

程序设计实习（实验班-2023春）

课程总结、期末安排

授课教师：姜少峰

助教：张宇博 楼家宁

Email: shaofeng.jiang@pku.edu.cn

关于期末考试、平时分

时间地点

- 上机考试，闭卷，3小时
- 6月23日下午（默认）2点
- 机房位置在理科1号楼2层，具体地点之后发通知

题目构成

- 计划8 - 9题，按计划总分50（平时分是另50）
- 构成：
 - 4 - 5“常规”题：难度低 -> 中，关于基本算法（模拟、简单的递归、贪心等）
 - 2作业原题：只会从编程题出，不会从实验报告出
 - 1“常规”偏难题，1课上讲的算法的活学活用题
- 无C++面向对象编程有关的题目

关于期末评分

- 考前暂不规定特别详细的评分细则，但有一些预期：

可认为是一个充分条件
- 假定平时分得满分，则4 - 5道常规题 + 2道作业题都做出至少可以优秀
- 整个评分非线性，过题比较少的同学也不会有特别低的分数
- 最后85+这个分段的区分度主要在最后的两个难题上（例如做出一个90+）
- 最后会根据同学们考后总体表现调整分数
 - 倾向于放宽优秀率限制

关于平时分

- 根据计划：每次实验报告、每道必做编程题都算1个单位，大作业算5个单位
- 所有按照单位的比例平分50分平时分
 - 我们有2次实验报告 + 10道编程题目，1个大作业
 - 每次实验报告、题目 $50/17 \approx 2.94$ 分，大作业 ≈ 14.71 分
- 为了方便计算，我们最后给每次实验报告、题目3分，大作业14分

课程回顾、主线串讲

大主线

- 程序设计的科学性：现代算法（随机，近似，数据科学，大数据）
 - 哈希方法：CountMin, MinHash, SimHash
 - 低维数据：几何近似算法
 - 高维数据：降维、利用稀疏性
 - 大数据上的计算：亚线性模型（数据流，MPC等）
- 程序设计的工程性：C++的语言特性，面向对象的设计模式

随机算法

随机算法

- 如何衡量一个随机算法的性能?
 - 精度 vs 失败概率, 即 $\Pr[\text{ALG is correct}] \geq 1 - \delta$
- 随机算法的一般设计方法
 - 设计一个无偏估计, 即让算法的数学期望性能达到精度要求
 - 得到常数概率的成功率
 - 多次独立试验提高成功率

Median Trick

- 现在有一个黑盒能以 $p > 0.5$ 概率正确回答某个Yes/No问题的答案
- 如何将该概率强化成任意的 $1 - \delta$?
- 期望看到：若重复 T 次，正确答案会占多数，即超过 pT 个
- 算法：重复 T 次，选取占多数（或者处于中位）的答案
- T 选多大才够？

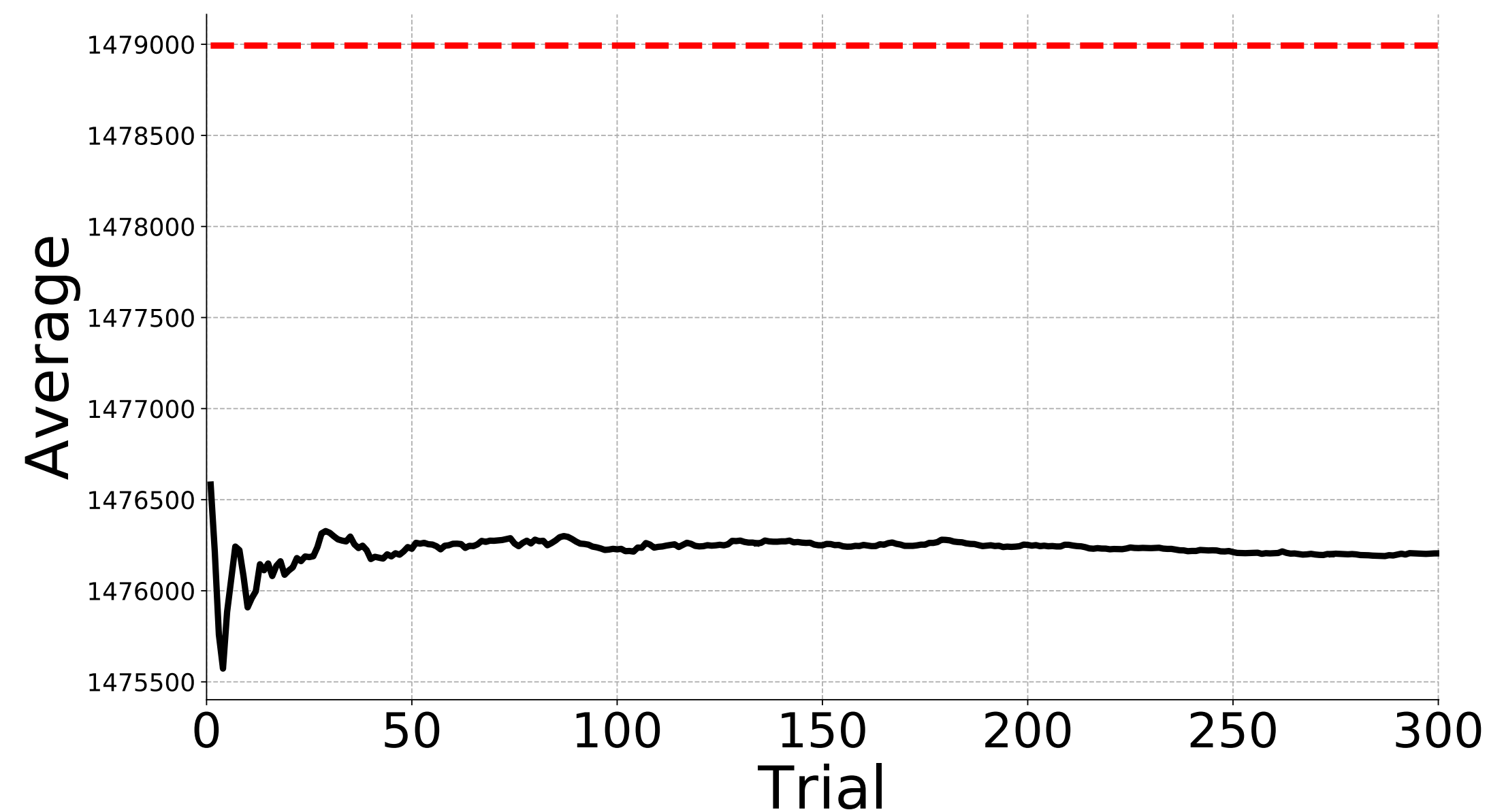
再论失败概率 δ

- 在之前的slides, 我们讨论了如何选取 δ
- 事实上: 常数 δ 就通常“不失一般性”
 - 经过 $O(\log n)$ 次重复, 可以达到 $1/\text{poly}(n)$ 的失败率
- 多大的常数?
 - 如max-cut等可以取最大来放大成功率的: 可以是任何常数
 - 如果需要用median trick, 则必须是 > 0.5 (注意严格不等号)

这对于多次运行也通常足够了, 毕竟一般只需要运行 $\text{poly}(n)$ 次

作业一：最大割随机近似算法实现、调优

- 作业一是一个实验报告（作业要求在教学网和课程网站）
- 给定测试数据，实现课上讲的最大割算法，并绘制重复次数-平均代价图



建议用python的matplotlib画图
作业题目里给出了画图的样例代码

快速测试矩阵相乘结果是否正确：Fingerprinting

- 给出三个 $n \times n$ 实矩阵 A, B, C ，测试是否 $AB = C$
- 确定性/暴力算法： $O(n^\omega)$ 时间，现在 $\omega \approx 2.37188$
- $O(n^2)$ 时间随机算法：如果 $AB = C$ 那么一定返回 Yes；但是 $AB \neq C$ 时可能答错
 - 随机选取一个 n 维随机 $\{0, 1\}$ 向量
 - 测试是否 $ABx = Cx$ ：是则输出 YES 否则 NO

达到 $\omega = 2$ 是重大 open question

可以在 n^2 时间计算 ABx ：先算 $y = Bx$ ，再算 Ay

作业二： 测试矩阵相乘是否正确

- 实现课上讲的随机检测方法
- 在openjudge评测，只有所有测试用例都通过才算通过
- 标程使用固定重复次数在设定时间内可以通过
- <http://cssyb.openjudge.cn/2023hw2/>

哈希方法

哈希方法

- Hash = 均匀映射，“最均匀”的是随机哈希
- 直接应用：随机负载均衡，互联网动态缓存算法consistent Hashing

Count-min Sketch

- Count-min sketch是一个数据结构，支持对于任何误差参数 $0 < \epsilon < 1$:
 - 插入 N 个 $[n]$ 上的元素后，给定一个 $x \in [n]$ ，估计 x 共出现了多少次
 - 具体来说：以大概率满足 $C \leq \hat{C} \leq C + \epsilon \cdot N$
 - 大 = $1 - 1/\text{poly}(n)$
 - “point query”
 - $\hat{C}$ 是估计量， C_x 是 x 插入次数
- 总共使用空间 $\frac{\text{poly log } n}{\epsilon}$ ，且单次查询和插入单个元素时间 $\frac{\text{poly log } n}{\epsilon}$

Count-min Sketch

大体思路

- 大致算法：

$m \ll n$ 是待定参数

- 构造一个随机哈希函数 $h : [n] \rightarrow [m]$
- 将输入元素均匀映射到 m 个 bucket，对每个 bucket $j \in [m]$ 记元素个数 $C[j]$
- 如果没冲突，那么最后对 $x \in [n]$ 查询输出 $C[h(x)]$ 就是精确解
- 冲突的影响：多个元素的 count 累加在一起被报告了出来

$m \ll n$ 因此必然有
(很多) 冲突

只会造成对结果的高估！

Count-mim Sketch: 完整算法

- 初始化:
 - 设置 $T = O(\log n)$ 个独立的随机哈希 $h^{(i)} : [n] \rightarrow [m]$, 这里 $m = 2/\epsilon$
 - 初始化 T 个 m 元 counter $C^{(i)}[1 \dots m] = 0 \quad (1 \leq i \leq T)$
- 插/删 $x \in [n]$ 时:

count-min也支持删除

 - 对每个 $i = 1, \dots, T$, 将 $C^{(i)}[h^{(i)}(x)]$ 增/减 1
- 查询 $x \in [n]$ 时: 返回 $\min_{1 \leq i \leq T} C^{(i)}[h^{(i)}(x)]$

应用：Count-min做数据流Heavy Hitter

至多有k个

- k-Heavy hitter (k-HH): A是长度N元素在[n]上的数组，求出现次数 $\geq N/k$ 的元素
- 应用
 - A是亚马逊/京东物品成交/访问记录，要快速得到当日最流行爆款商品
 - A是路由器上的IP访问日志，快速找到异常流量防DoS攻击
- 这些应用都是大数据/小内存，因此我们考虑数据流算法
- 数据流：数组A中元素以任意顺序以数据流方式给出，算法在流结束时给出k-HH

相关问题：返回top-k出现次数元素
k-HH一定在top-k，但是反之不然！

一般关注空间复杂度；
处理全流总时间一般追求 $npoly \log n$

数据流可以理解成无随机访问，类似于磁带

数据流近似k-HH：完整算法

- 输入参数：误差 $0 < \epsilon < 1$ ，整数 $k \geq 1$ ，输入元素取值范围 n
要求元素去重
- 初始化： $M = 0$ ，一个在 $[n]$ 上的 count-min 数据结构 $P(\epsilon)$ ，一个集合 L
- 在数据流中，当 $x \in [n]$ 被插入时：
重复元素会被去除
 - $M++$ ；将 x 插入 P ；若 $P.\text{count}(x) \geq M/k$ ，将 x 插入 L
 - 检查 L 中所有元素 y ，如果 $P.\text{count}(y) < M/k$ ，则把 y 从 L 删除
- 数据流结束时，返回 L

任何时候 L 中元素 $\text{count} \geq M/k - \epsilon M$ ，因此设

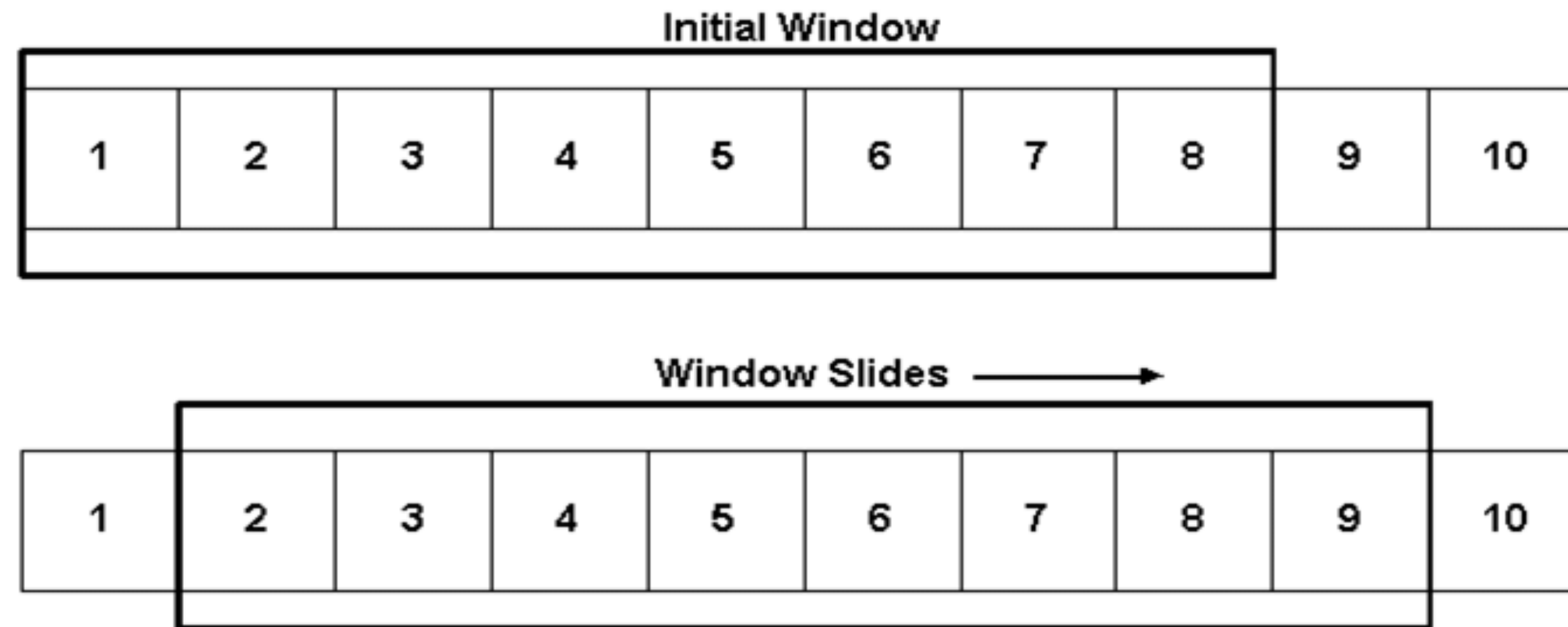
$$\epsilon \leq \frac{1}{2k}, \text{ 则有 } |L| = O(k)$$

作业三第一题： k-HH的数据流算法

- <http://cssyb.openjudge.cn/khh23/>
- 用刚刚介绍的基于count-min sketch的方法解决数据流近似k-HH问题
- 为了采用数据流输入和控制空间，此题为代码填空
- Deadline： 3月31日

字符串滚动哈希：Karp-Rabin方法

- 一种针对字符串的哈希，支持快速“sliding window”型更新



Karp-Rabin Hash

很多编程语言的字符串类型的哈希实现
(如Java)

- 为叙述简单, 考虑长度至多是 n 的binary string
- 设 q 是一个 (足够大) 的素数:

一般地, 需要以 $|\Sigma|$ 为底数 (Σ 是字母表)

$$h(S) := \sum_{i=1}^n S_i \cdot 2^{n-i} \bmod q$$

- 性质: $h(S_{i:j}) = h(S_{1:j}) - 2^{j-i}h(S_{1:i}) \bmod q$ $h(S \parallel T) = 2^{|T|}h(S) + h(T) \bmod q$

对 $1 \leq i \leq n$ 预处理 $h(S_{1:i})$ 即前缀
和后可以 $O(1)$ 时间求子串 h 值

对 $1 \leq i \leq n$ 预处理 $2^i \bmod q$ 后
可以 $O(1)$ 时间计算拼接

作业三第二题：二维字符串匹配

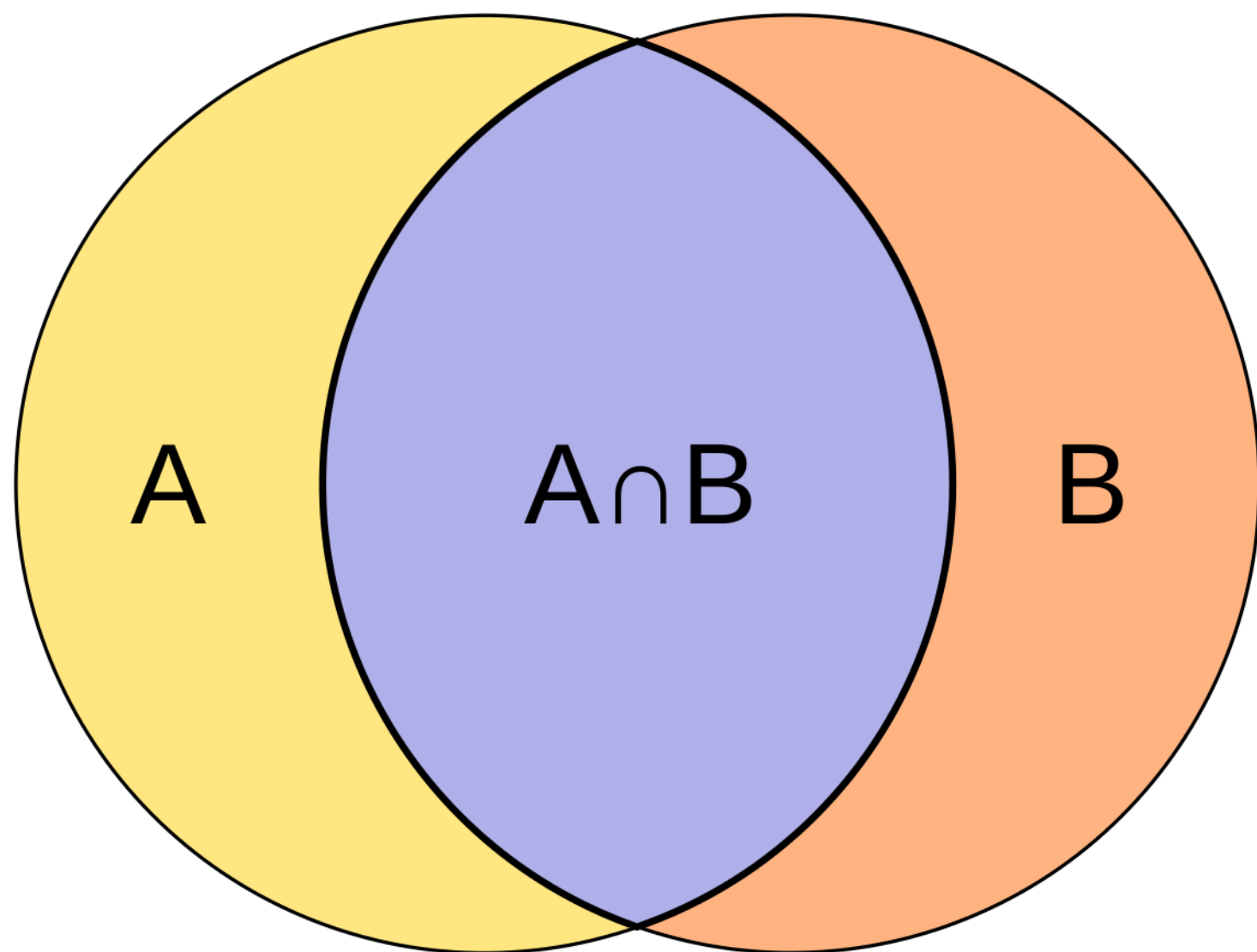
- <http://cssyb.openjudge.cn/khh23/>
- 如何扩展到二维
 - 将（子）矩阵的哈希值理解为按逐行拼接的字符串的哈希值
 - 子矩阵哈希值 $h(S_{p:q,r:s})$ 可以通过预处理“前缀和” $h(S_{1:i,1:j})$ 迅速求得
- deadline 3月31日

Jaccard Similarity

- 两个集合 $A, B \subseteq [n]$

是一个 $[0,1]$ 的数

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



MinHash

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

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

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）：

如何实现？

- 采用T个独立的随机哈希 $h^{(1)}, \dots, h^{(T)}$

- 对每个 A_i 和 $h^{(t)}$, 计算 $h_{\min}^{(t)}(A_i)$

$O(|A_i| \cdot T)$ 时间

计算的是 $h_{\min}^{(t)}$ 相等的t所占比例

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

$O(|T|)$ 时间

作业四：用MinHash近似Jaccard Similarity

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

Cosine Similarity

- 考虑两个向量 $x, y \in \mathbb{R}^d$, 定义 $\sigma(x, y) := \frac{\langle x, y \rangle}{\|x\|_2 \|y\|_2}$
 $\sigma(x, y) = \cos(\theta(x, y))$
- 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.: $\log(\text{总文档数} / \text{单词出现在多少文档})$
 - 对单词 w , $\text{TF-IDF}(w) = \text{TF}(w) * \text{IDF}(w)$, 做成一个下标是单词、值TF-IDF向量

一般非常的高维、稀疏

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

$\theta(x, y) \in [0, \pi]$ 是 x 和 y 的向量夹角

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!

- $f(x) := (h^{(1)}(x), \dots, h^{(T)}(x))$

- 上述估计量就等于 $(f(x) \oplus f(y))/T$

- 即： $\text{dist}_H(f(x), f(y)) \approx \theta(x, y)/\pi$

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

Hamming空间找近似最近邻

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

这里典型情况要求 $\epsilon \geq 1$

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

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）

低维：几何近似算法

Idea 1: 格点离散化, 线性时间时间 $(1 + \epsilon)$ -近似直径

T可通过选取任意点u, 求u到最远点的距离得到 (见第一讲)

即满足 $1/2 \cdot \text{diam}(P) \leq T \leq \text{diam}(P)$

- 先 $O(n)$ 时间找一个直径的2-近似值T
- 作 $\ell := \epsilon \cdot T / \sqrt{2}$ 的网格, 并round到中心
- 因此每个点移动了 $\leq \sqrt{2}\ell \leq \epsilon \cdot \text{diam}(P)$
- 因此新点集 P' 满足

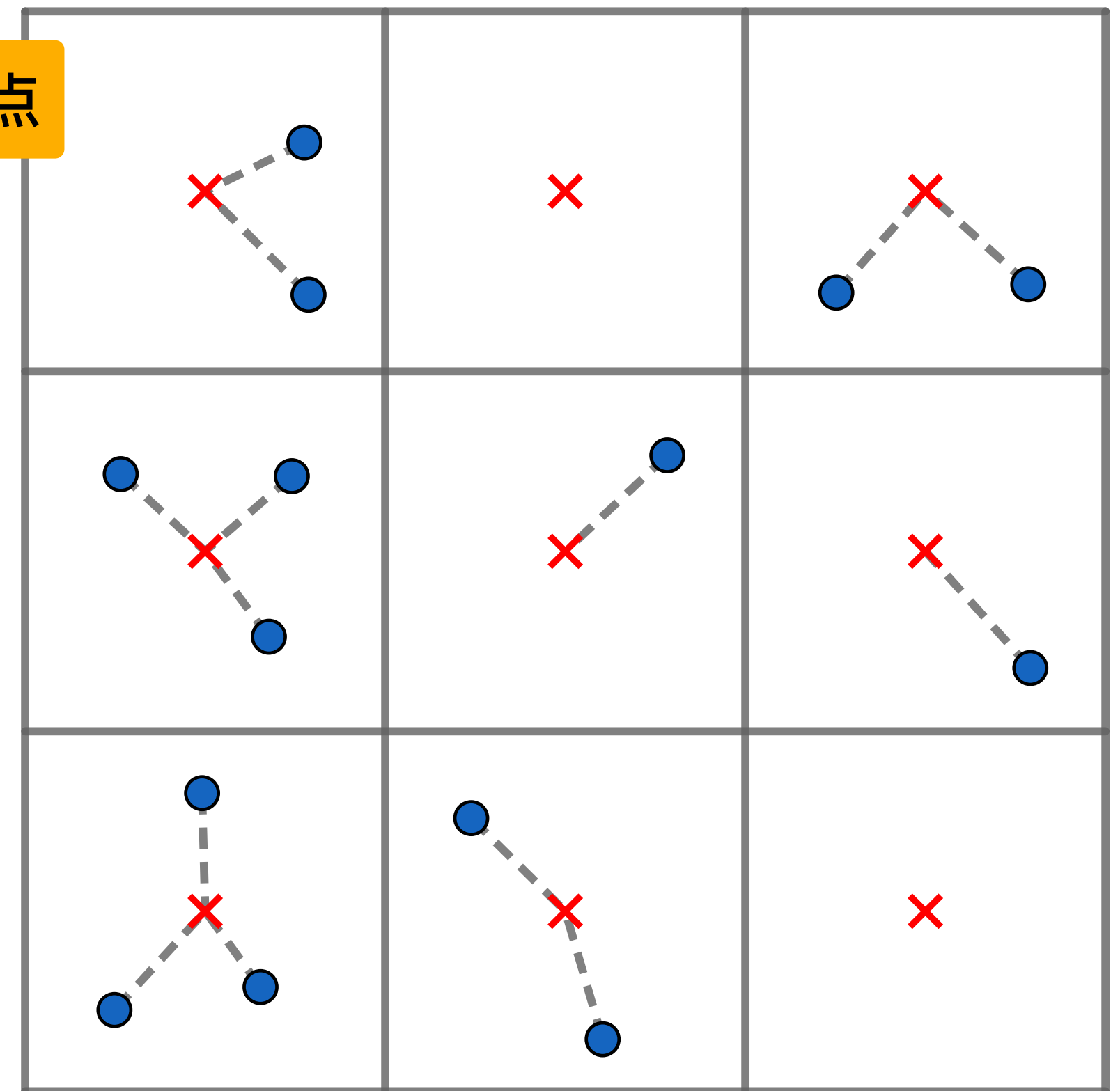
可在 $O(nd \log n)$ 时间构造 P'

$$\text{diam}(P') \in (1 \pm \epsilon) \cdot \text{diam}(P)$$

- 算法: 在 $|P'|$ 上暴力求直径, 复杂度 $|P'|^2 \cdot d$

P' 所有点都在 $\text{diam}(P') \times \text{diam}(P')$ 大方格内, 小方格 $\ell \geq \Omega(\epsilon \cdot \text{diam}(P))$, 故 $|P'| \leq (O(1/\epsilon))^2$

可以是任意确定点



总复杂度: $O(nd \log n) + \epsilon^{-O(d)}$

作业：近似求欧氏点集直径

- <http://cssyb.openjudge.cn/23hw6/>
- Deadline: 4月12日

Idea 2: 递归格点离散化, 四分树划分

以二维为例

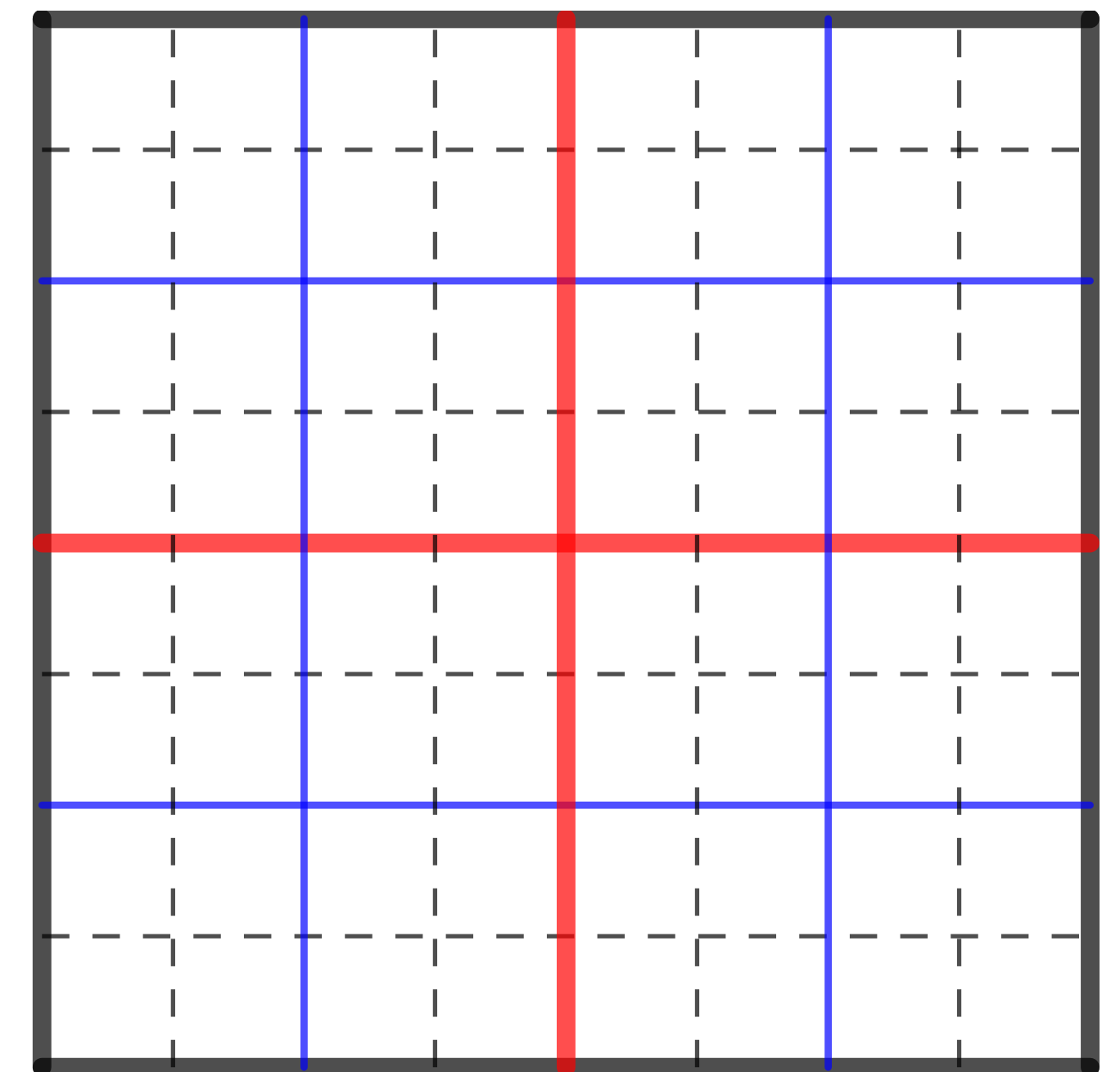
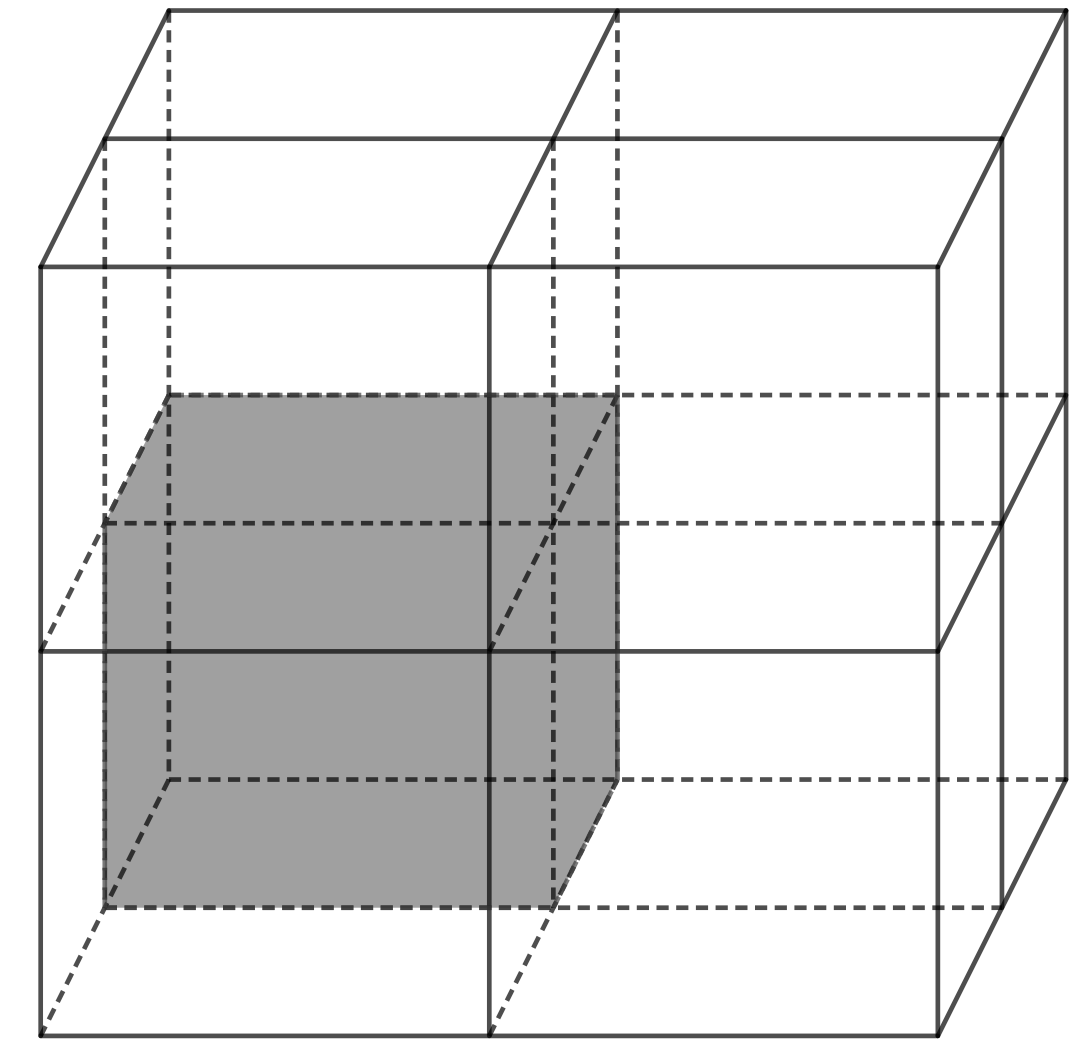
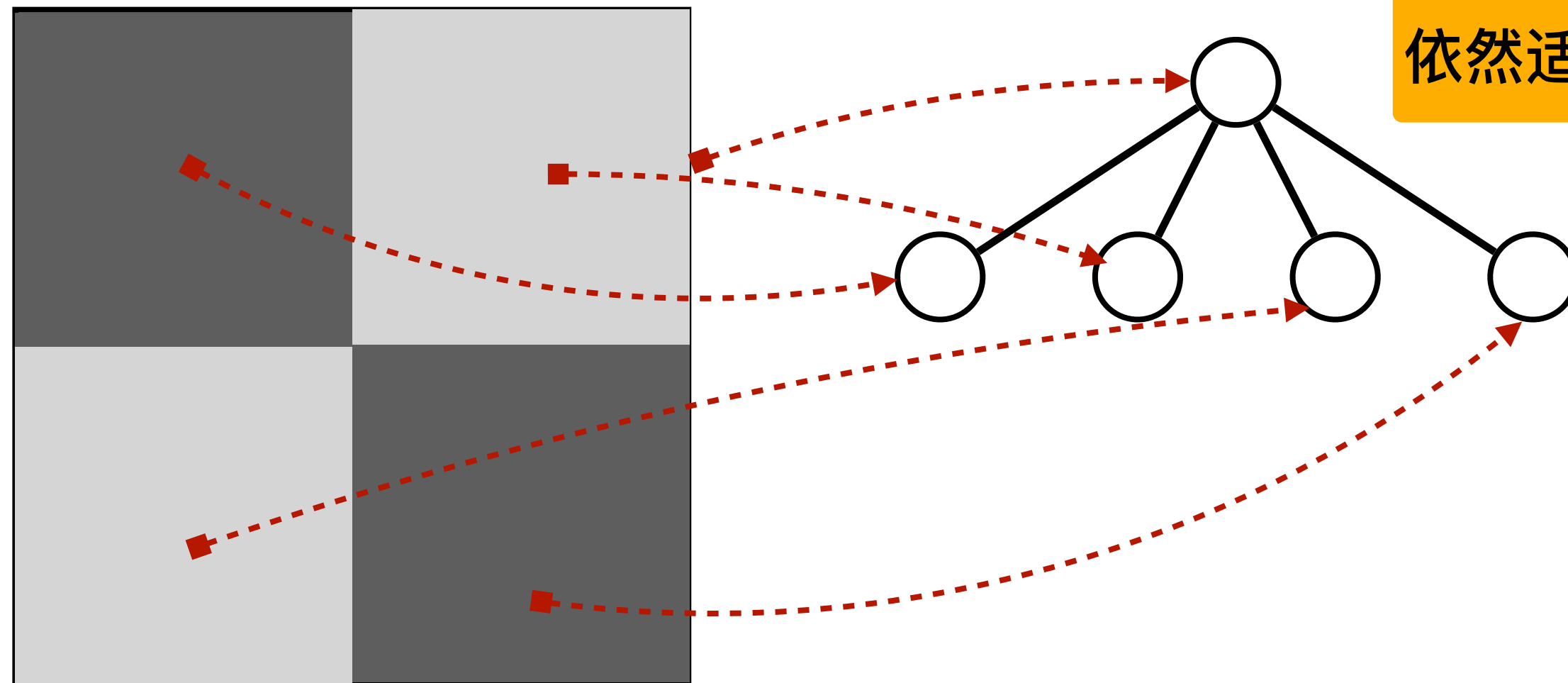
d维叫“d维四分树”

d维是从 $[\Delta]^d$ 划分

- 从最大的 $\Delta \times \Delta$ 的正方形开始
- 递归将当前 $\square$ 等分成4个边长为一半的 $\square$
- 组织成树形结构, 直到只含单点停止, 至多 $\log_2 \Delta$ 层

d维分成 2^d 个边长为一半的 $\square$

依然适用于d维



伪代码

d维

全局调用 $\square = [\Delta]^d$

BuildTree($\square$, P)

两个停止情况：不含数据点直接返回空；只含1个点返回当前 $\square$

if ($P \cap \square = \emptyset$) return NULL

let T.root = ($\square$, $\square \cap P$)

除了记录 $\square$ ，也要记录 $\square$ 里含有的数据点集

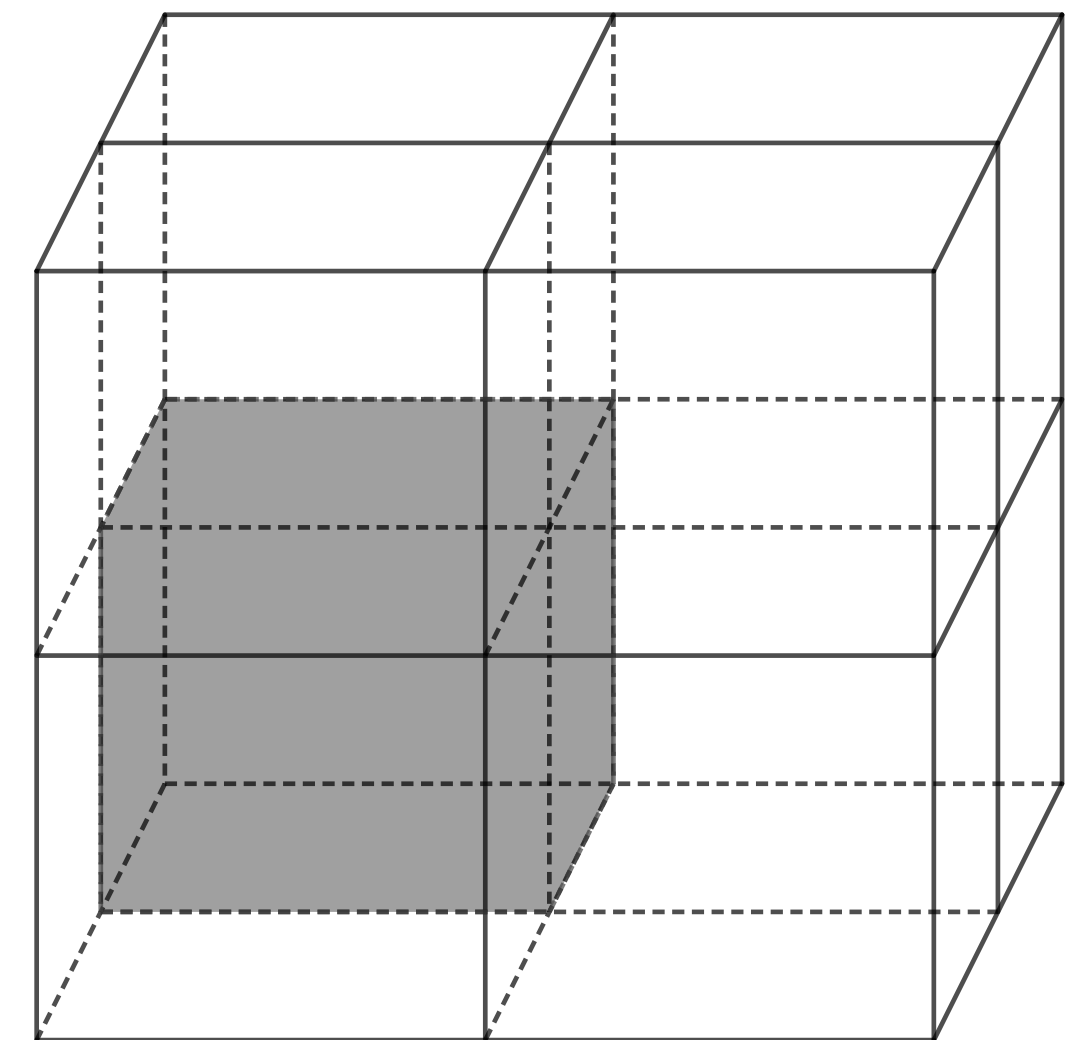
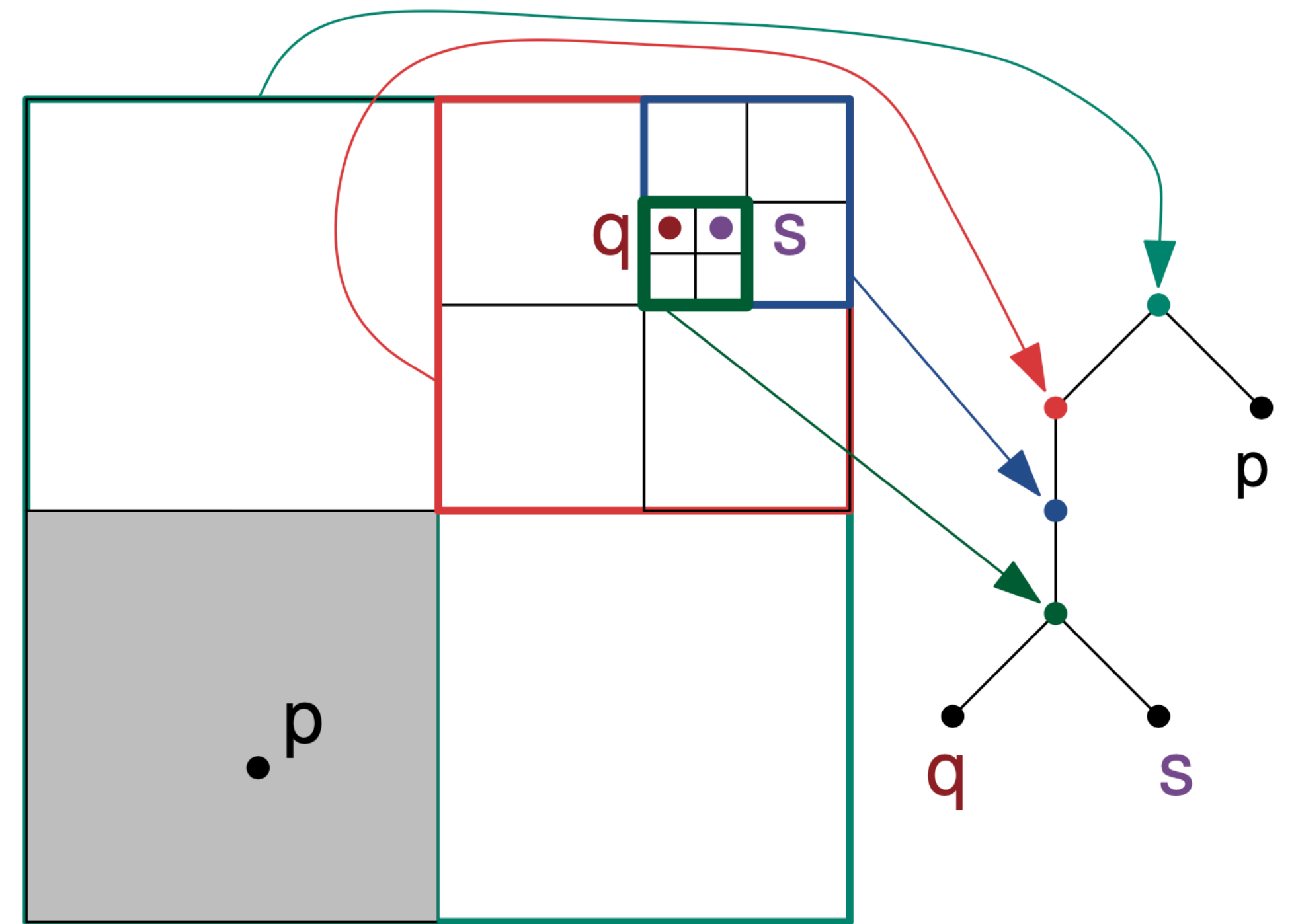
if ($|P \cap \square| = 1$) return T

evenly divide $\square$ into 2^d sub-squares $\square_1, \dots, \square_{2^d}$

for $i = 1, \dots, 2^d$, let T.child $_i$ = BuildTree($\square_i$, $P \cap \square_i$)

return T

整个算法只会产生 $O(n \log \Delta)$ 个非空 $\square$ (非空即 $\square \cap P \neq \emptyset$)



利用四分树进行 ϵ -近似最近邻查询

- 问题：

- 对输入点集 $P \subseteq [\Delta]^d$ 进行预处理

即 $\hat{q}$ 到 q 的距离是 q 到 q 在 P 最近邻距离的 $(1 + \epsilon)$ 倍

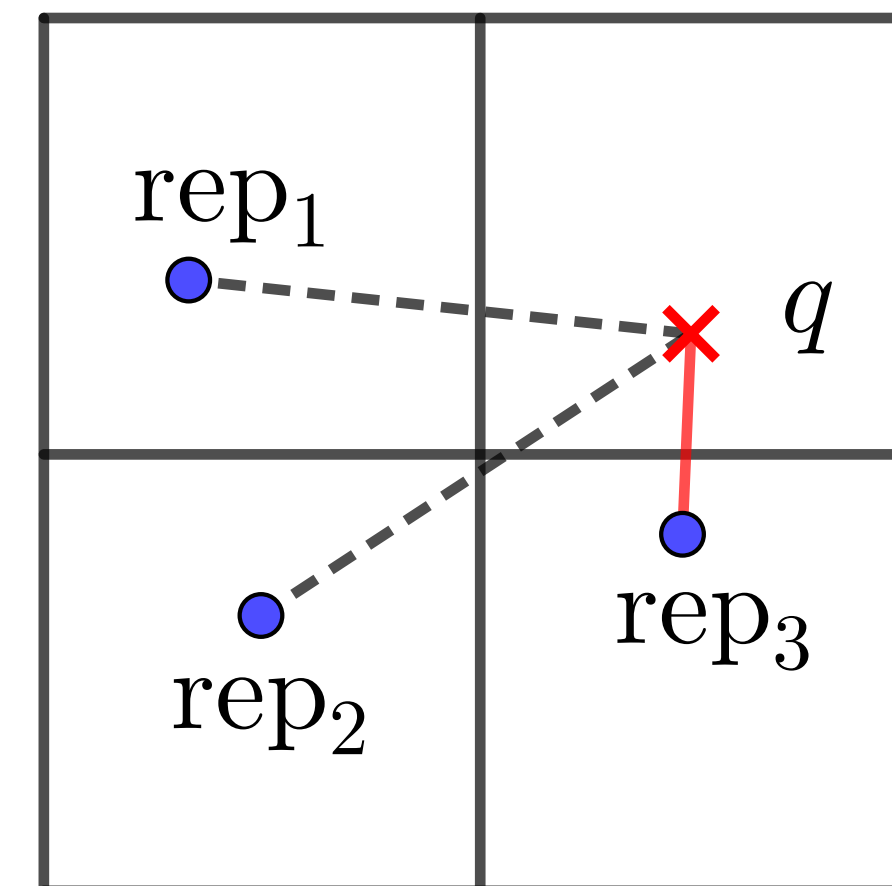
- 对任意 $q \in [\Delta]^d$ 回答 $\hat{q} \in P$, 使得 $\|\hat{q} - q\| \leq (1 + \epsilon) \cdot \min_{q' \in P} \|q' - q\|$

- 预处理：BuildTree($\square$, P), 这里调用 $\square = [\Delta]^d$

- 我们将给出一个 $O(\epsilon^{-d} \log \Delta)$ 时间的查询算法返回满足要求的 $\hat{q}$

完整算法

输入：查询点 q ，误差参数 $0 < \epsilon < 1$ ；查询所需时间 $O(\epsilon^{-d} \log \Delta)$



设集合 A_i 为需要检查的四分树第 i 层的格子的集合 ($0 \leq i \leq \log \Delta$)

初始 A_0 放入最大格子 $\square = [\Delta]^d$, $A_i = \emptyset$ ($1 \leq i \leq \log \Delta$), $r = \infty$, $\hat{q} = \text{NULL}$

for $i = 1, \dots, \log \Delta$

i 是层数，按层进行迭代

$\hat{q}$ 是当前解， r 是 $\|q - \hat{q}\|$

设 S 为所有 A_{i-1} 的格子的 2^d 个子 $\square$ 的集合

更新当前解为本层最近的rep

对每个 $\square_w \in S$ ，若 $\|q - rep_w\| < r$ ，更新 $r = \|q - rep_w\|$ 及 $\hat{q} = rep_w$

对每个 $\square_w \in S$ ，若 $r > (1 + \epsilon) \cdot (\|q - rep_w\| - \text{diam}(\square_w))$ ，将 $\square_w$ 放入 A_i

此处是“需要放入 $\square_w$ ”的判据，因此是 $>$

返回 $\hat{q}$

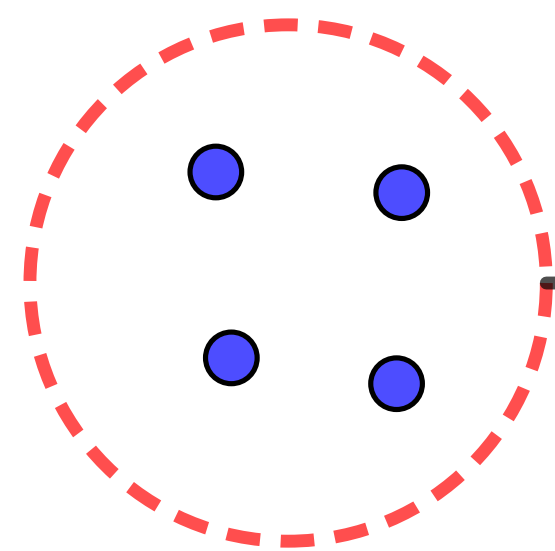
作业：欧氏空间近似最近邻查询

- <http://cssyb.openjudge.cn/23hw7/>
- Deadline: 4月19日

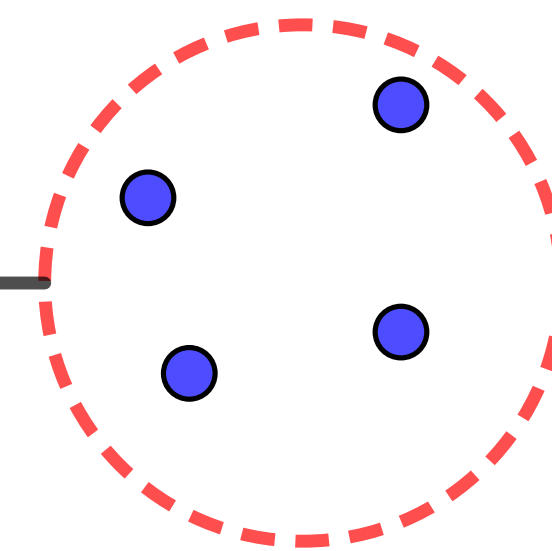
Idea 3: 考虑点对的离散化, WSPD

直观讨论

- 输入点集 $P \subseteq [\Delta]^d$, 设 $|P| = n$
- 考虑所有点对的距离 (共有 $O(n^2)$ 对), 试图将点对距离离散化
- 直觉: 设有 $S, T \subseteq P$ 两组点, 且他们距离很远
 - 那么可以把 S 和 T 分别“缩点”, 从而将 $S \times T$ 上的距离等于同一个值



即使不显式缩点, 左右两边点的距离相对来说都差不多; 例如左右两圆直径=1, 但相距100



WSPD

定义

- 一个 ϵ -WSPD是一系列 $\{\{A_i, B_i\}\}_i$, 使得

点集之间的距离等于最近点的距离, 即

$$\text{dist}(S, T) := \min_{x \in S, y \in T} \|x - y\|$$

- $\forall i, A_i, B_i \subseteq P$ 且 $\max(\text{diam}(A_i), \text{diam}(B_i)) \leq \epsilon \cdot \text{dist}(A_i, B_i)$ 即两组点 A_i, B_i 距离很远

- $\bigcup_i A_i \times B_i = P \times P$, 即任何 $p \neq q \in P$ 可以找到某个 i 使得 $p \in A_i, q \in B_i$

可以理解为对 $O(n^2)$ 个距离对的划分



基于四分树的WSPD：完整算法

WSPD用四分树的 $\square$ 表示，总复杂度 $O(n\epsilon^{-d} \log \Delta)$ ，所得WSPD有 $O(n\epsilon^{-d} \log \Delta)$ 项

全局对 $\Delta = [\Delta]^d$ 调用WSPD($\square, \square$)

WSPD($\square_u, \square_v$)

定义： $\delta(\square_u) = \text{diam}(\square_u)$ 若 $|\square_u \cap P| \geq 2$ ，否则 $\delta(\square_u) = 0$

if ($u = v$ and $\delta(\square_u) = 0$) return // 单点 $\square$ 不要配对到自己身上

if ($\delta(\square_u) < \delta(\square_v)$) swap u, v

if ($\delta(\square_u) \leq \epsilon \cdot \text{dist}(\square_u, \square_v)$)

直接看 $\square$ 而不是 $P \cap \square$

return $\{ \{ \square_u, \square_v \} \}$

运行至此， $\delta(\square_u) \geq \delta(\square_v)$ ，因此 $\{ \square_u, \square_v \}$ 确实满足WSPD条件，又因为算法会尝试一直递归到底，因此**确保最后输出的是合法WSPD**

return $\bigcup_{i=1}^{2^d} \text{WSPD}(\square_{u_i}, v)$

$\square_{u_i}$ 是 $\square_u$ 的子 $\square$

WSPD应用：精确求最近点对

- 这是一个 $O(n \cdot 2^d \log \Delta)$ 时间的精确算法：
 - 构造 $\epsilon = 0.5$ 的WSPD $\{\{A_i, B_i\}\}_i$ $O(n \cdot 2^d \log \Delta)$ 时间
 - 对所有的 $\{A_i, B_i\}$ ，若 $|A_i| = |B_i| = 1$ ，记录 A_i 和 B_i 中唯一点的距离
 - 在所有距离中的最近点对就是答案 $O(n \cdot 2^d \log \Delta)$ 时间
- 原因：设最近点对是 (p, q) 且 $p \in A_i$ ， $q \in B_i$ ，则 $|A_i| = |B_i| = 1$

因此算法不会
遗漏最近点对

若有多余点 x 在 A_i ，则容易看出 $\|p - x\| < \|p - q\|$ ，矛盾



作业：最近点对精确求解

- <http://cssyb.openjudge.cn/23hw8/>
- Deadline: 4月19日

降维：
应对Curse of Dimensionality

第一类降维： JL Transform

概述

[Johnson-Lindenstrauss, 1984]; 简称JL

JL是一种一般的降维方法，保n个 $\mathbb{R}^d$ 上的数据点集 P 中任何两点之间的距离

给定误差 ϵ ，JL是一个映射 $f: \mathbb{R}^d \rightarrow \mathbb{R}^m$ ，这里 $m = O(\epsilon^{-2} \log n)$

- 使得： $\forall x, y \in P, \|f(x) - f(y)\|_2 \in (1 \pm \epsilon) \cdot \|x - y\|_2$; $O(dm)$ 时间可算 $f(x)$

m 决定了降维的维度，称为target dimension，是降维的最重要性能指标

相对误差 ϵ ，是“最好”的近似比
可直接用于很多与距离有关的问题
以得到改进的近似算法

- 输入的 d 可大可小，目标维度只与 $\log n$ 和 ϵ^{-1} 有关

尤其对于例如 $d = n$ 时特别有效

这说明：对于近似算法来说，“高维” = $O(\log n)$ 维

应用：快速Linear Regression

线性回归定义

一般考虑 $n \gg d$, n 是数据点个数, d 是 feature 个数

- 输入: $(x_1, y_1), \dots, (x_n, y_n)$, 其中每个 $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$
y 是一个“label”, 即 x 对应的“结果”

- 要求: 找到一个 $w \in \mathbb{R}^d$ 使得 $\sum_i (\langle x_i, w \rangle - y_i)^2$ 即最小二乘误差最小化

- 解释: 想要找到一个 y 和 x 之间的线性关系

即希望 X 与 Y 线性相关

- 即设 x_i, y_i 分别是来自某个 X, Y 分布上采样的, 那么希望有 $Y = w_0 + \sum_{j=1}^d w_j \cdot X_j$

这里 w_0 项是“常数项”, 但可以不单独写在求和外面, 因为可以给 X 增加第 0 维, 取值恒等于 1, 并把 w_0 可看作是 $w_0 X_0$

Linear Regression求解公式

- 将输入 $(x_1, y_1), \dots, (x_n, y_n)$ 写成矩阵 $X \in \mathbb{R}^{n \times d}$ 第 i 行是 x_i , 向量 $y := (y_1, \dots, y_n)$
- 那么问题可以写作 $\arg \min_{w \in \mathbb{R}^d} \|Xw - y\|^2$
 - 这里 $(X^\top X)^{-1}$ 未必存在, 存在条件是 X 的 d 列线性独立, 即 d 个feature是独立的
- 一般公式: 最优的 $w^* = (X^\top X)^{-1} X^\top y$
 - 也是通过对每个 w_i 求导后令导数 = 0得到方程, 然后这是方程的解
- 什么复杂度? $X^\top X$ 需要 $O(nd^2)$, 是主要复杂度项 (另外, 求逆需要 $O(d^3)$)

加速方法：Subspace版本的JL

这是因为 Xw 中 w 是 d 维的

- 特殊性：需要考虑的 $v = Xw - y$ 虽然是 n 维，但rank/独立变量数/自由度只有 d
- 设 $\mathcal{U}$ 是 $\mathbb{R}^n$ 的一个 d 维的子空间

在我们应用中是所有 $w \in \mathbb{R}^d$ 在变换 Xw 下的集合，即 $\{Xw : w \in \mathbb{R}^d\}$ 可以（不失一般性）看作是 n 维空间只有前 d 个维度非0的子集

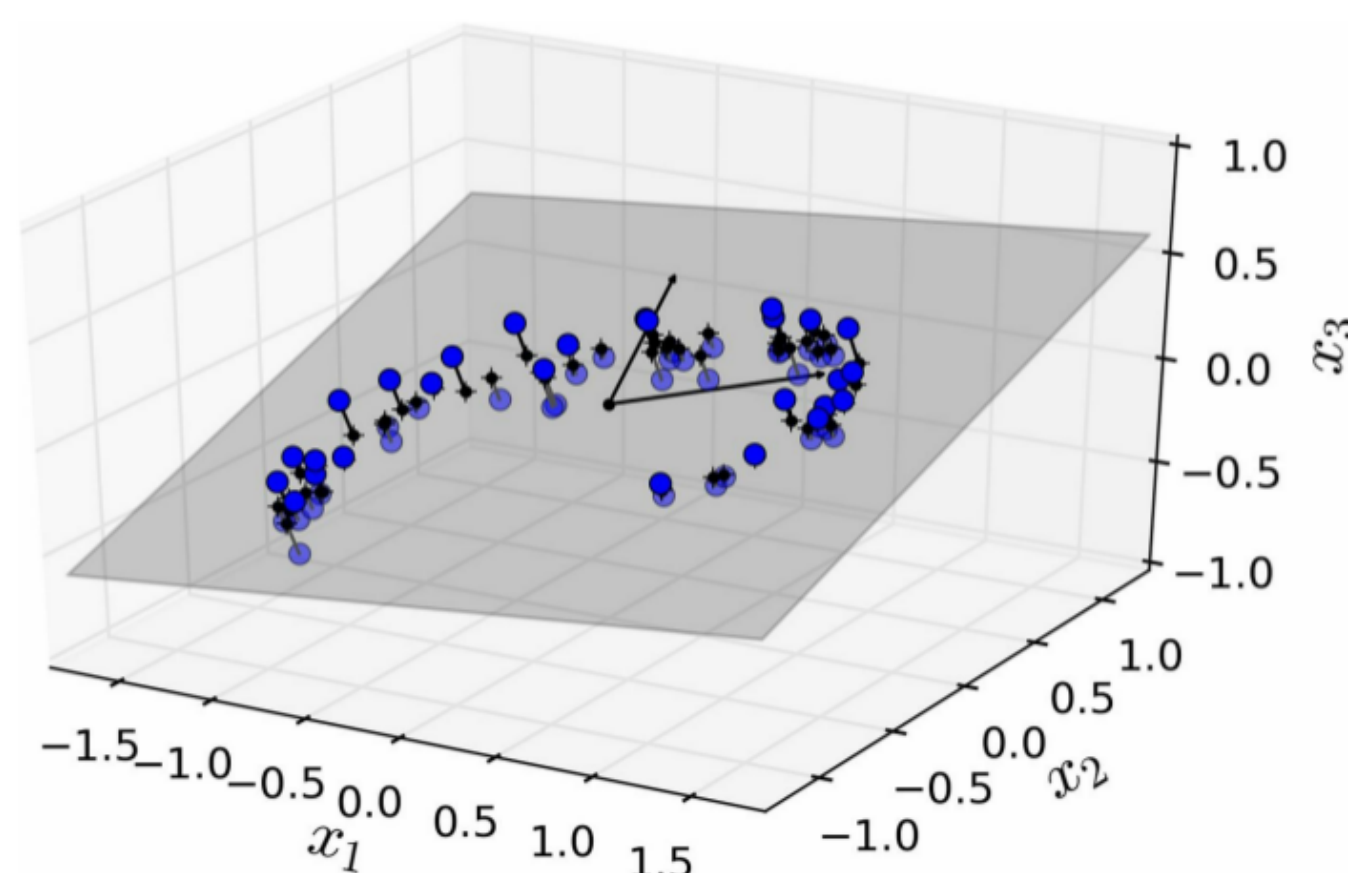
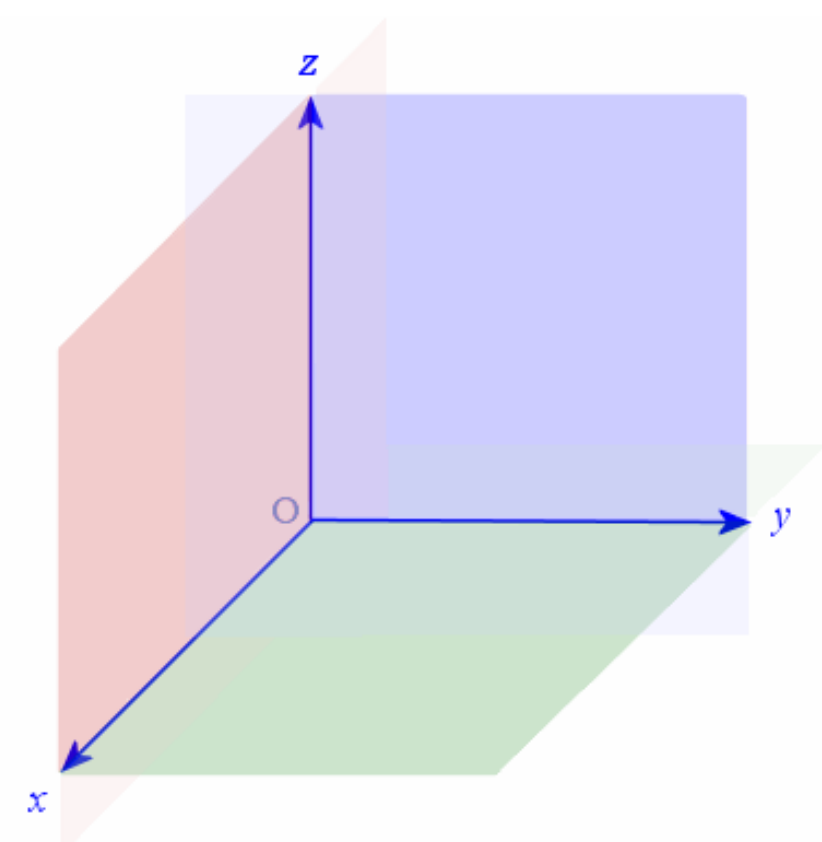


Figure 8-2. A 3D dataset lying close to a 2D subspace

例如 $\mathbb{R}^3$ 的x-y平面就是一个2维子空间，并且任意 $\mathbb{R}^3$ 上的2维子空间都可以通过旋转平移得到x-y平面

$Xw - y$ 可等价写作 $X'w'$ ： X' 和 w' 在 d 维基础上加一维

$$X' = \begin{bmatrix} X_{11}, \dots, X_{1d} & y_1 \\ \vdots & \ddots & \vdots & \vdots \\ X_{n1}, \dots, X_{nd} & y_n \end{bmatrix} \quad w' = \begin{bmatrix} w_1 \\ \vdots \\ w_d \\ -1 \end{bmatrix}$$

Subspace版本的JL

- 设 $\mathcal{U}$ 是 $\mathbb{R}^n$ 的一个 d 维的子空间

回忆: A 是 $\frac{1}{\sqrt{m}} \cdot (w_1, \dots, w_m)$, 其中每个 w_i 是 n 维高斯或者归一化的 $\{-1, 1\}^n$ 独立向量

定理: $m = O\left(\frac{d \log(\epsilon^{-1}) + \log(1/\delta)}{\epsilon^2}\right)$ 的JL矩阵 $A \in \mathbb{R}^{m \times n}$ 满足:

将 ϵ 和 δ 看作常数, 这基本上就是 $m = O(d)$

$$\forall v \in \mathcal{U}, \quad \Pr[\|Av\|^2 \in (1 \pm \epsilon) \cdot \|v\|^2] \geq 1 - \delta$$

- 如果把这样的 m 代入, 就可将regression复杂度从 $O(nd^2)$ 降到 $O(d^3)$

也就是说: 再大的 n 也可以规约到 $n = d$ 的规模解出来!

高效求解linear regression：完整算法

- 根据刚刚提到的方法生成每列只有一个非0元素的 $m \times n$ 矩阵 S
- 预处理/计算 $A'X$ 和 $A'y$ ，这里 $A' = S$

m可以灵活选取，未必按照理论上界
但注意：需要 $m \geq d$
- 在 $\hat{X} = A'X$ 和 $\hat{y} = A'y$ 的新输入上利用 $w = (\hat{X}^T \hat{X})^{-1} \hat{X}^T \hat{y}$ 公式计算近似最优解 w
- 输出 w

需要采用高斯消元

作业九： 利用subspace版本JL高效求解regression

- <http://cssyb.openjudge.cn/23hw9/>
- 截止日期2023年6月1日

第二类降维：PCA/SVD，目标函数

假设均值0且每列归一

输入 $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^d$ ，求 orthonormal 的 m 个向量 $\mathbf{v}_1, \dots, \mathbf{v}_m \in \mathbb{R}^d$ ，最大化

$$\frac{1}{n} \sum_{i=1}^n \underbrace{\sum_{j=1}^m \langle \mathbf{x}_i, \mathbf{v}_j \rangle^2}_{\text{squared proj. length}}$$

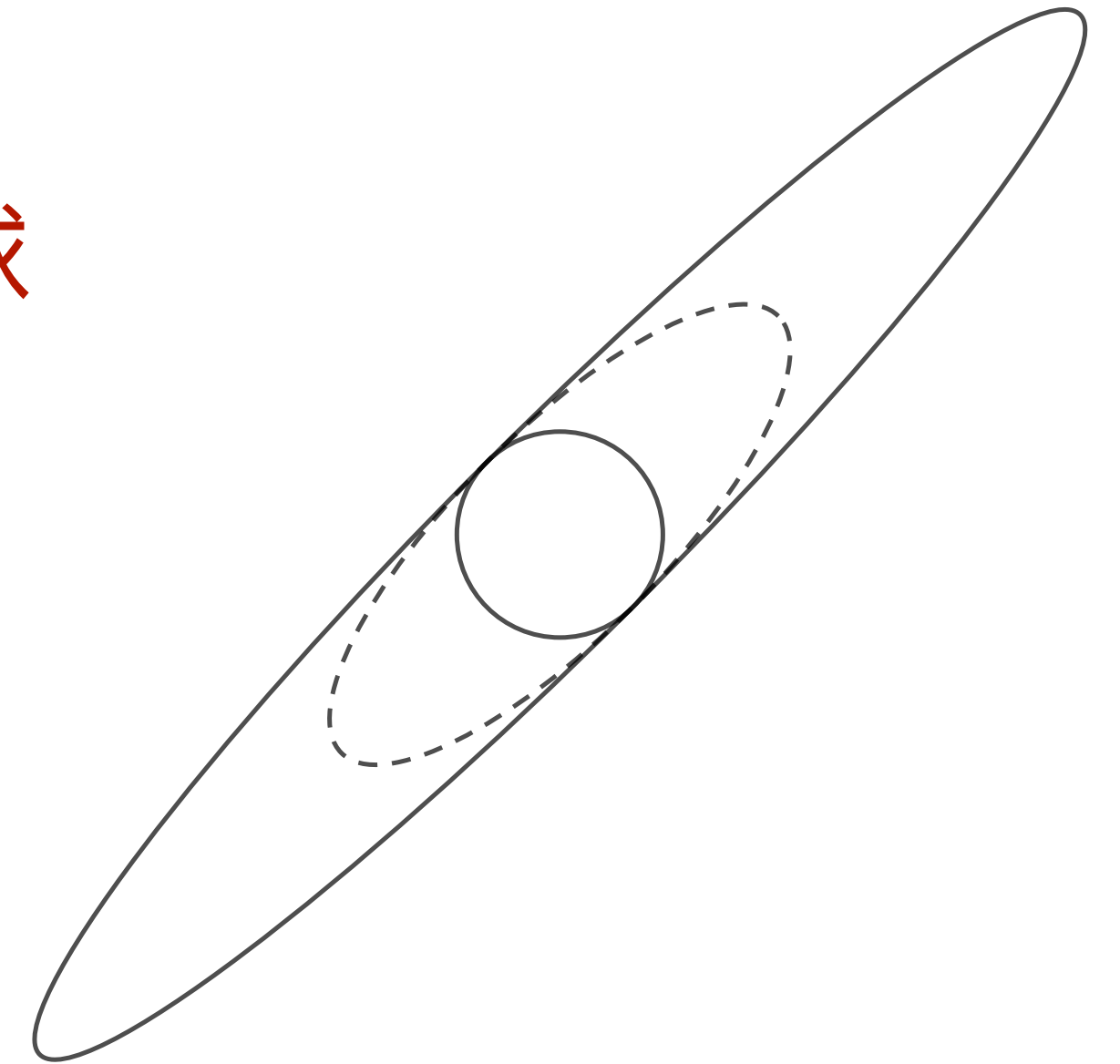
最优的这 m 个向量叫做 $\mathbf{x}_1, \dots, \mathbf{x}_n$ 的前 m 个 principle components

近似求Top Principle Component: Power Iteration

算法设计思路

- 回忆: $\mathbf{A} = \mathbf{X}^T \mathbf{X} = \mathbf{Q} \mathbf{D} \mathbf{Q}^T$ 本质上是将单位球映射到椭球
 - 椭球的最长轴就对应top principle component
- 任取一个向量 $\mathbf{u}$, 只要不与 $\mathbf{Qe}_1$ 垂直, 那么反复应用 $\mathbf{A}$ 后:
 - 将在 $\mathbf{Qe}_1$ 方向反复被拉伸
 - 若假设最长轴确实比其他轴长, 则最后 $\mathbf{u}$ 会非常贴近于 $\mathbf{Qe}_1$ 的方向

$\mathbf{Qe}_1$



如果看椭球, 则会在 $\mathbf{Qe}_1$ 上拉的非常长, 总体非常扁

算法

输入: $\mathbf{A} = \mathbf{X}^\top \mathbf{X}$

可以每一维独立 $\mathcal{N}(0,1)$ 后归一化生成

选取一个随机方向向量 $\mathbf{u}_0$

For $i = 1, 2, \dots$

$$\mathbf{u}_i := \mathbf{A}^i \mathbf{u}_0$$

根据实际情况设定“ $\approx$ ”的标准

如果 $\mathbf{u}_i / \|\mathbf{u}_i\| \approx \mathbf{u}_{i-1} / \|\mathbf{u}_{i-1}\|$ 则停止并返回 $\mathbf{u}_i / \|\mathbf{u}_i\|$

作业十： 利用Power Iteration求Top PC

- <http://cssyb.openjudge.cn/23hw10/>
- 截止日期： 2023年6月8日23:59

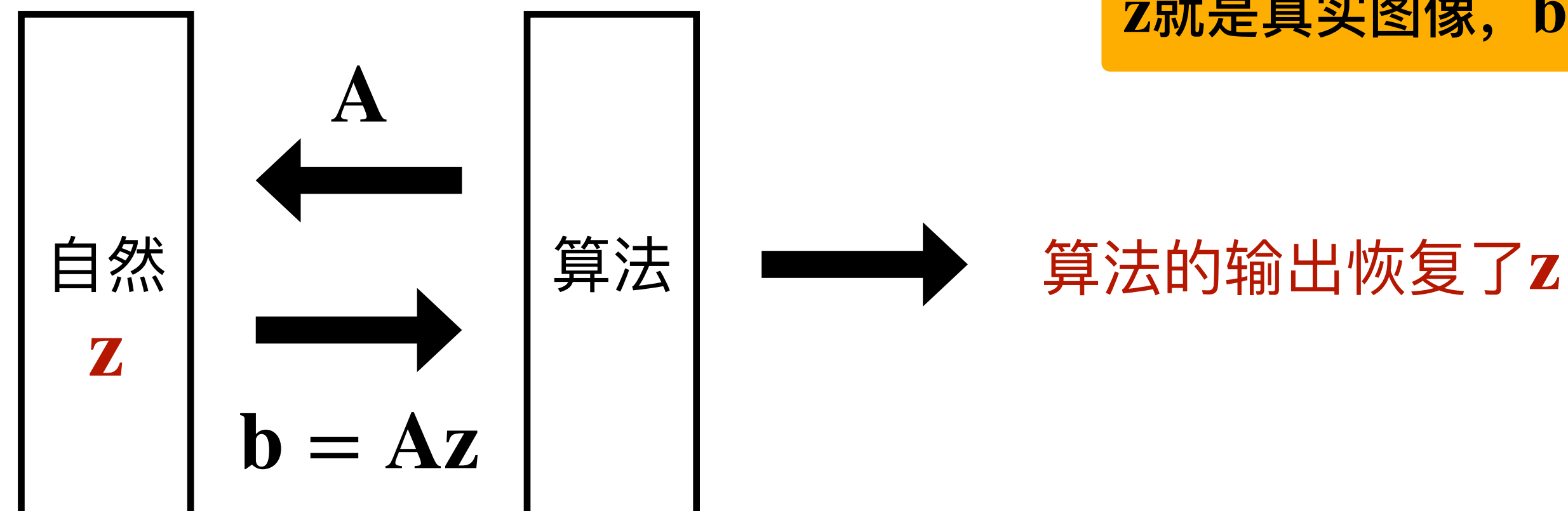
稀疏性

压缩感知具体设定

- 我们设计 m 个linear measurements（线性测量） $\mathbf{a}_1, \dots, \mathbf{a}_m \in \mathbb{R}^n$
- “自然”生成一个未知的信号 $\mathbf{z} \in \mathbb{R}^n$
- 某种硬件告诉我们信号在线性测量上的值 $\mathbf{b}$, $b_1 = \langle \mathbf{a}_1, \mathbf{z} \rangle, \dots, b_m = \langle \mathbf{a}_m, \mathbf{z} \rangle$
- 仅从 $\mathbf{b}$ 恢复 $\mathbf{z}$

目标是要同一组测量 $\mathbf{a}_1, \dots, \mathbf{a}_m$,
对“所有”可能的 $\mathbf{z}$ 都要有效恢复

想象设计一个影像传感器（摄像头）
 $\mathbf{z}$ 就是真实图像， $\mathbf{b}$ 就是传感器给出的测量值



压缩感知定理：精确版

定理：设 $\mathbf{A} \in \mathbb{R}^{m \times n}$ 其中 $m = O(k \log(n/k))$ 且 $\mathbf{A}$ 的每个元素都从 $\mathcal{N}(0,1)$ 独立采样。则 $\mathbf{A}$ 以高概率，同时对所有的 n 维的 k -稀疏的 $\mathbf{z}$ ，下面问题的唯一最优解恰好等于 $\mathbf{z}$ ：

$$\begin{aligned} \min \quad & \|\mathbf{x}\|_1 \\ \text{s.t.} \quad & \mathbf{Ax} = \mathbf{b} \end{aligned}$$

一般情况：如何处理绝对值

添加辅助变量 y 来刻画“绝对值”，改写成

$$\begin{aligned} \min \quad & \sum_i y_i \\ \text{s.t.} \quad & \mathbf{Ax} = \mathbf{b} \\ & y_i - x_i \geq 0, \quad \forall i \\ & y_i + x_i \geq 0, \quad \forall i \end{aligned}$$

否则若 $y_i > |x_i|$ 则可以降低 y_i 到 $y_i = |x_i|$ ，不影响其他任何变量，但降低了目标值

在这个新LP中：对每个 i ，有 $y_i \geq |x_i|$ ，并且由于目标是min，必然有 $y_i = |x_i|$

基于压缩感知的Sparse Recovery：完整算法

- 预先给定的参数： k , n , 代表信号是 n 维的 k -稀疏的实向量
- 算法先生成一个 $m \times n$ 的随机高斯矩阵 $\mathbf{A}$, 其中 $m = O(k \log(n/k))$
- 算法得到 $\mathbf{b} \in \mathbb{R}^m$, 使得 $\mathbf{b} = \mathbf{A}\mathbf{z}$
- 算法继续计算右边LP的最优解 $\mathbf{x}$ 作为输出

每一维都是独立标准正态 $\mathcal{N}(0,1)$

压缩感知定理告诉我们大概率有 $\mathbf{x} = \mathbf{z}$

$$\begin{aligned} \min \quad & \sum_i y_i \\ \text{s.t.} \quad & \mathbf{Ax} = \mathbf{b} \\ & y_i - x_i \geq 0, \quad \forall i \\ & y_i + x_i \geq 0, \quad \forall i \end{aligned}$$

作业十一：压缩感知实验报告

- 本次作业为实验报告，具体要求请见下面链接中的PDF文档：

<https://disk.pku.edu.cn:443/link/A36D6D818A27DDA8C44D7E0FF2090C11>

- 作业需要调用一个解LP的算法
 - 我们在作业描述的文档中提供了一个简单的单纯形法的实现，你可以参考
 - 也可以使用类库（尤其是你打算使用其他语言），但请注明使用了什么类库
- 截止日期：2023年6月15日

本次实验报告不限制编程语言

**感谢同学们一学期以来的支持！
祝同学们学业进步、前程似锦！**