data, get $B$ for same cost
- goal: amortize high cost over $B$ items
- scanning $N$ items costs $\Theta(N/B)$
- sorting $N$ items costs $\Theta(N/Blg B)$
  e.g., $\frac{m}{B}$-way mergesort
& priority queue in $\Theta(\frac{1}{B} lg_m B \frac{N}{B}) < 1$!
- can do all this without ever knowing $B$ & $M$! “cache oblivious”
  $\Rightarrow$ works well on multilevel hierarchy too
**Integer DS:** words store ints \( c \in \{0, 1, \ldots, u-1\} \)

\( w \) bits \( \Rightarrow u = 2^w \)

- hashing is one example:
  - \( O(1) \) time w.h.p. insert/delete/search
  - insert/delete/predecessor/successor (like BSTs): for \( O(n \ poly\lg n) \) space,
    \[ \Theta \left( \min \left\{ \frac{\log w}{\log \log w}, \frac{\log w}{\log \log n} \right\} \right) \leq \Theta \left( \frac{\sqrt{\log w}}{\sqrt{\log n}} \right) \]

- sorting in \( O(n) \) time / \( O(1) \) priority queue
  - \( \frac{w = O(\log n)}{} \)
  - \( w = \Omega (\log^{2+\epsilon} n) \)

  \[ \text{radix sort} \]

**String DS:** preprocess text \( T \) to search for substring \( P \) in \( O(|P|) \) \(-\text{indep. of} \ T! \)

- find longest common prefix of 2
  - (preprocessed) strings in \( O(1) \) time

**Succinct DS:** above in \( O(|T|) \) bits, not words

- store \( n \) parentheses in \( n + o(n) \) bits
  - & find matching/parent paren(s) in \( O(1) \) time

**Dynamic graphs:** insert/delete edges

- \( O(\lg n) \) for trees (\( \approx \) solved in 85!)
- \( O(\lg n \cdot (\lg \lg n)^3) \) for undirected graphs
- we’ll see \( O(\lg^2 n) \)
Class format:
- video lectures + handwritten & scribe notes (from 2018)
  - DEMO: note sync & jump, playback speed
  - completion & scribe feedback form DEMO
  - DUE BY NOON ON TUESDAY
- questions on Coauthor DEMO
- class (M7-9:30) for every 2 lectures (REQUIRED ATTENDANCE)
- Q&A / additional material (based on feedback)
- group puzzle solving "SOLVED"
  - learn material, collaboration skills, warmup for:
- attack open problems "OPEN"
  - build research skills, thrill of unknown, fun & challenge of advancing frontiers of research
  - implement data structures "CODING"
- Coauthor software to coordinate in/outside class MUST POST/BE @MENTIONED EACH WEEK DEMO STATS
- weekly psets: 1 page in, 1 page out USUALLY DUE WEDNESDAYS AT NOON DEMO PS1
- scribe notes: revise/improve old notes (DUE TUESDAY NIGHT) including students' feedback (FOLLOWING CLASS)
- project & presentation e.g. from class
  - pose and/or try to solve open problem
  - implement & experiment with DS
- Survey a subfield (not well-covered in lectures)
- Wikipedia (write/improve several articles)