$\left\{ \text{Csc 80030} \;\middle|\; \text{Lecture 9} \right\}$

Probabilistic Analysis & Randomized Algorithms

Hoeteck Wee · hoeteck@cs.qc.edu

PART 0

- HOMEWORK: HW4 due today, HW5 due Apr 29
- PRESENTATIONS: May 6, 13 – 40-min student presentations
- TODAY: sublinear time algorithms

PART 1 | sublinear time algorithms

Massive data sets

- EXAMPLES: genome project, sales logs, world-wide web, network traffic, clickstream patterns, health data, large remote database, ...
  - hardly fit in storage
- QUESTION: are traditional notions of an efficient algorithm sufficient?
  - is linear time good enough?
  - example: election vs polls
  - trade-off between efficiency and correctness

Additional examples

- Outbreak of diseases
  - how rapidly? wide-spread? correlated with income level / zip code?
  - speed is paramount
- Is the lottery uniform?
  - approximate answer good enough
  - can work with fewer samples than statistical χ^2-test
- Pattern matching in genome data
  - want quick answers, before further refinement
  - desire trade-off between running time and accuracy

Monotonicity testing

- PROBLEM. Given a list L of n distinct integers, check if the list is monotone (i.e. sorted).
  - can append index of an item to enforce distinctness
  - model: comparisons and random access take $O(1)$ time
  - need $\Omega(n)$ time for exact answer

Monotonicity testing

- PROBLEM. Given a list L of n distinct integers, check if the list is monotone (i.e. sorted).
- NOTION OF APPROXIMATION. L is ϵ-close to monotone if changing/deleting ϵn entries makes it monotone.
- PROPERTY TESTER. randomized algorithm that on input L, ϵ,
  - If L is monotone, accept w.p. $\geq 2/3$
  - If L is ϵ-far from monotone, reject w.p. $\geq 2/3$
  - "in-between" cases, don't care.
- NOTE. can improve $2/3$ to any $1 - \delta$ by taking majority of $O(\log 1/\delta)$ repetitions.

Warm-up

- NAIVE ALGORITHM: Pick a random subset S of indices and check that L restricted to S is monotone.
  - consider $L = (2, 1, 4, 3, 6, 5, \ldots)$
  - requires $\Omega(\sqrt{n})$ samples
- Can do do better! [Ergün et. al, STOC 98, JCSS 2000]

Monotonicity testing

monotonicity testing

Input: a list $L = (x_1, \ldots, x_n)$ of distinct integers

1. Repeat $2/\epsilon$ times:
   1.1 Pick a random index i from $[n]$.
   1.2 Perform a binary search for x_i.
   1.3 Reject if we fail to find x_i or if we find out-of-order elements.
2. Accept if all the binary searches succeed.

- running time is $O(\frac{1}{\epsilon} \log n)$.
- always accept if L is monotone.

Analysis

- KEY LEMMA. Let G denote the set of indices for which binary search succeeds. Then, the elements of L restricted to G is monotone.
  - If $|G| < (1-\epsilon)n$, then algorithm rejects with prob. $1-(1-\epsilon)^{2/\epsilon} \geq 1-\mathrm{e}^{-2} \geq 2/3$.
  - If $|G| \geq (1-\epsilon)n$, then L is ϵ-close to monotone.
- PROOF. for all $i, j \in G$, look at least common ancestor index where the binary searches for x_i and x_j diverge.

PART 2 | student presentations

Student presentations

- 40-min presentations on different papers
  - mandatory attendance
  - May 6 and 13, 4 pm - 6.15 pm
- Topics
  - Load balancing: Talwar, Wieder [STOC 2007]; Godfrey [SODA 2008]
  - Sublinear time algorithms: Chazelle, Rubinfeld, Trevisan [ICALP 2001]; Bogdanov, Obata, Trevisan [FOCS 2002]
  - Spectral algorithms: Trevisan [STOC 2009]

THE END | next, spring break!