CS 420: Advanced Algorithm Design and Analysis
Spring 2015 – Lecture 6

Department of Computer Science
University of British Columbia

January 22, 2015

Announcements

Assignments...

- Asst1...back today (with some discussion)
- Asst2... due today
- Asst3 out....(due next Thursday)

Upcoming Exams / Q/A Sessions ...

- review session: Tuesday, Feb. 03, 5:30-7:00; room TBA
- exam: Wednesday, Feb. 04, 5:30-7:00; room TBA
  - covers material up to Lecture 8 (January 29)

Readings...

- material on hashing [Kleinberg, 13.6; Cormen+, chap 11; Erickson, chapt 12]
- material on closest-pair problem [Kleinberg]

Looking ahead...

Our goal, in the next few lectures is to understand how we might circumvent this lower bound, by *stepping outside the abstract comparison-based model*. We will consider:

- exploiting assumptions about the structure/size of the key space $\mathcal{U}$
- exploiting assumptions about the distribution of keys in S
- exploiting assumptions about the pattern of successive queries
- (if time permits) other issues: randomization, error tolerance...

Last class...

inputs are drawn from a restricted *universe* $\mathcal{U} = \{0, 1, \ldots u - 1\}$ (cont.)

- overcoming space concerns with previous structures
  - hashing (the role of randomization)
  - universal hashing
    - properties
    - application...perfect hashing

Today...

Compact universal families $\mathcal{H}$ exist and are efficient to construct

- Kleinberg&Tardos (section 13.6) describe one construction based on modular arithmetic

Applications of universal hashing (cont.)

- finding the closest pair of points in a point set: Kleinberg&Tardos (section 13.7)

Compact universal families $\mathcal{H}$ exist and are efficient to construct

Kleinberg&Tardos (section 13.6) describe one construction based on modular arithmetic

Assumptions:

- table T has size $m > |S|$, where m is a *prime number*
- keys are drawn from universe $\mathcal{U} = \{0, 1, \ldots m^2 - 1\}$
  (extends easily to $\mathcal{U} = \{0, 1, \ldots m^k - 1\}$, $k > 2$)

Compact universal families $\mathcal{H}$ exist and are efficient to construct

Kleinberg&Tardos (section 13.6) describe one construction based on modular arithmetic

Basic idea:

- any element $x \in \mathcal{U}$ can bed expressed as a *pair* $(x_0, x_1) \in \{0, 1, \ldots m-1\}^2$, where $x_0 = x \div m$ and $x_1 = x \bmod m$.
- for any $a = (a_0, a_1) \in \{0, 1, \ldots m-1\}^2$, we can define a hash function

$$h_a(x) = (a_0 x_0 + a_1 x_1) \bmod m.$$

Compact universal families $\mathcal{H}$ exist and are efficient to construct

Kleinberg&Tardos (section 13.6) describe one construction based on modular arithmetic

To demonstrate universality of $\mathcal{H}$, the set of all m^2 such functions, we need to show that if $x \neq y$ then for at most $|\mathcal{H}|/m = m$ choices for $a = (a_0, a_1)$, $h_a(x) = h_a(y)$.

But

$h_a(x) = h_a(y)$

$\Rightarrow (a_0 x_0 + a_1 x_1) \equiv (a_0 y_0 + a_1 y_1) \bmod m$

$\Rightarrow a_0(x_0 - y_0) \equiv a_1(x_1 - y_1) \bmod m$

Compact universal families $\mathcal{H}$ exist and are efficient to construct

Kleinberg&Tardos (section 13.6) describe one construction based on modular arithmetic

But if $(x_0 \neq y_0)$ $(x_1 \neq y_1$ is similar) then

- $(x_0 - y_0)$ has a *unique inverse* mod m (here we use primality)
- so...$a_0 \equiv a_1(x_1 - y_1)(x_0 - y_0)^{-1} \bmod m$
- so...for fixed a_1 (m choices) there is a *unique* choice for a_0

Finding the closest pair of points in a point set

Problem definition

Given a collection of n points in real d-dimensional space, identify the pair of points $\{p_i, p_j\}$whose *separation* ($||p_i - p_j||$) is smallest.

Naive solution...

Compute all $\Theta(n^2)$ inter-point distances, and minimize...
Can we do better?

Closest pair in $\Re^1$

Simple $O(n \lg n)$-time algorithm by reduction to sorting.
Is there an analogous result in higher dimensions?

Finding the closest pair of points in a point set

A divide-and-conquer approach in $\Re^2$

Algorithm closest-pair in $\Re^2$

1: split point set into two halves, by x-coordinate
2: find the closest pair within each half (recursively)
3: using the left-separation σ_L and right-separation σ_R, find a left-right pair (if any) that has separation $\sigma < \min\{\sigma_L, \sigma_R\}$

Cost?

- $O(n \lg n)$, because (i) we can pre-sort by dimension, (ii) maintain this sort in subproblems, and use the y-sort to implement the combine step in $O(n)$ time.
- extends to higher dimensions as well

See Kleinberg&Tardos (Section 5.4) for full details...

Finding the closest pair of points in a point set

A randomized incremental approach in $\Re^2$

Algorithm randomized closest-pair in $[0,1]^2$

re-order input points randomly: $p_1, p_2, \ldots, p_n$
$\sigma_{\min} \leftarrow \sigma(p_1, p_2)$; $i \leftarrow 3$
while $i < n + 1$ **do**
    while $\mathrm{N}_{\sigma_{\min}}(p_i) \cap \{p_1, \ldots, p_{i-1}\} = \emptyset$ **do**
        $i \leftarrow i + 1$
    end while
    $p_j \leftarrow$ closest point in $\{p_1, \ldots, p_{i-1}\}$ to p_i
    $\sigma_{\min} \leftarrow \sigma(p_i, p_j)$
    $i \leftarrow i + 1$
end while

where $\mathrm{N}_{\sigma_{\min}}(p)$ denotes the $\sigma_{\min}$-neighbourhood of p

Finding the closest pair of points in a point set

A randomized incremental approach in $\Re^2$

Algorithm randomized closest-pair in $[0,1]^2$

re-order input points randomly: $p_1, p_2, \ldots, p_n$
$\sigma_{\min} \leftarrow \sigma(p_1, p_2)$; $i \leftarrow 3$
while $i < n+1$ **do**
    while $\mathrm{N}_{\sigma_{\min}}(p_i) \cap \{p_1, \ldots, p_{i-1}\} = \emptyset$ **do**
        $i \leftarrow i+1$
    end while
    $p_j \leftarrow$ closest point in $\{p_1, \ldots, p_{i-1}\}$ to p_i
    $\sigma_{\min} \leftarrow \sigma(p_i, p_j)$
    $i \leftarrow i+1$
end while

where $\mathrm{N}_{\sigma_{\min}}(p)$ denotes the $\sigma_{\min}$-neighbourhood of p

Finding the closest pair of points in a point set

A randomized incremental approach in $\Re^2$

The algorithm proceeds in a sequence of *stages* during which the *stage invariant* $\mathrm{N}_{\sigma_{\min}}(p_i) \cap \{p_1, \ldots, p_{i-1}\} = \emptyset$ holds.

Between stages we update $\sigma_{\min}$ by computing
$p_j \leftarrow$ closest point in $\{p_1, \ldots, p_{i-1}\}$ to p_i

The cost depends on

- the cost of testing the stage invariant
- the number and cost of stage transitions

Testing the stage invariant

- divide space $[0,1]^2$ into cells of side length $\sigma_{\min}/2$
- point p belongs to $\mathrm{cell}(p) = (\lfloor \frac{p.x}{\sigma_{\min}/2} \rfloor, \lfloor \frac{p.y}{\sigma_{\min}/2} \rfloor)$
- by construction *no cell contains more than one point* among $\{p_1, \ldots, p_{i-1}\}$
- stage invariant fails if point p_i has a point among $\{p_1, \ldots, p_{i-1}\}$ in the *neighbourhood* of $\mathrm{cell}(p_i)$

Cost of stage transitions

- when $\sigma_{\min}$ is updated need to rebuild the neighbourhood search structure
- cost is proportional to i if we rebuild on i-th insertion
  recall implicit initialization
- how many stages do we expect?
  $\Theta(\lg n)$

Total expected cost

- find operations: $O(n)$ (only look in $O(1)$ cells per point)
- distance calculations: $O(n)$ (only compute distance with $O(1)$ neighbours)
- rebuild operations: $O(s)$, where s is the number of stages
- insert operations: $n + \sum_{1 \leq i \leq n}(iX_i)$, where $X_i = 1$ if the i-th insert leads to a closest pair update (and $X_i = 0$ otherwise).

So the total expected cost is $O(n)$.

See Kleinberg&Tardos (Section 13.7) for full details...

Next time...

Dictionaries with non-uniform access patterns

- fixed (known) access frequencies
- unknown/changing access frequencies