# 程序设计实习（实验班-2023春）
# 距离度量及其计算

授课教师：姜少峰
助教：张宇博 楼家宁

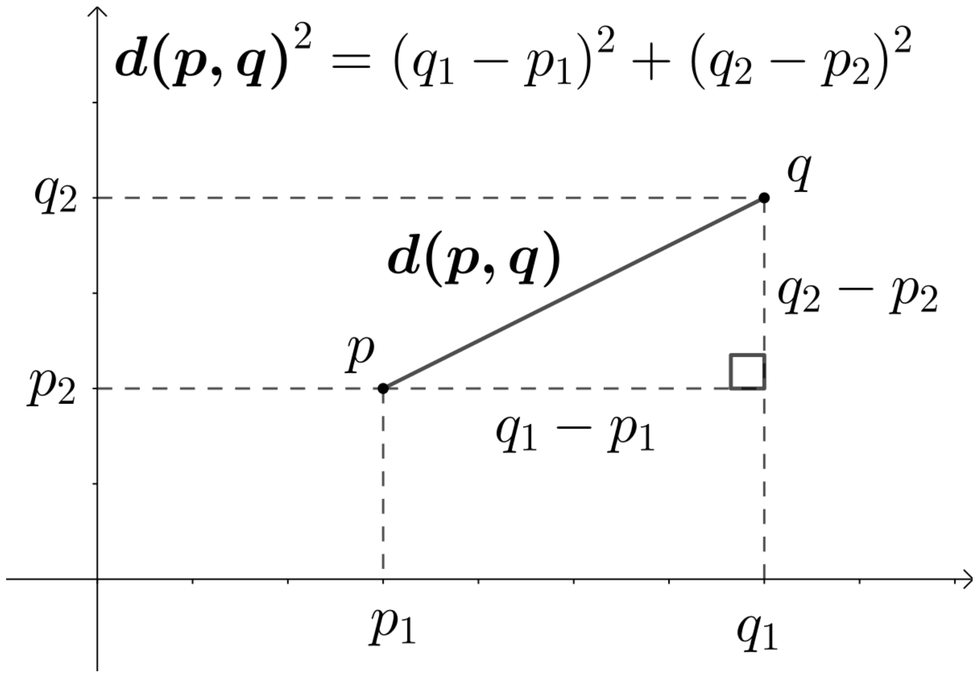
Email: shaofeng.jiang@pku.edu.cn

# 距离与相似度

- 给定一个数据集$X$，定义一个距离/相似度度量$d: X \times X \to \mathbb{R}_+$
- 相似度：越大越相似，越小越不相似
- 距离：“不相似度”
  - 越小越相似，越大越不相似
  - 一般满足三角形不等式

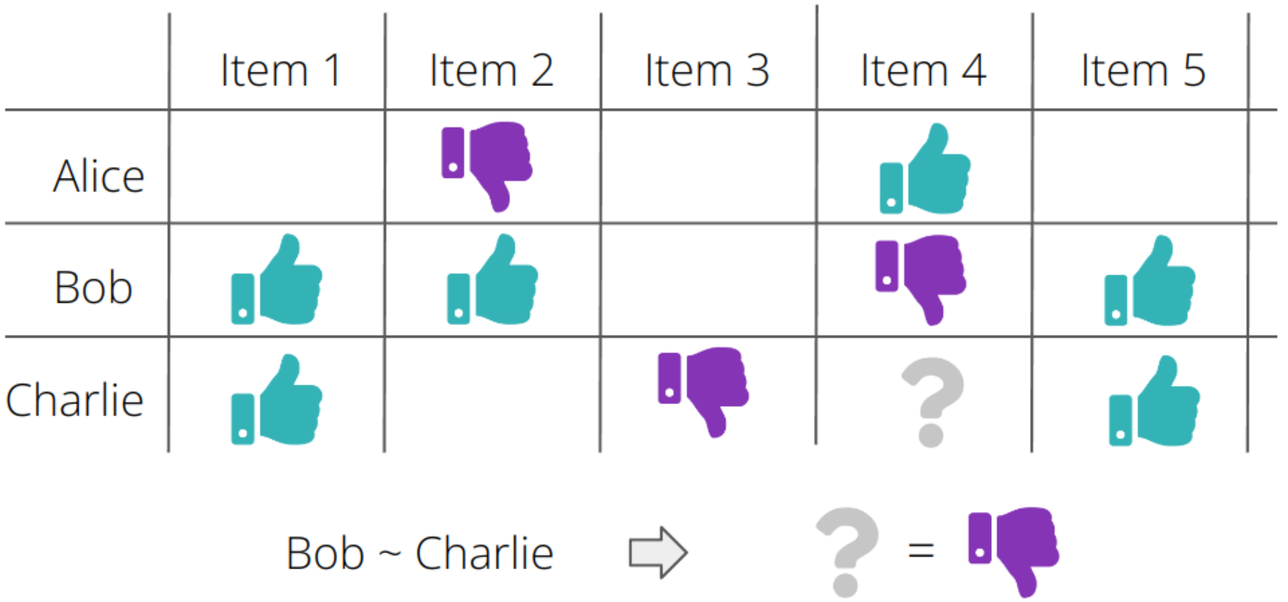
$$d(x,y) + d(y,z) \geq d(x,z)$$

最常见的距离：欧氏距离



# 基于距离的应用举例

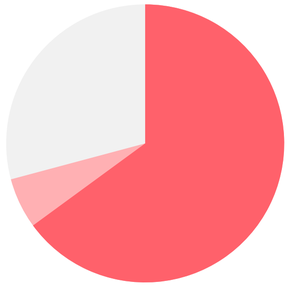
# 推荐系统


Item 1
Item 2
Item 3
Item 4
Item 5
Alice
Bob
Charlie
Bob ~ Charlie
=

# 抄袭检测

## 类似的：网页去重

**29**
**RESULTS FOUND**



- Identical
- Minor Changes
- Omitted Words

**73%**
**MATCH**

See Alerts

There are billions of galaxies filled with billions of stars. Each star has the potential to have planets orbiting it. Does life exist on some of those planets? In this module, students discover how scientists find planets and other astronomical bodies through the wobble (also known as Doppler spectroscopy or radial-velocity) and transit methods. Students compare zones of habitability around different star types, discovering the zone of liquid water possibility around each star type. Finally, students explore how scientists use spectroscopy to learn about atmospheres on distant planets.

first is to study the effects of exposure to microgravity on biological systems to reduce the risks of manned space flight. The second is to use the microgravity environment to broaden scientific knowledge about the influence of gravity on living systems.

RESULTS (20)

**Galaxy | Wikipedia.org**

*https://en.wikipedia.org/wiki/Galaxy*

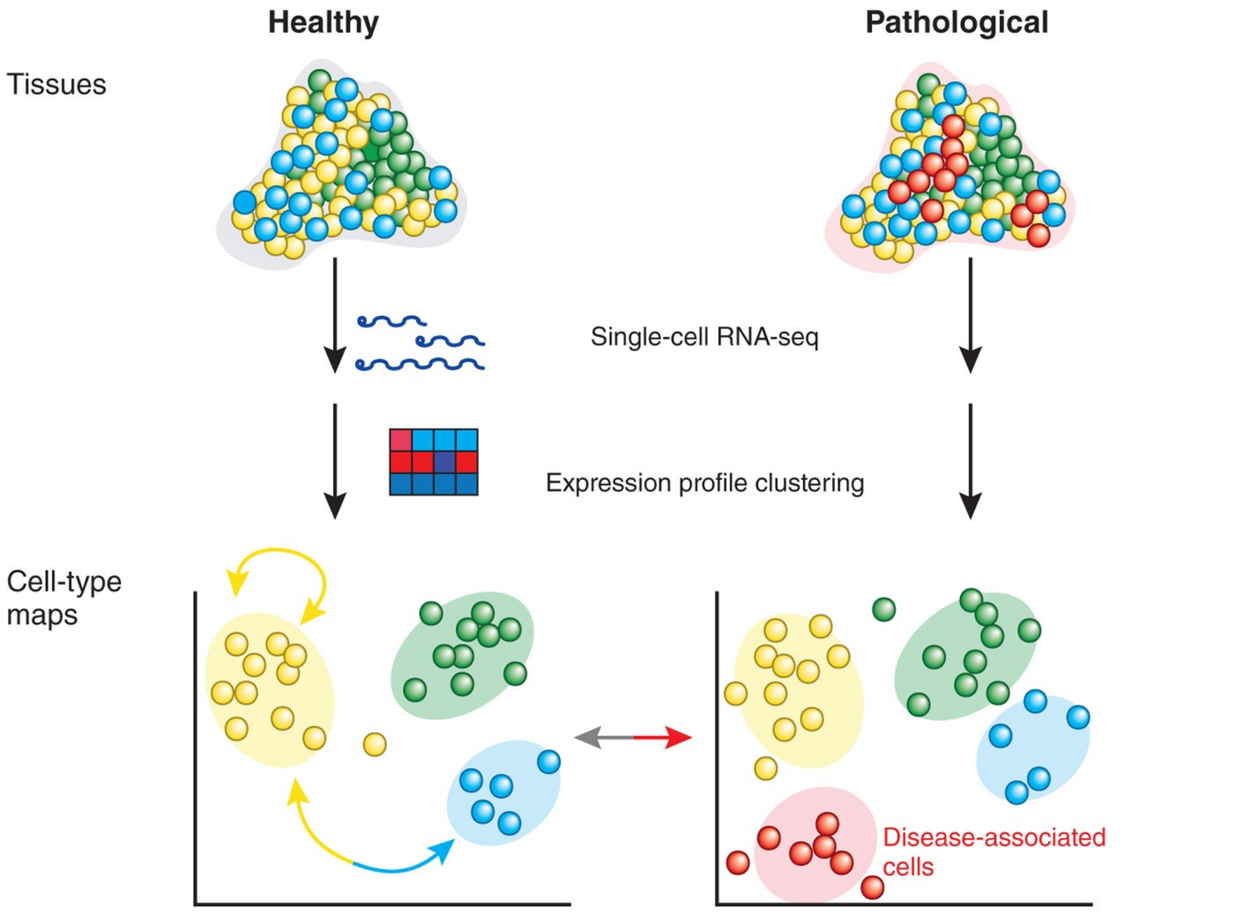
Skip to content...

46% Similar words

**Space - High Adventure Science: Is there lie in space? | Wikipedia.org**

*https://en.wikipedia.org/wiki/space-high-a...*

# 生物学应用


Healthy
Pathological
Tissues
Single-cell RNA-seq
Expression profile clustering
Cell-type maps
Disease-associated cells
Types of analyses
Within cell type
• Stochasticity, variability of transcription
• Regulatory network inference
• Allelic expression patterns
• Scaling laws of transcription
Between cell types
• Identify biomarkers
• (Post)-transcriptional differences
Between tissues
• Cell-type compositions
• Altered transcription in matched cell types

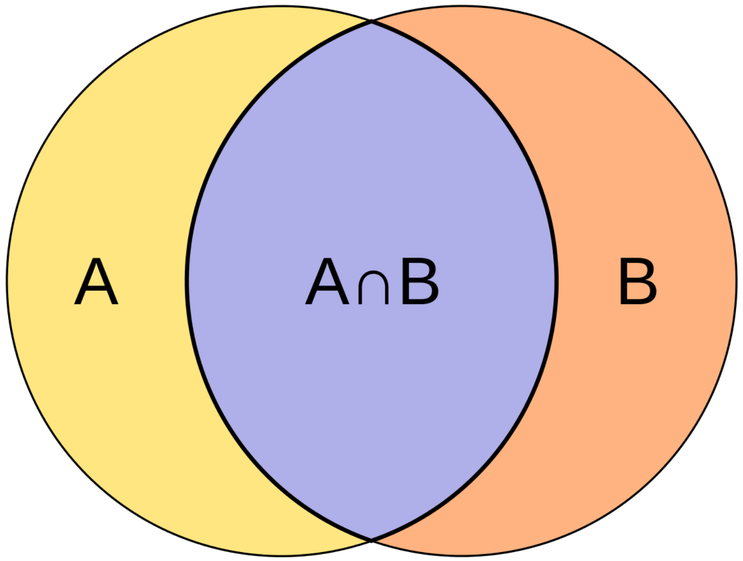
# 集合的相似度度量

# Jaccard Similarity

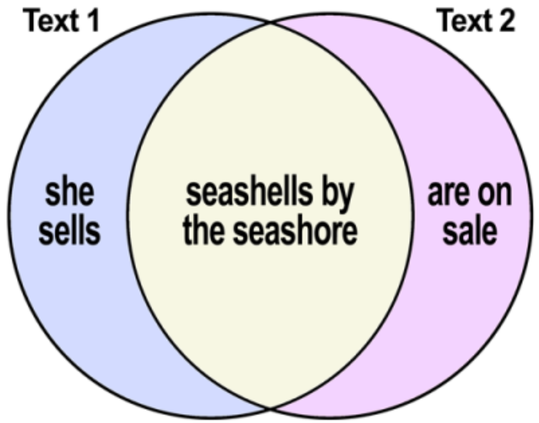
- 两个集合$A, B \subseteq [n]$

是一个[0,1]的数

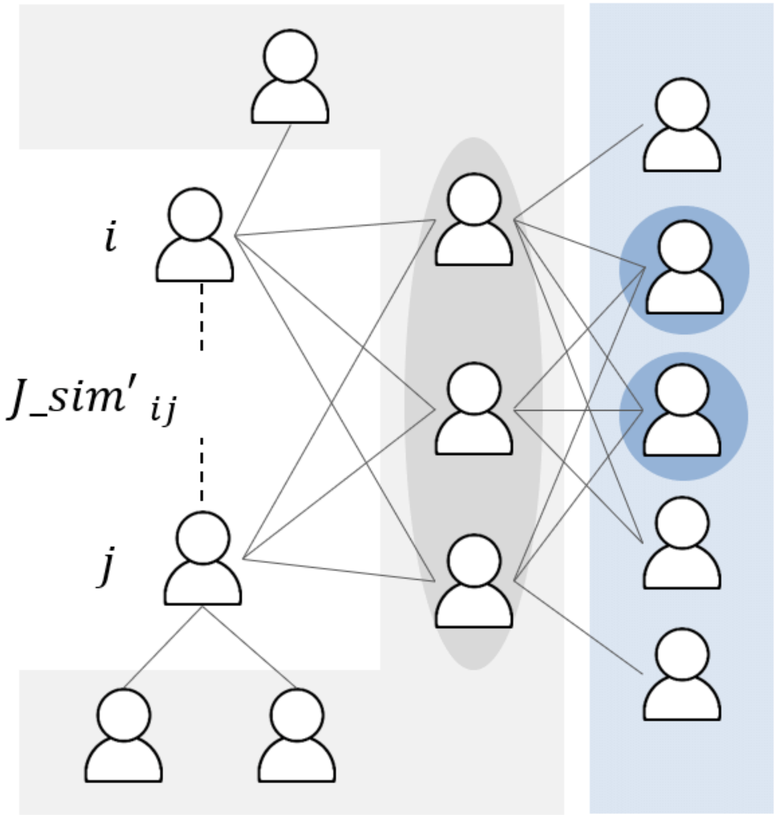
$$J(A,B) := \begin{cases} \dfrac{|A \cap B|}{|A \cup B|} & A \cup B \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$



文本的相似度（考虑单词的集合）




社交网络上用户的社交距离（考虑各自邻居的集合）



# 如何计算J(A, B)?

$$J(A,B) := \begin{cases} \frac{|A \cap B|}{|A \cup B|} & A \cup B \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

- 直接计算：借助集合数据结构（设O(1)代价），可以在$O(|A| + |B|)$计算
- 如果数据集有很多点（集合）$\{A_i\}_i$，经常需要查询两个点的$J(A_i, A_j)$呢?

# 一种基于哈希的思路

- 对每个集合$A_i$，希望计算/预处理一个哈希值$h(A_i)$
- 使得$J(A_i, A_j)$可以通过$h(A_i)$和$h(A_j)$直接得到
- 只需要$O(|A_i|)$时间预处理$h(A_i)$，之后查询$J(A_i, A_j)$都是$O(1)$

# MinHash

- 设所有的集合$A_i$的元素都来自某个universe [n]
- 设$h:[n]\to[0,1]$为一个随机哈希 注意此处映射到实数
- 对每个$A_i$,计算$h_{\min}(A_i):=\min_{x\in A_i} h(x)$

Claim：$\Pr[h_{\min}(A)=h_{\min}(B)]=J(A,B)$

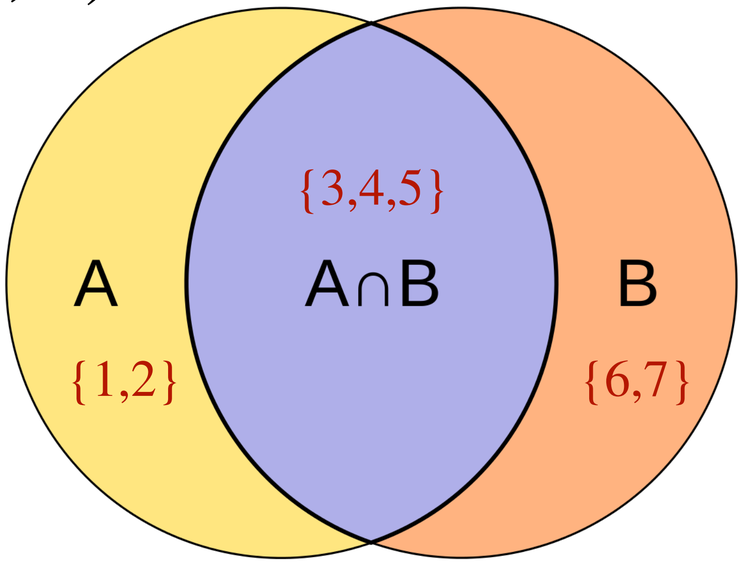
# $h_{\min}$的性质

- 一般地，考虑一个任意的$S \subseteq [n]$，并考察$h_{\min}(S) := \min_{x \in S} h(x)$
- 由于所有$x$上的$h(x)$是独立同分布的，任何$x \in S$成为min的概率都是一样的
- 更一般的，如果把$S$元素按照$h(x)$升序排列，则会产生一个（均匀）随机排列
  - 因此MinHash等价于把$[n]$上的元素做随机排列

直接存随机排列是低效的
hash可在用到某value才现算现存

# 验证MinHash主要性质

Claim：$\Pr[h_{\min}(A) = h_{\min}(B)] = J(A, B)$




- 考虑$A = \{1,2,3,4,5\}$和$B = \{3,4,5,6,7\}$两个集合
- 可以假定$h$在$\{1,\ldots,7\}$上各不相等
- 观察：$h_{\min}(A) = h_{\min}(B)$仅当$h_{\min}$取在$A \cap B$
  - 例如，$h_{\min}(B)$只能取在$\{3\ldots7\}$上，若$h_{\min}(A)$取在$\{1,2\}$上则一定不能相等
- 一共7个元素，min取在中间3个元素上，概率是3/7

$|A \cup B|$　　$|A \cap B|$　　$|A \cap B| / |A \cup B| = J(A, B)$

# 转化成随机算法

- 设$X := \begin{cases} 1 & h_{\min}(A_i) = h_{\min}(A_j) \\ 0 & \text{oth.} \end{cases}$，则$\Pr[X = 1] = J(A_i, A_j)$
- 完整算法（多次试验取平均值）：
  - 采用T个独立的随机哈希$h^{(1)}, \ldots, h^{(T)}$

    如何实现?
  - 对每个$A_i$和$h^{(t)}$，计算$h^{(t)}_{\min}(A_i)$

    $O(|A_i| \cdot T)$时间
  - 对查询$A_i, A_j$，计算$\frac{1}{T} \cdot |\{t \in [T] : h^{(t)}_{\min}(A_i) = h^{(t)}_{\min}(A_j)\}|$

    计算的是$h^{(t)}_{\min}$相等的t所占比例

    $O(|T|)$时间

# T取多大?

- 设估计量$X := \frac{1}{T} \cdot |\{t \in [T] : h_{\min}^{(t)}(A_i) = h_{\min}^{(t)}(A_j)\}|$

**Chernoff bound.** 设$X_1, \ldots, X_n \in [0,1]$是独立随机变量。

令$X = X_1 + \ldots + X_n$。则对$t \in [0,1]$有

$$\Pr[|X - \mu| \geq t\mu] \leq 2\exp(-t^2\mu/3)$$

- 设$X_t \in \{0,1\}$, such that $X_t = 1 \iff h_{\min}^{(t)}(A_i) = h_{\min}^{(t)}(A_j)$

Chernoff告诉我们$T = O(\log n)$有失败概率$1/\text{poly}(n)$

# 作业四：用MinHash近似Jaccard Similarity

- http://cssyb.openjudge.cn/23hw4/
- Deadline: 4月5日

# 向量的相似性度量

# Cosine Similarity

- 考虑两个向量$x, y \in \mathbb{R}^d$，定义$\sigma(x, y) := \dfrac{\langle x, y \rangle}{\|x\|_2 \|y\|_2}$

$\sigma(x, y) = \cos(\theta(x, y))$

- 典型应用：文本相似度（TF-IDF）
  - TF = term frequency：一个单词在文档中出现的频率
  - IDF = inverse document freq.：log(总文档数 / 单词出现在多少文档)
  - 对单词w，TF-IDF(w) = TF(w) * IDF(w)，做成一个下标是单词、值TF-IDF向量

一般非常的高维、稀疏

| Word | df | idf |
|---|---|---|
| Romeo | 1 | 1.57 |
| salad | 2 | 1.27 |
| Falstaff | 4 | 0.967 |
| forest | 12 | 0.489 |
| battle | 21 | 0.246 |
| wit | 34 | 0.037 |
| fool | 36 | 0.012 |
| good | 37 | 0 |
| sweet | 37 | 0 |

We see that "Romeo", "Falstaff", and "salad" appears in very few plays, so seeing these words, one could be quite certain which play it is. In contrast, "good" and "sweet" appears in every play and are completely uninformative as to which play it is.

# Cosine Similarity的哈希：SimHash

- 类似于MinHash，我们想找到一个h，使得$\Pr[h(x) = h(y)]$可以反映$\sigma(x, y)$
- 算法：

如何生成：生成d个独立的标准正态变量，然后将该随机向量归一化

  - 生成一个$d$维随机高斯向量$w$（等价于在$d$维单位球面取一个均匀随机点）
  - $h(x) := \text{sgn}(\langle w, x \rangle)$，其中$\text{sgn}(x) = \begin{cases} 1 & x \geq 0 \\ 0 & \text{otherwise} \end{cases}$是符号函数

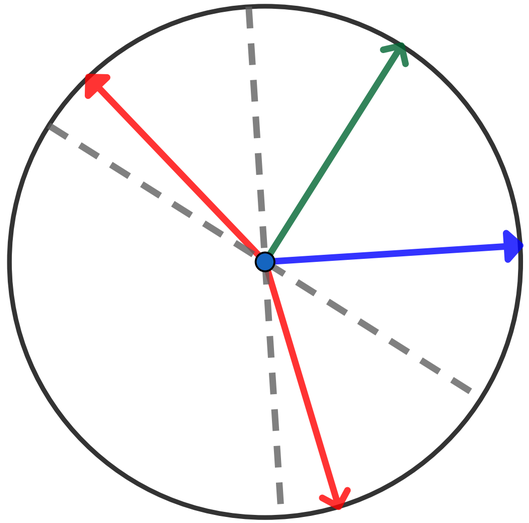
- Claim: $\forall x, y \in \mathbb{R}^d, \quad \Pr[h(x) \neq h(y)] = \dfrac{\theta(x, y)}{\pi}$

$\theta(x, y) \in [0, \pi]$是x和y的向量夹角

# 验证SimHash主要性质

$$\text{Claim: } \forall x, y \in \mathbb{R}^d, \quad \Pr[h(x) \neq h(y)] = \frac{\theta(x, y)}{\pi}$$

**异号：一个夹锐角另一个钝角形成$2\theta$区域，总共$2\pi$**



对于d维：

- 异号$\{w : \langle x, w\rangle \geq 0, \langle y, w\rangle < 0\}$
- 对应于两个超平面相交，二面角$\theta$
- 总共的二面角取值范围$2\pi$

# 多次试验取平均值：到Hamming空间的映射

- 类似MinHash，可以取T个独立SimHash取平均值
- 针对$x, y \in \mathbb{R}^d$的估计量：$\frac{1}{T}\sum_{t=1}^{T} \mathbb{I}(h^{(t)}(x) \neq h^{(t)}(y))$
- 事实上，可以看作是将$\mathbb{R}^d$上的点对应到T维的Hamming！
  - $f(x) := (h^{(1)}(x), \ldots, h^{(T)}(x))$
  - 上述估计量就等于$(f(x) \oplus f(y))/T$
  - 即：$\mathrm{dist}_{\mathrm{H}}(f(x), f(y)) \approx \theta(x, y)/\pi$

T维Hamming上的点是T维的binary string，距离是异或，即$\mathrm{dist}_H(x, y) = |\{i : x_i \neq y_i\}|$

# Hamming空间近似最近邻查询

# 目标

- 设有n个d维的Hamming空间$\mathbb{H}^d$上的点
- 性能要求：
  - 误差参数$\epsilon > 0$

这里典型情况要求$\epsilon \geq 1$

  - 预处理时间$O(dn + n^{1+1/(1+\epsilon)})$
  - 给定任何$q \in \mathbb{H}^d$，可在$O(n^{1/(1+\epsilon)})$时间找到$(1+\epsilon)$-近似最近邻

即：若最近邻距离是r，则返回的点距离 $\leq (1+\epsilon)r$

# 算法：预处理

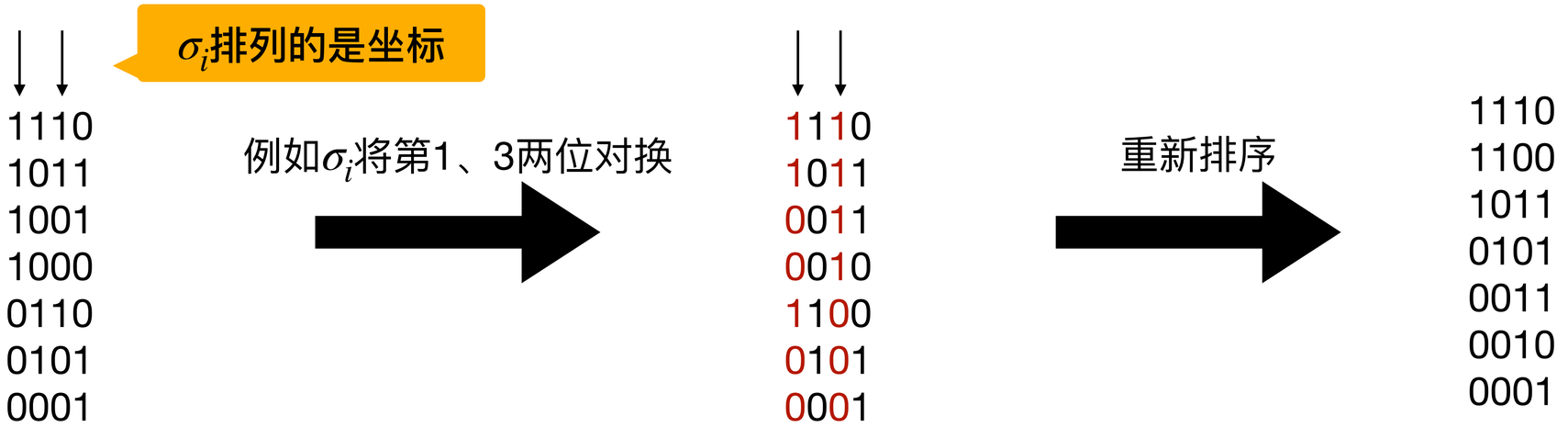
- 设数据点集是$P \subseteq \mathbb{H}^d$，有n个点

实际中经常N可以取很小，比如N = 5或10

  - 设$N = O(n^{1/(1+\epsilon)})$，找N个坐标d的随机排列$\sigma_1, \ldots, \sigma_N$

共生成N份数据的copy，总共Nn个点

  - 对每个排列$\sigma_i$，将所有坐标按$\sigma_i$排列后的n个数据点按照字典序排序



搜索空间大小是Nn（而不是n）

# 算法：给定$q \in \mathbb{H}^d$，找q的近似最近邻

## 暴力版本

- 设 LCP(S, T)为S串和T串的最长公共前缀
- 对所有$1 \le i \le N$，对所有数据点$x \in P$，求$\mathrm{LCP}(\sigma_i(q), \sigma_i(x))$
- 在所有上述$N \cdot n$个LCP中，找最大的前2N个对应的二进制串，做成集合$S$
- 返回q与$S$中的Hamming distance最近邻作为答案
- 瓶颈：找S需要$N \cdot n$时间，下页介绍利用预处理的排序降低到$N \cdot \mathrm{poly}(\log n)$

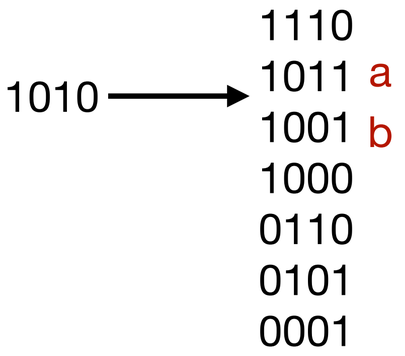
注意：q和x都要用$\sigma_i$重排所有数位

所以$|S| \le 2N$

需要$|S| \le O(N)$次比较

# 高效搜索公共前缀最大的2N个Binary String

- 先考虑N = 1，即只有一个排列。设n个d维的binary string已按字典序排好
- 算法：
  - 二分查找q，找到最接近q的前后两个string a和b
  - 观察：在有序表中，离q越远则与q的公共前缀越短
  - 因此可以从a向前、从b向后每次贪心向公共前缀较大的一个推进
- $N > 1$时，每组二分查找$\sigma_i(q)$前后string $a_i, b_i$，然后从所有$a_i, b_i$考虑贪心推进




使用后缀数组等技术可优化复杂度，但一般没必要因为d比较小

可以利用优先队列进一步优化复杂度；但考虑到$N$一般不大，这步暴力可能反而更高效

# 编程实现的一些提示

根据之前讨论：对SimHash来说，一般$O(\log n)$足够

- 实际应用中长度d比较小，比如$d \leq 64$，因此可以用int64存储
- 字典序排序 = 按int64值排序，字典序二分查找 = 按int64值查找
- 求两个int64型a和b的公共前缀长度可以非常高效（下一页）
- 对于典型应用，N可以取很小，比如N = 10或者5
  - 甚至精确求前2N大LCP也不必，可以直接使用二分查找到的前后相邻串

不再做贪心推进，可能miss掉LCP大的串

# 求最高非0位

## int32 version

```
int msb32(unsigned int n)
{
    int res = 0;
    if (n & 0xffff0000) { res += 16; n &= 0xffff0000; }
    if (n & 0xff00ff00) { res += 8; n &= 0xff00ff00; }
    if (n & 0xf0f0f0f0) { res += 4; n &= 0xf0f0f0f0; }
    if (n & 0xcccccccc) { res += 2; n &= 0xcccccccc; }
    if (n & 0xaaaaaaaa) res += 1;
    return res;
}
```

# 作业五：Hamming空间近似最近邻查询

- http://cssyb.openjudge.cn/23hw5/
- Deadline: 4月12日（3月22日开始）

# * 参数（$n^{1/(1+\epsilon)}$）的选取 — 非正式讨论

**细节见Moses Charikar, STOC 2002**

SimHash发明人

- 考虑查询点$q \in \mathbb{H}^d$，设$q^*$是q的最近点，并设$\text{dist}_H(p, q) = r$
- 性质：存在一个整数$k \leq d$，使得
  - 存在一个排列$\sigma_i$，使得$q^*$的前k位与q相同
  - 考虑所有距q $\geq (1+\epsilon)r$的点，与q前k位相同的串<2N个（考虑所有排列）
- 观察：考虑一个随机位$i \in [d]$：

因此$q^*$会有排列前k个都相等

  - 与q距离近的点p有大概率$p_i = q_i$；与q距离远的点p有大概率$p_i \neq q_i$

因此 $\geq (1+\epsilon)r$距离的点在随机排列中前k与q都相等概率很小

# * Locality Sensitive Hashing (LSH)

## 统合MinHash与SimHash

- 称hash h是一种LSH，若$\Pr[h(x) = h(y)]$很大，则代表x和y相似
  - 我们希望相似的元素故意追求哈希冲突，即放到同样的bucket里
  - 对于不相似的元素避免哈希冲突，要放到不同的bucket里
- 验证：MinHash和SimHash都是LSH
  - MinHash h: $\Pr[h(S) = h(T)] = J(S, T)$
  - SimHash h: $\Pr[h(x) = h(y)] = 1 - \theta(x, y)/\pi$

# * 一般LSH to Hamming

因此只要有LSH就可以使用高效的最近邻查询！

## 以MinHash为例

实际中可以用各种非严格随机的哈希，或Universal Hashing

- 考虑一个映射到$\{0,1\}$的随机哈希$b: X \to \{0,1\}$
- 若$x \neq y$则$\Pr[b(x) = b(y)] = 0.5$，否则$\Pr[b(x) = b(y)] = 1$
- 考虑将b与h复合$b(h(\cdot))$，则$\Pr[b(h(A)) = b(h(B))] = \dfrac{1 + J(A,B)}{2}$
- 下面可以类似SimHash操作，得到Hamming空间的点
  - 找$T$组独立的h和b，T维二进制串$f(S) = (b^{(1)}(h^{(1)}(S)), \ldots, b^{(T)}(h^{(T)}(S))$即为Hamming表示

对应于“多次试验取平均值”

$(f(A) \oplus f(B))/T = 1 - J(A,B)$

# 工业界广泛应用

- The algorithm is used by the Google Crawler to find near duplicate pages.
- In 2007 Google reported using Minhash and LSH for Google News personalization
- In 2021 Google announced its intent to also use the algorithm in their newly created FLoC (Federated Learning of Cohorts) system

- Google索引了~50 Billion $= 5 \cdot 10^{10}$网页，每个网页只需要64bit，共占用 < 500G空间，在单台高端服务器都可以进行全网网页的相似度查询


- 我们介绍的这套LSH + Hamming最近邻方法获Paris Kanellakis Award

颁给具有重大实际影响力的理论成果