# Locality-Sensitive Hashing

## Algorithmic Problems Around the Web #5


Yury Lifshits

http://yury.name


CalTech, Fall'07, CS101.2, http://yury.name/algoweb.html



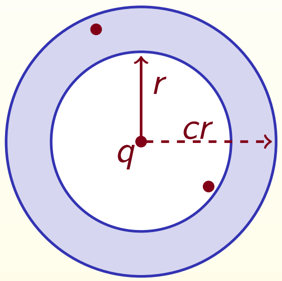
# Outline





# Approximate Algorithms

***c*-Approximate *r*-range query:** if there at least one $p \in S: \ d(q,p) \le r$ return some $p': \ d(q,p') \le cr$




***c*-Approximate nearest neighbor query:** return some $p' \in S: \ d(p',q) \le cr_{NN}$, where $r_{NN} = \min_{p \in S} d(p,q)$

Today we consider only range queries



# Today's Focus

**Data models:**

- $d$-dimensional Euclidean space: $\mathbb{R}^d$
- Hamming cube: $\{0,1\}^d$ with Hamming distance

**Our goal:** provable performance bounds

- Sublinear search time, near-linear preprocessing space
- Logarithmic search time, polynomial preprocessing space

**Still an open problem:** approximate nearest neighbor search with logarithmic search and linear preprocessing

## Part I

## Locality-Sensitive Hashing: General Scheme

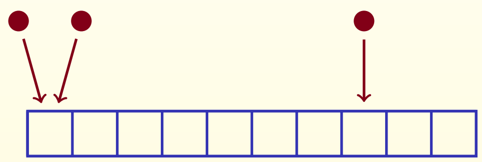


# Definition of LSH

Indyk&Motwani'98

**Locality-sensitive hash family** $\mathcal{H}$ with parameters $(c, r, P_1, P_2)$:

- If $\|p - q\| \le r$ then $\mathcal{Pr}_{\mathcal{H}}[h(p) = h(q)] \ge P_1$
- If $\|p - q\| \ge cr$ then $\mathcal{Pr}_{\mathcal{H}}[h(p) = h(q)] \le P_2$





# The Power of LSH

Notation: $\rho = \frac{\log(1/P_1)}{\log(1/P_2)} < 1$

**Theorem**

*Any $(c, r, P_1, P_2)$-locality-sensitive hashing leads to an algorithm for $c$-approximate $r$-range search with (roughly) $n^{\rho}$ query time and $n^{1+\rho}$ preprocessing space*

Proof in the next four slides



# LSH: Preprocessing

Composite hash function: $g(p) =< h_1(p), \dots, h_k(p) >$

Preprocessing with parameters $L, k$:

1. Choose at random $L$ composite hash functions of $k$ components each
2. Hash every $p \in S$ into buckets $g_1(p), \dots, g_L(p)$

Preprocessing space: $\mathcal{O}(Ln)$

## LSH: Search

1. Compute $g_1(q), \ldots, g_L(q)$
2. Go to corresponding buckets and explicitly check $d(p,q) \leq ?cr$ for every point there
3. Stopping conditions: (1) we found a satisfying object or (2) we tried at least $3L$ objects

Search time is $\mathcal{O}(L)$



## LSH: Analysis (1/2)

In order to have probability of error at most $\delta$ we set $k, L$ such that

$$P_2^k n \approx 1 \qquad L \approx (1/P_1)^k \log(1/\delta)$$

Solving these constraints:

$$k = \frac{\log n}{\log(1/P_2)}$$

$$L = (1/P_1)^{\frac{\log n}{\log(1/P_2)}} \log(1/\delta) = n^{\frac{\log(1/P_1)}{\log(1/P_2)}} \log(1/\delta) = n^{\rho} \log(1/\delta)$$



## LSH: Analysis (2/2)

The expected number of $cr$-far objects to be tried is $P_2^k L n \approx L$

For true $r$-neighbor the chance to be hashed to the same bucket as $q$ is at least
$1 - (1 - (1/P_1)^k)^L \geq 1 - (1/e)^{\frac{L}{(1/P_1)^k}} \geq 1 - \delta$

Preprocessing space $\mathcal{O}(Ln) \approx n^{1+\rho+o(1)}$
Search $\mathcal{O}(L) \approx n^{\rho+o(1)}$



# Part II

# Andoni&Indyk'06 Hashing

# Ball Grids Hashing: Idea

1. Apply low distortion embedding $A$ into $t$-dimensional Euclidean space
2. Set up $U$ $4w$-step grids of $w$-radius balls that all together cover $t$-dimensional space
3. Hash object $p$ to the id of the first ball covering $A(p)$



# BG Hashing: Initialization

Parameters: $t = \log^{2/3} n, w = r \log^{1/6} n, U = 2^{t \log t} \log n$

- Construct $d \times t$ matrix $A$ taking every element at random from normal distribution $N(0, \frac{1}{\sqrt{t}})$
- For every $1 \leq i \leq U$ choose a random shift $\bar{v}_i \in [0, 4w]^t$



# BG Hashing: Computing

1. Compute $p' = A(p)$
2. From $i = 1$ to $U$ check whether $p'$ is covered by $i$-th grid of balls. If so return $i$ and ball's center and stop.
3. If no such ball found return FAIL

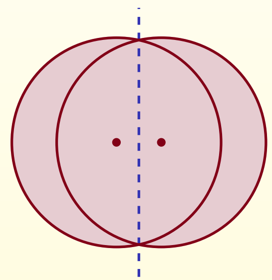


# BG Hashing: Analysis

**Fact:** Probability of $\frac{\|Ap - Ap'\|}{\|p - p'\|} \notin [1 - \varepsilon, 1 + \varepsilon]$ is at most $\exp(-\varepsilon^2 t)$

Given two points $p, s \in \mathbb{R}^t : \|p - s\| = \Delta$:

$$Pr[h(p) = h(s)] = \frac{B(p, w) \cap B(s, w)}{B(p, w) \cup B(s, w)}$$



## BG Hashing: Final Result

3-pages computational proof:

$$\rho = \frac{\log(1/P_1)}{\log(1/P_2)} = 1/c^2 + o(1)$$

**Theorem (Andoni & Indyk 2006)**

*Consider $c$-approximate $r$-range search in $d$-dimensional space. Then for every $\delta$ there is a randomized algorithm with (roughly) $n^{1/c^2+o(1)}$ query time and $n^{1+1/c^2+o(1)}$ preprocessing space. For every query this algorithm answers correctly with probability at least $1-\delta$*



## Future of LSH

**Achievements:**

- Provably sublinear search time
- Utilization of low-distortion embedding

**Current drawbacks**:

- Probability of error can not be amplified only in preprocessing stage, it can not be decreased to $1/n$
- Asymptotic analysis of power degree: from what place $n^{1/c^2+o(1)}$ is really sublinear?
- For nearest neighbor search $c = \max \frac{r_{NN}(q)}{r_{FN}(q)}$, where $r_{FN}(q)$ is the farthest neighbor. This might be pretty close to 1



## Exercise

Prove that $2^{\mathcal{O}(t)}$ number of randomly chosen $(w, 4w)$ ball grids is enough to cover $t$-dimensional space with probability $1/2$

Thanks for your attention! Questions?



## References

**Course homepage** http://yury.name/algoweb.html

Y. Lifshits
The Homepage of Nearest Neighbors and Similarity Search
http://simsearch.yury.name

A. Andoni, P. Indyk
Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. FOCS'06
http://web.mit.edu/andoni/www/papers/cSquared.pdf