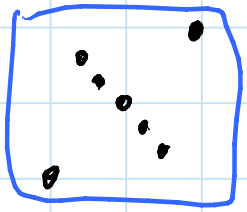
6.851	Students I	May 6, 2010

Peter Caday & Rishi Gupta: THEORY

dynamic optimality in BSTs

- cf. Lectures 1 & 2
- alternating Greedy: (from Demaine et al.)
  apply ◪-Greedy, ◨-Greedy, repeat
- Conjecture: never more than 6 iterations!
- simulated annealing algorithm to find bad examples easily finds 6, but not 7
  - troubled by lots of local minima...
- Conjecture: # pts. in step 3
  $< \frac{1}{3}$(# pts. in steps 1 & 2)
  $\Rightarrow$ geometric
- either would imply alternating greedy is $O(1)$-competitive

worst case



- Q: is Greedy = O(unified)?
  - true, with constant 2, up to $n=10$ & $n=11$ permutations

- Q: is unified = O(Signed Greedy)?
  - worst cases computed up to $n=10$, $n=11$ permutations ~ maybe a pattern
  - constant ≈ 1.3

Cai GoGwilt & Matthew Hwang:

THEORY

density DSs for Integrated Circuit design

- "dummy fill" common in an IC layer to make density uniform
- problem: given point set & rectangle (size) 

  query: max. weight offset of rectangle
  update: insert point (no deletions)

- 1D: $O(1)$ query, $O(n^{\varepsilon})$? insert
  - more interesting than expected
- 2D: $O(1)$ query, $O(n^{1+\varepsilon})$ insert

- OPEN: objects are rectangles instead of pts
- OPEN: weighted points
  update = change weight of point
- in general VLSI seems a rich area for data structural problems

Reid Kleckner: EXPERIMENT

hopscotch hashing in Python?

- hopscotch hashing [Herlihy, Shavit, Tzafir 2008]
  - good caching behavior
  - behave well under high load
  - can be made concurrent (maybe avoid GIL)
- Python uses dictionaries everywhere
- Python dictionaries [Tim Peters]
  use quadratic probing with low load factor
- hopscotch hashing:
  - linear probing with constant cutoff H
  - insertion skips around to move elements out of the way...
  - analysis unclear
  - not dealing with clustering? (parking lot)
  - good performance results
- new results:
  - great for random ints. (like in paper)
  - 0.5% improvement for repeated lookups
  - more for unique keys
  - bad for Python's hash() ~ tends to cluster (especially if user gives crappy __hash__)
  - even with multiplication method on top?!

Jacob Steinhart & Ye Wang:

THEORY

parallel functional DSs

- idea: functional ⇒ easier to do concurrent access
- want results to correspond to some linearization of ops.
- k = # ops/unit time ≈ parallelism
- priority queues:
  - binary heap with path copying
  - $O(\lg n)$ memory/update
  - modification to support parallel inserts with some delay ($O(\lg k)$ merging...) before they take effect
  - maybe parallel deletes too
- BSTs:
  - do updates bottom up
  - still in progress

Andrew Winslow: kinetic 3D hull

THEORY

(cf. Lecture 5 on kinetic DSs)

- adapt [Bosch 1997] optimal kinetic 2D hull to 3D ~ nontrivial but seems possible
- really kinetic upper/lower envelope
- sketch: binary divide & conquer
  at each node, merge & make certificates
- challenges: (3D vs. 2D)
  - vertices & faces can have high degree
  - hull isn't ordered
- "nuke them with "general position" on steroids"
  (for now anyway ~ future work to avoid this)
  - $O(1)$ size faces! & deg-3 vertices
- some new certificates, some adapted
- same performance as 2D
  but severe restriction on point sets