问求算法设计与实践08

哈希

Hash Functions

- 将对象从原空间映射到一个’小空间’中(维度低，值域范围小)

- 计算快
- 冲突少（函数设计，散列表设计）

Hash Functions

- 哈希对象
  - 数字
  - 字符串
  - 树，图
  - 文件
  - …

Hash Functions

- $F(x)=x \bmod P$;
- $F(x)=x^{10} \bmod P$;
- …

问题来了..

- 维护一个数据集
- 每次可能插入一个新的数字
- 每次可能查询数字x是否在集合内

排序+二分

- 插入O(NlogN)
- 查询O(logN)

平衡二叉树

- 插入O(logN)
- 查询O(logN)

Hash Table！

- 插入O(1)
- 查询O(1)

字符串哈希

Sample Input

```
[expelliarmus] the disarming charm
[rictusempra] send a jet of silver light to hit the enemy
[tarantallegra] control the movement of one's legs
[serpensortia] shoot a snake out of the end of one's wand
[lumos] light the wand
[obliviate] the memory charm
[expecto patronum] send a Patronus to the dementors
[accio] the summoning charm
@END@
4
[lumos]
the summoning charm
[arha]
take me to the sky
```

Sample Output

```
light the wand
accio
what?
what?
```

字符串哈希算法

- BKDRHash，APHash，DJBHash，JSHash，RSHash，SDBMHash，PJWHash，ELFHash…
- https://www.byvoid.com/blog/string-hash-compare/

BKDRHash

- 对于字符串S=‘acb’
- H(S)=‘a’*31^2+’c’*31+’b’

BKDRHash

- 递推计算
- for i=0:len-1
  - H=H*31+S[i];

BKDRHash

- 如果有两个串的hash值相同，我们'几乎'可以说他们就是同一个串！
- 实际做题中，合适的选取seed，基本可以全对
- 实际应用中，可能还是会有一些反例

Rabin-Karp Algorithm

- 给定字符串A，B
- 问B是否是A的子串

- 暴力 O(length(A)*length(B))
- KMP O(length(A)+length(B))
- RK 平均O(length(A)+length(B))

预处理

- 给定一个字符串A
- 给定多个查询 (B,i,j), 表示B是否与Ai...Aj为相同的字符串
- 怎么在O(1)时间内求出H(Ai...Aj)?

前缀和trick

- Pre[i]=H(A1…Ai)
- Pre[i]=Pre[i-1]*31+Ai

- H(Ai…Aj)=?

前缀和trick

- Pre[i]=H(A1…Ai)
- Pre[i]=Pre[i-1]*31+Ai

- H(Ai…Aj)=Pre[j]-Pre[i-1]*31^(j-i+1)

溢出int?

- 用unsigned longlong
- 天然的模系统
- 加减乘法都满足模的性质

Problem B

- 给定一个字符串
- 找出至少出现M次的最长子串

Problem B

- 用前缀和预处理好哈希值
- 二分长度len
- 找出所有长度为len的哈希值 (O(N)个)
- 将哈希值排序，看是否有至少M个相同的哈希值(或者用一个哈希表来判断)
- 复杂度O(NloglogN)或O(NlogN)

BNUOJ 34490 Justice String

- 给定两个串A，B
- 问A中是否有一个子串，与B长度相等，且最多只有2个字母不一样

BNUOJ 34490 Justice String

- 完全一样
    - KMP, RK
- 一个字母不一样
    - 在A中枚举起点
    - 二分一个最远的距离使得匹配
    - 看后半段是否匹配

BNUOJ 34490 Justice String

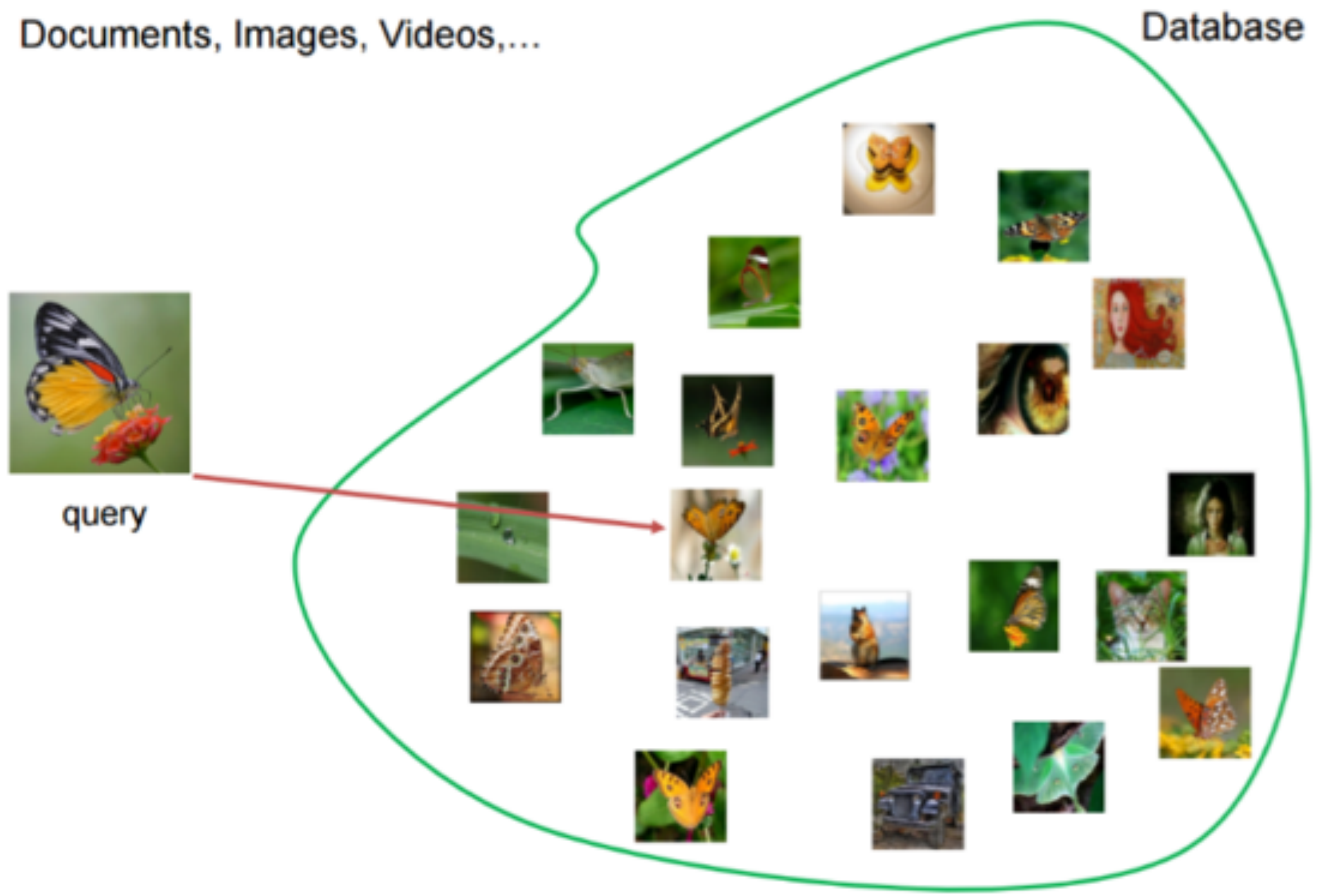
- 两个个字母不一样
  - 在A中枚举起点 O(N)
    - 二分一个最远的距离使得匹配 O(logN)
    - 以匹配点后一个点为起点
    - 二分一个最远的距离使得匹配 O(logN)
    - 看最后一段是否匹配 O(1)
- O(NlogN)

Similarity-Preserving Hashing

- 当我们不仅仅考虑如何避免冲突
- 由于哈希函数一般是映射到一个海明空间
- 我们希望相似的对象，拥有’相似’的哈希值(海明距离更近)

Nearest Neighbor (NN) Search

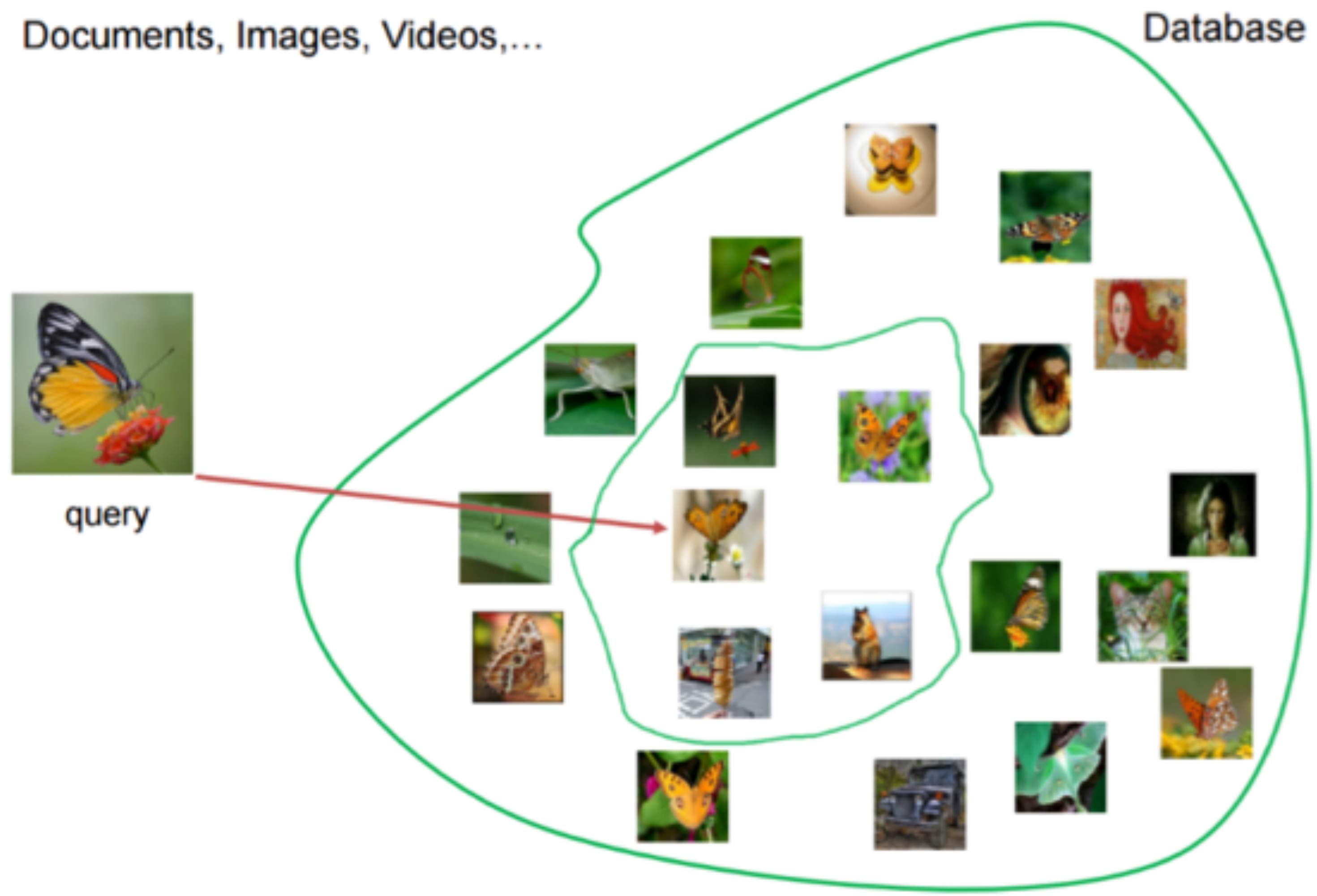
Given a database $X = \{x_1, x_2 ...\} \quad x_i \in \mathbb{R}^D$ and a query $q \in \mathbb{R}^D$.
Return a point y which $y = \arg\min_{y \in X} dist(y, q)$.



There are lots of methods proposed to fast NN search (i.e K-D tree), but they fail in the large scale or high-dimensional cases.

Approximate Nearest Neighbor (ANN) Search

A point z is the approximate nearest neighbor to q , if $dist(z,q) \leq (1+\epsilon)dist(y,q)$



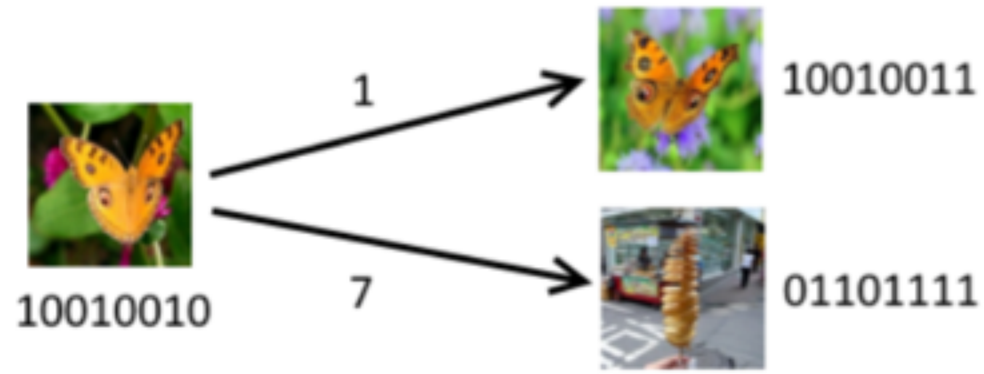
The most famous method of ANN is Locality Sensitive Hashing (LSH), which is based on hashing.

Hashing Technique

Hash function map a object to a limited space

$$\text{i.e } f : \mathbb{R} \rightarrow \{0, 1, ...255\}$$

Unlike traditional hashing methods which try to reduce colliding, hashing methods of ANN try to preserve distance in origin space.



Use low-dimensional hamming distance to approximate euclidian distance in origin space.

pros
- constant or sub-linear retrieve time
- low storage cost

cons
- hard to optimize a discrete problem

Supervised Version

For real complex application images,videos...

- Object may have semantic information
- Two images are semantically similar cannot fully depend on the distance in feature (GIST,SIFT...) space
- There are tags or other supervised information could be utilized

In the setting of Supervised version, we are given the ground-truth of similarity between any two training samples.

Locality-Sensitive Hashing

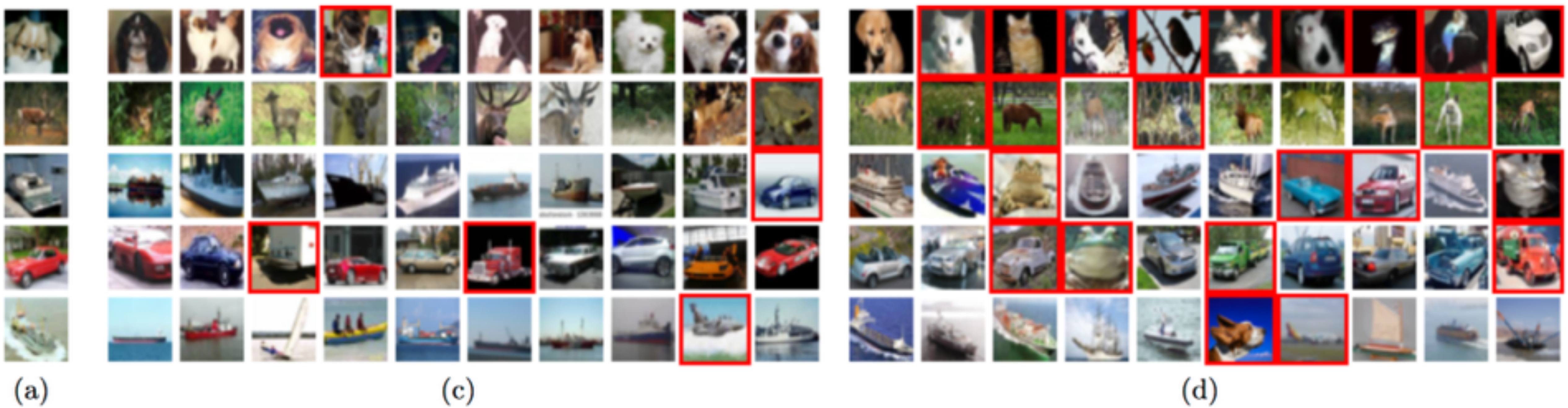
- Bit Sampling
- Random Projection
- …

Learning to Hash

- Generating Hashing Functions from data, rather than being independent of data
- Recent research indicates that Learning to Hash can achieve the same accuracy with much smaller codes length!

- Gong, Yunchao, and Svetlana Lazebnik. "Iterative quantization: A procrustean approach to learning binary codes." *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011.

LFH(Zhang et al. 2014)

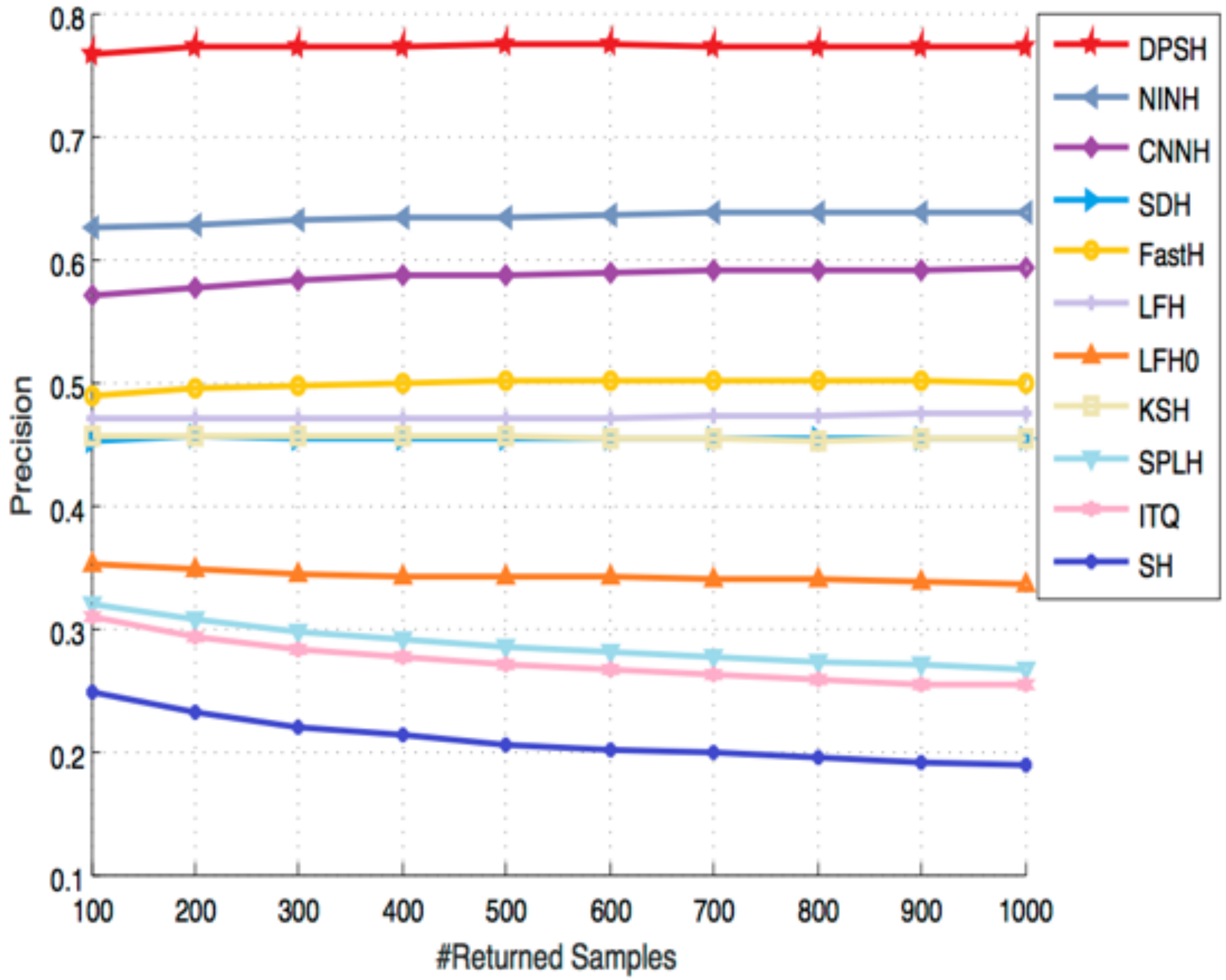


COSDISH(Kang et al. 2016)

- MAP

Method	CIFAR-10 (60K)			
	8-bits	16-bits	32-bits	64-bits
COSDISH	**0.4986**	**0.5768**	**0.6191**	**0.6371**
SDH	0.2642	0.3994	0.4145	0.4346
LFH	0.2908	0.4098	0.5446	0.6182
TSH	0.2365	0.3080	0.3455	0.3663
KSH	0.2334	0.2662	0.2923	0.3128
SPLH	0.1588	0.1635	0.1701	0.1730
COSDISH_BT	**0.5856**	**0.6681**	**0.7079**	**0.7346**
FastH	0.4230	0.5216	0.5970	0.6446

DPSH(Li et al. 2015)



- http://arxiv.org/abs/1511.03855

DPSH(Li et al. 2015)

Figure 3. Retrieval results (top 15 returned images) for ten query images from CIFAR-10 using Hamming ranking on 48-bits hash code. Red rectangles indicate mistakes.