6.851	Lecture 21	May 8, 2012

<u>TODAY</u>: Dynamic graphs III (of 3)

- dynamic connectivity lower bound:
  - block operations
  - bit-reversal bad access sequence
  - tree over time
  - sum lower bound
  - connectivity lower bound

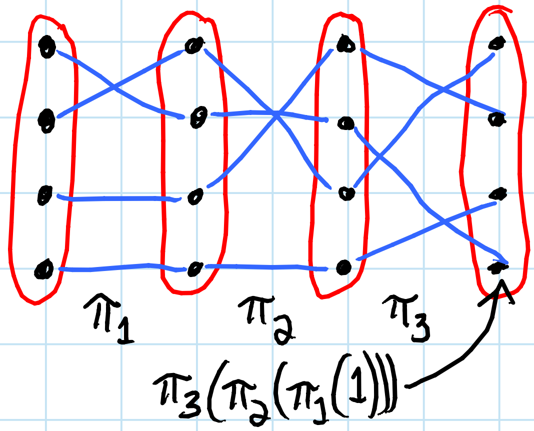
Dynamic connectivity lower bound:

[Pătraşcu & Demaine – SICOMP 2006]

inserting/deleting edges & connectivity queries require $\Omega(\lg n)$ cell probes/op.
even if connected components are paths
even amortized (but here prove for worst case)
$\Rightarrow$ link-cut & Euler-tour trees are optimal

Proof:

- consider $\sqrt{n} \times \sqrt{n}$ grid with perfect matching between columns i & $i+1$ for each i, forming permutation π_i



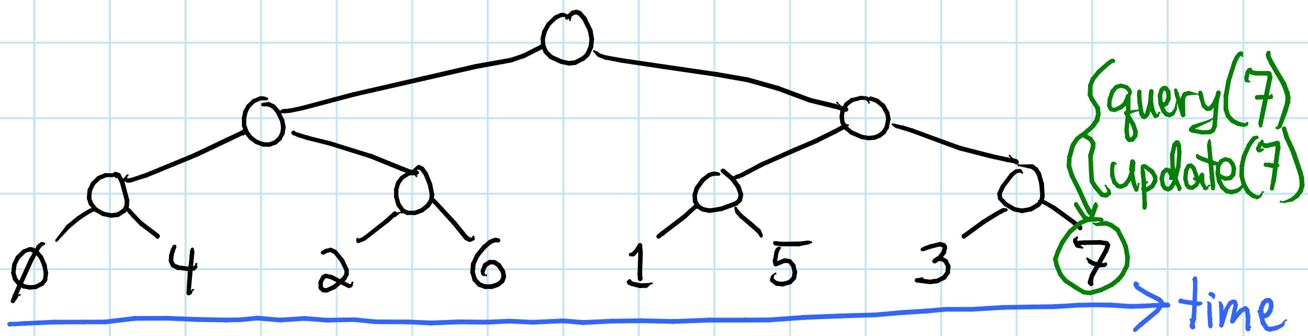
- block operations:
  - update(i, π): $\pi_i \leftarrow \pi$
    $= O(\sqrt{n})$ edge deletions & insertions
  - verify-sum(i, π): $\sum_{j=1}^{i} \pi_j = \pi$?
    compose
    $= O(\sqrt{n})$ connectivity queries
- Claim: $\sqrt{n}$ updates + $\sqrt{n}$ verify sums require $\Omega(\sqrt{n} \cdot \sqrt{n} \cdot \lg n)$ cell probes
  $\Rightarrow \Omega(\lg n)/\text{op}$.

Bad access sequence:

- for i in bit-reversal sequence:

  000 → 000 → 0
  001 → 100 → 4
  010 → 010 → 2
  011 → 110 → 6
  ⋮ ⋮ ⋮

  - verify-sum$(i, \sum_{j=1}^{i} \pi_j)$ ⇒ answer = yes (but DS must check)
  - update$(i, \underbrace{\pi_{\text{random}}}_{\text{uniform random permutation}})$
- build tree over time:



  - left & right subtrees of each node interleave

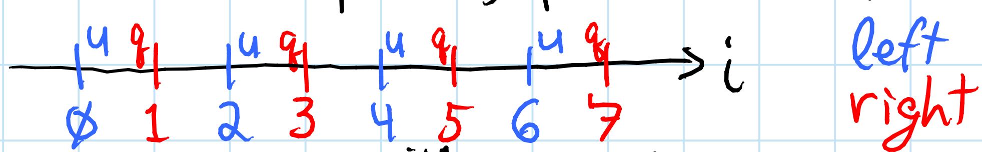
- Claim: for every node v in tree,
  say with ℓ leaves in its subtree,
  during right subtree of v (time interval)
  must do $\Omega(\ell\sqrt{n})$ expected cell probes
  reading cells last written during left subtree

- sum lower bound over all nodes:
  - read r of write w only counted at $lca(r, w)$
  - linearity of expectation

⇒ $\Omega(n \lg n)$ lower bound total
(each leaf in $\Theta(\lg n)$ subtrees)

Proof of claim:

- left subtree has $\ell/2$ updates with $\ell/2$ rand. perms.
- any encoding of these permutations must use $\Omega(\ell\sqrt{n}\lg n)$ bits [information/Kolmogorov theory]
- if claim fails, find smaller encoding $\Rightarrow$ contradict.
- setup: know the past (before v's subtree)
- goal: encode (verified) sums in right subtree
  $\Rightarrow$ can recover (updated) perms. in left subtree



$$\pi_i = \underbrace{\pi_{i-1}^{-1}\circ\cdots\circ\pi_1^{-1}}_{\text{farther left}\Rightarrow\text{known}} \circ \underbrace{\sum_{j=1}^{i+1}\pi_j}_{\text{right } q} \circ \underbrace{\pi_{i+1}^{-1}}_{\text{not yet updated}}$$

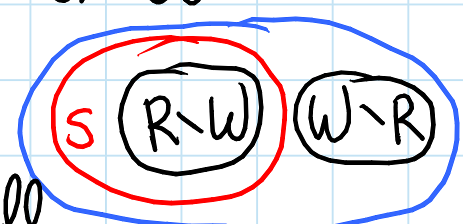
Warmup: query is sum(i) $\rightarrow \sum_{j=1}^{i}\pi_j$ (partial sums)

- let $R = \{$cells read during right subtree$\}$
  $W = \{$cells written during left subtree$\}$
- encode $R\cap W$ (address & contents of each cell)
  $\Rightarrow |R\cap W|\cdot O(\lg n)$ bits [assume poly. space $\Rightarrow w = \Theta(\lg n)$]
- decoding alg. for sums in right subtree:
  - simulate sum queries in right subtree
  - to read cell written in right subtree: easy
    in left subtree: $R\cap W$
    in past: known

$\Rightarrow |R\cap W|\cdot\cancel{O(\lg n)} = \Omega(\ell\sqrt{n}\,\cancel{\lg n})$
$\Rightarrow |R\cap W| = \Omega(\ell\sqrt{n})$ ✓

Verify-sum instead of sum:

- permutations π given to verify-sum encode the information we want ($\Rightarrow$ no info. LB)
- setup:
  - know (fixed) past
  - don't know updates in left subtree
  - don't know queries in right subtree
  - but know that queries return YES
- decoding idea:
  - simulate all possible input permutations for each query in right subtree
  - know one returns YES, all others NO
- trouble: incorrect query simulation reads cells $R' \neq R$
  - if read $r \in R' \setminus R$, it must be incorrect
  - but can't tell whether $r \in W \setminus R$ or past $\setminus (R \cap W)$
  - can't afford to encode R or W
- idea: encode separator S for $R \setminus W$ & $W \setminus R$



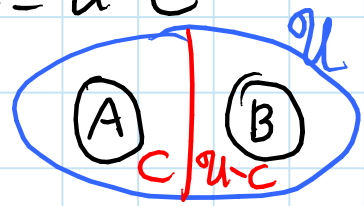
- when decoding, to read cell written in right subtree: easy
  in $R \cap W$: encoded explicitly
  in S: must be in past $\Rightarrow$ known
  not in S: must not be in $R \Rightarrow$ incorrect; ABORT
- only one simulation returns YES; rest NO or ABORT

$\Rightarrow$ recover desired permutation

$\Rightarrow |\text{encoding}| = \Omega(\ell \sqrt{n} \lg n)$

Separators:

- given universe $\mathcal{U}$ & number m
- separator family $\mathcal{S}$ for size-m sets if $\forall A, B \subseteq \mathcal{U}$ with $|A|, |B| \le m$ & $A \cap B = \emptyset$:
  $\exists C \in \mathcal{S}$ such that $A \subseteq C$ & $B \subseteq \mathcal{U} \setminus C$
- claim: $\exists$ separator family $\mathcal{S}$ with $|\mathcal{S}| \le 2^{O(m + \lg\lg \mathcal{U})}$



- proof sketch:
  - perfect hash family $\mathcal{H}$ with $|\mathcal{H}| \le 2^{O(m + \lg\lg \mathcal{U})}$
    [Hagerup & Tholey – STACS 2001]
    gives mapping from A & B to $O(m)$-size table
  - store A-or-B bit in each table entry
  - $2^{O(m)}$ such vectors
  $\Rightarrow 2^{O(m)} \cdot 2^{O(m + \lg\lg \mathcal{U})} = 2^{O(m + \lg\lg \mathcal{U})}$

Encoding: $R \cap W$ + separator of $R \setminus W$ & $W \setminus R$

- size: $|R \cap W| \cdot O(\lg n) + O(|R| + |W| + \lg\lg n)$
  $= \Omega(\ell \sqrt{n} \lg n)$

$\Rightarrow |R \cap W| = \Omega(\ell \sqrt{n})$ $\Rightarrow$ claim

or $|R| + |W| = \Omega(\ell \sqrt{n} \lg n)$ $\Rightarrow \Omega(\lg n)$ for op.

$+ \lg\lg n$

$\square$

Update-query trade-off: (possible by same technique)

$$t_q \lg \frac{t_u}{t_q} = \Omega(\lg n) \quad \& \quad t_u \lg \frac{t_q}{t_u} = \Omega(\lg n)$$

- for $t_u = \Omega(t_q)$, trees can match
  (small mods. to link-cut trees)
- for $t_u = \Omega(\lg n\, (\lg \lg n)^3)$, can match
  [Thorup-STOC 2000]