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 $\Rightarrow$ 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 \( d \geq 1 \) dimensions
- motivation: relational databases
- can preprocess \( n \) points in 3D to find all points in query box in \( O(l \log n) \) time
- kinetic DS: moving points

Dynamic optimality: is there one best BST?
- \( O(1) \)-competitive against any BST?
- any balanced BST is \( O(l \log n) \)-competet.
- Tango Trees are \( O(l \log \log n) \)-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 \( \Theta(l \sqrt{N/B}) \)
- sorting \( N \) items costs \( \Theta(N/B \log y/B) \)
  e.g., \( \frac{N}{B} \)-way merge sort & priority queue in \( \Theta(\frac{1}{B} \log y/B \frac{N}{B}) < 1! \)
- can do all this without even knowing \( B \) & \( M \)!
  "cache oblivious"
  \( \Rightarrow \) works well on multilevel hierarchy too
Integer DS: words store ints $c \in \{0, 1, \ldots, u-1\}$

$\rightarrow w$ 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 $O(n \text{ polylog } n)$ space,
  - $\Theta(\min \{\log w n, \frac{\log w}{\log \log n}\}) \leq O(\sqrt{\log n})$
- sorting in $O(n)$ time / $O(1)$ priority queue
  - for $w = O(\log n) \& w = \Omega(\log^{a+3} n)$
    - radix sort

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

- 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 parentheses in $O(1)$ time

Dynamic graphs: insert/delete edges & query: are $v$ & $w$ connected via path?
- $O(\log n)$ for trees (solved in 85!)
- $O(\log n \cdot (\log \log n)^3)$ for undirected graphs
- we'll see $O(\log^2 n)$
Class format:
- video lectures from 2012
- completion & feedback form
  \textit{DUE TUESDAYS AT NOON}
- Piazza for raising questions
- class (W2:30-5) for every 2 lectures
- Q&A (based on form/Piazza feedback)
- group puzzle solving
  - build collaboration skills, warmup for:
  - attack open problems
    - build research skills, thrill of unknown, fun & challenge of advancing frontiers of research
- weekly psets: 1 page in, 1 page out
  \textit{USUALLY DUE MONDAYS AT NOON}
- final project: written & presented
  - pose and/or try to solve open problem
    (e.g. from open problem session)
  - implement & experiment with DS
  - survey a few papers (not well-covered)
  - improve Wikipedia

\textbf{Proposal DUE APRIL 9, 2014}

Problems: groups of 5-10 people
- don't worry about solving ~ about journey
- solved problems: write progress in Piazza
  - private note ~ we'll post answers for record
- when done/bored, move on to open problem
Problem 1: insert/delete/successor/pred. in $O(\log n)$
+ insert-after/before in $O(1)$ amortized
  $\implies$ given node $x$, e.g. found by pred./succ.
  insert $y = x \pm \varepsilon$

[+ delete-here in $O(1)$ amortized]
  $\implies$ given node $x$, delete it

interesting with or without this

Problem 2: insert/delete/pred./successor in $O(\log n)$
[+ split in $O(\log n)$]
  $\implies$ given DS & key $x$, split DS into
  DS of all items $\leq x$ & DS of all items $> x$

[+ concatenate in $O(\log n)$]
  $\implies$ given 2 DSs A & B such that
  $a < b$ for all $a \in A$ & $b \in B$,
  combine into 1 DS of all items in $A \cup B$

either op. is interesting

Open problem: given $n$ points $P$ in 2D, no 2 on common row [or column],
find minimum point set $Q \subseteq P$ such that:
for any 2 points $\in Q$ not on common row/col.,
the rectangle they span contains another point $\in Q$

- NP-hard?
- $O(1)$-approximation?