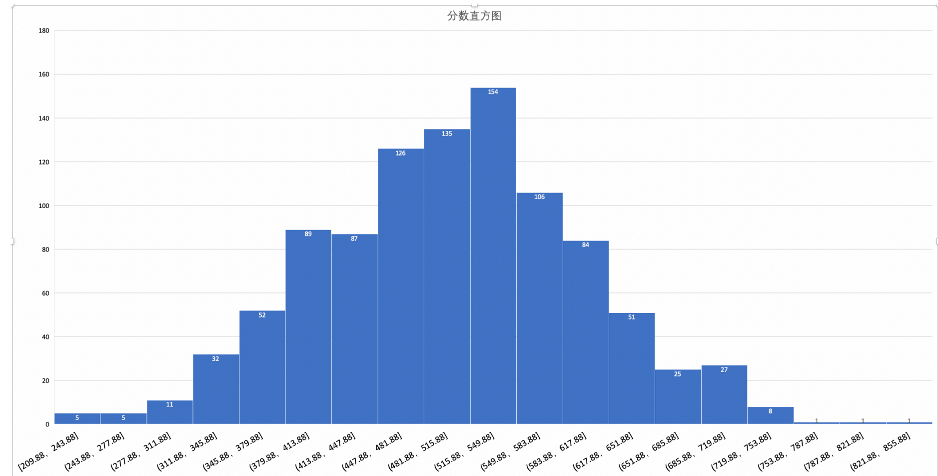
Quantiles And Histograms
分位点和直方图

田志鹏 20190704

1. 背景概念
2. GK算法
3. Doubles Sketch

什么是直方图

分数	排序后
583.38	209.88
430.72	216.23
393.98	217.83
678.00	223.75
464.72	233.03
326.20	258.91
399.29	267.94
514.51	268.27
369.88	271.6
577.91	275.39
531.11	290.61
409.91	296.09
410.50	297.88
558.07	298.66
464.21	298.75
604.29	300.8
424.15	306.45
477.79	306.83
487.45	308.87
586.04	311.06
579.00	311.51
493.73	312.75
326.72	314.87
525.72	315.81
464.30	318.91
614.52	319.62
398.39	320.49
520.14	321.49
330.54	324.18
418.35	324.89
483.42	326.2

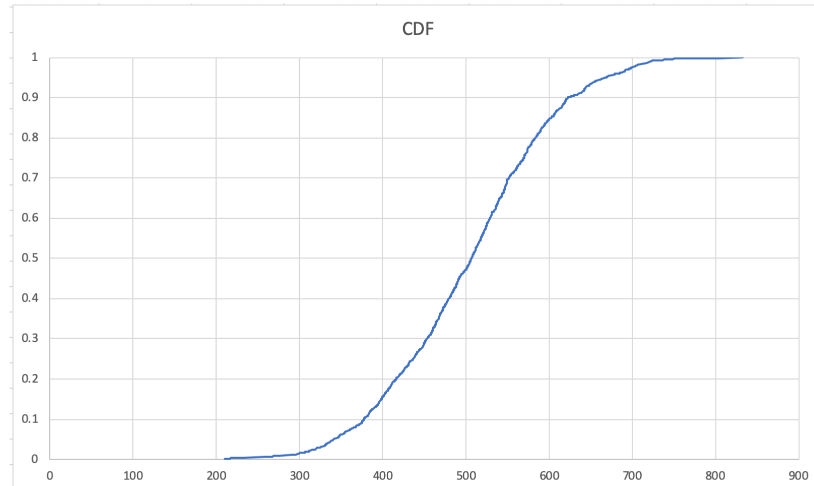


随机生成的某高中学生1000人的高考分数, 最低209, 最高832

如图横轴表示分数段, 纵轴是频次, 就是一个直方图

什么是分位点φ-quantile

0.2	414分
0.5	506分
0.8	581分
0.9	622分



- 中位数 0.5-quantile
- 数据从小到大排序, rank = φ * n 的数据值称为φ-quantile
- 所以问题就是, 求任意一个排名的分数

分位点大家可能没听过, 但中位数大家应该知道, 就是大小排在最中间的那个数. 其实中位数就是0.5分位点

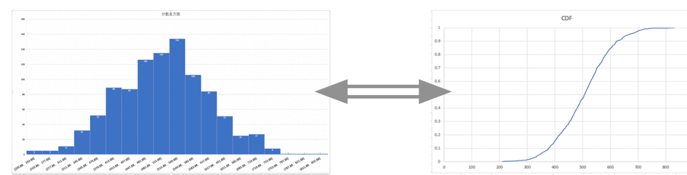
左侧我随意查了几个分为点的值: 0.2分位点是414分表示 有20%的人分数小于414分, 有80%的人分数大于414分

这个图横坐标是分数, 纵坐标是小于等于这个分数的频率. 从这个图很方便找各种分位点值. 比如想找0.7的, 在纵轴找0.7, 线上的点对应横轴的值就是0.7分位点的值.

问题场景

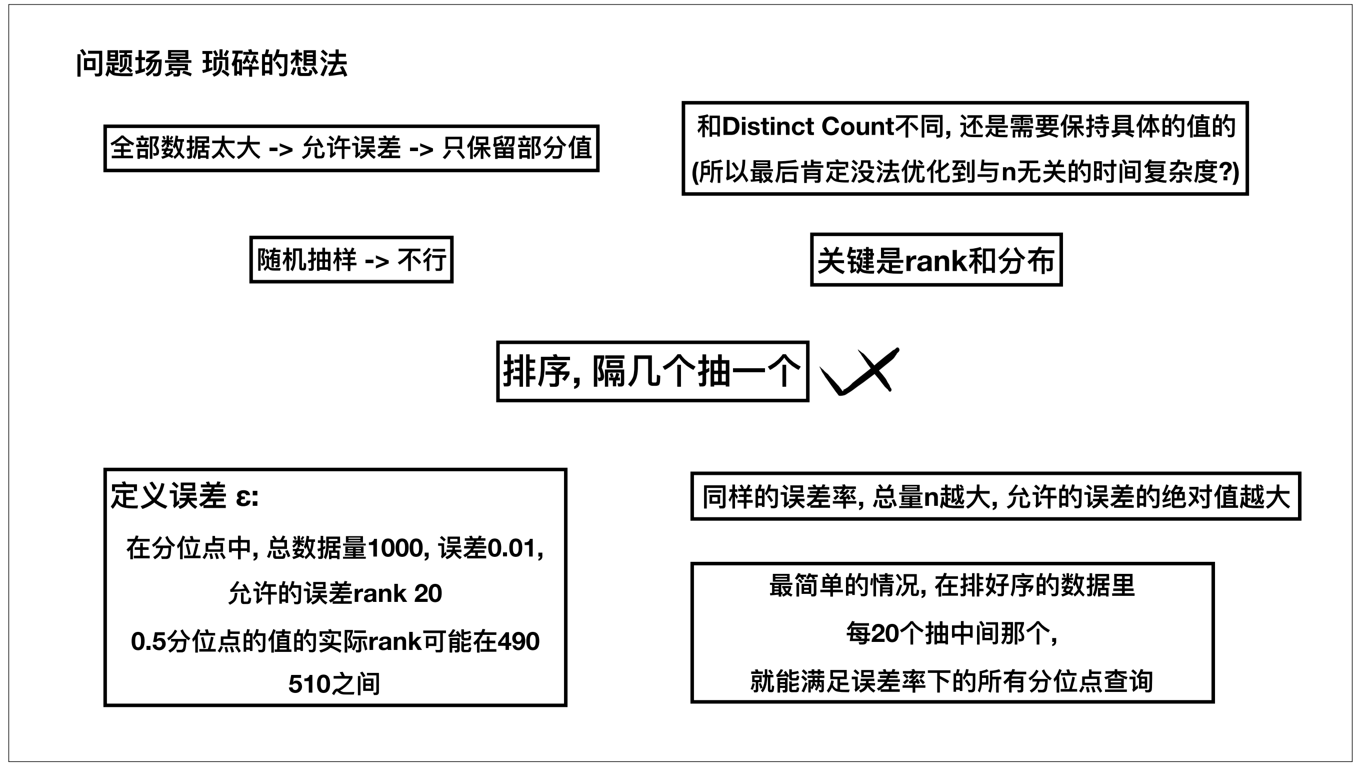
- 直观的解决办法: 排序
- 时间/空间/准确性
- 大数据 and/or 实时计算 场景下如何解决

- 分位点和直方图的相似性(课后作业)



考虑排序所需要的时间复杂度我们知道至少是n*logn, 空间复杂度也要n, 一般情况下这么处理倒还好
但是如果在大数据场景下就很费力, 很难实现, 但是如果能够接受一定的误差率, 那么还是有解决办法的, 今天讲的两个方法都是在保证一定误差率下的估计方法.

不过我的标题是分位点和直方图, 接下来会只拿分位点来讲, 各位猜到是为什么了吗



和Distinct Count不同, 还是需要保持具体的值的, 具体值没了就没法回答quantile
假设我们采取完全随机抽样的方式, 那加入恰好把所有小的值都丢掉了, 留下大的值, 那结果肯定不准.
我们的关键是rank和分布, 完全随机抽样把这个信息全丢了

然后考虑我们可以接受的偏差Epsilon, 注意这里的偏差epsilon不是值偏了多少, 而是排名偏了多少
然后同样误差率....

问题场景

流式数据场景特点和要求:

- **数据根本不是按顺序来的, 每个数据只读一次**
- **总量n也不确定**
- **确保误差率ε**
- **可以分布式算, 可以merge**
- **空间复杂度可控(sublinear)**

和Distinct Count不同, 还是需要保持具体的值的, 具体值没了就没法回答quantile

假设我们采取完全随机抽样的方式, 那加入恰好把所有小的值都丢掉了, 留下大的值, 那结果肯定不准.

我们的关键是rank和分布, 完全随机抽样把这个信息全丢了

我们回头看CDF这个图, 假设在这个图里, 隔一个抽一个, 最后的结果能不能反应整体的分布和排序? 这个是可以的

然后考虑我们可以接受的误差Epsilon,

然后同样误差率

所以最好情况下这个还可以

Greenwald and Khanna 2001　数据结构

元组:　$\{v_i, min_i, max_i\}$　vi: 代表值, min: rank的下界, max: rank的上界

示例:　{ 267.94, 1, 7 } { 308.87, 8, 15 } { 338.44, 16, 25 } ...　ε = 0.01

提问:　倒数第20名(0.02-quantile)多少分?

回答:　338.44分, 最大的偏差是25名

结论:　只要元组长度不那么长, 就能满足偏差的要求
每个元组也不能太短, 组太多浪费空间

但是这样的结构维护的时候插入困难, GK描述了一种新的结构

在一个无序的不确定总量的数据上应用和刚才类似的思路, 就是要抽一部分数据. 接下来两个算法都是在总的数据中抽一部分
第一个名叫GK算法, 我们先讲这个算法用的一个结构.
要维护这样一组数据, vi就这个范围内随便取的一个值, 我这取得是最大值, 叫做代表值, min和max是这个代表值所能代表的排名范围, 就像选人大代表一样哈(-. -)
比如267这个值, 代表了排名1到7的人.

这里重新强调一下刚才的误差, 不是说我回答338分, 误差0.01是说338分乘以0.01, 而是问的这个20名次上下偏0.01

GK算法 数据结构

元组: $\{v_i, g_i, \Delta_i\}$

vi: 代表值，mini: rank的下界, max: rank的上界　　value
g: min i - min i-1　　gap
delta: max I - min i　　delta

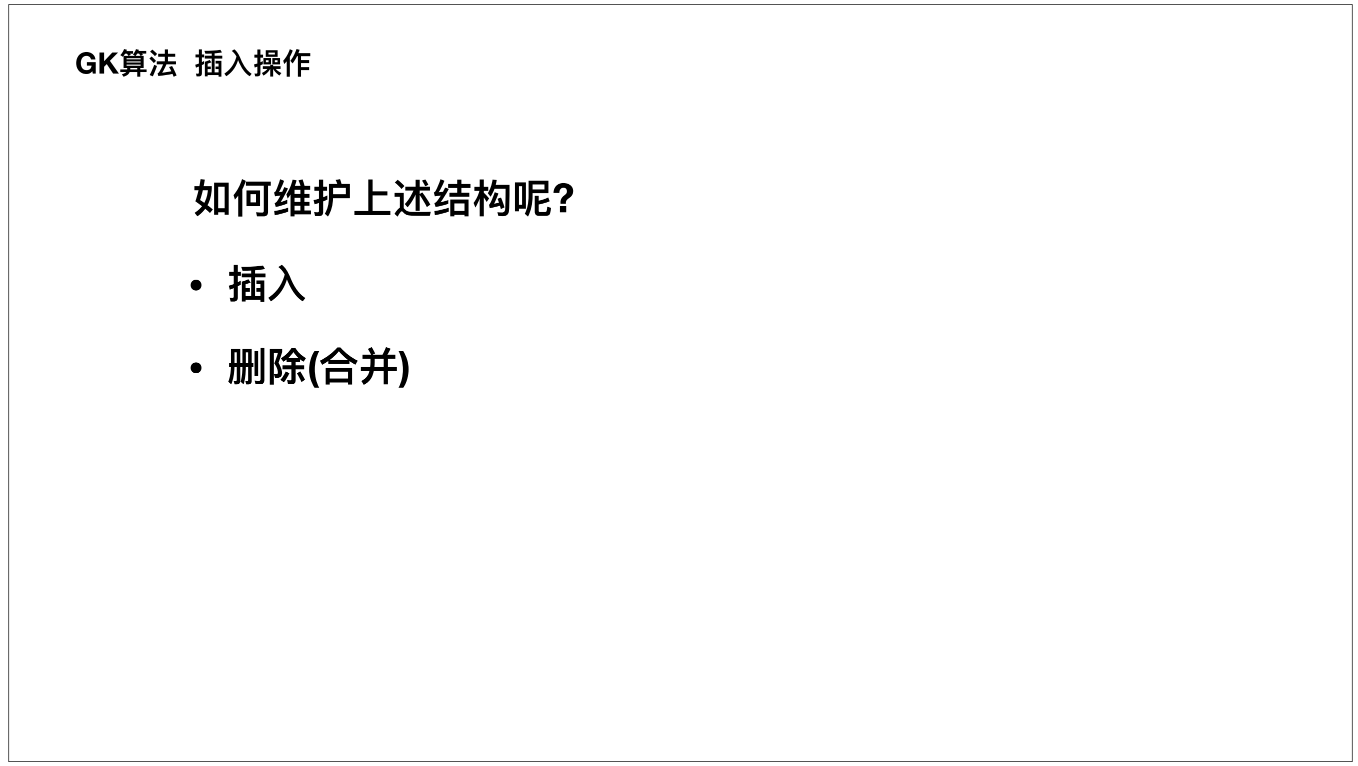
示例:

{ 267.94, 1, 7 } { 308.87, 8, 15 } { 338.44, 16, 25 } ...

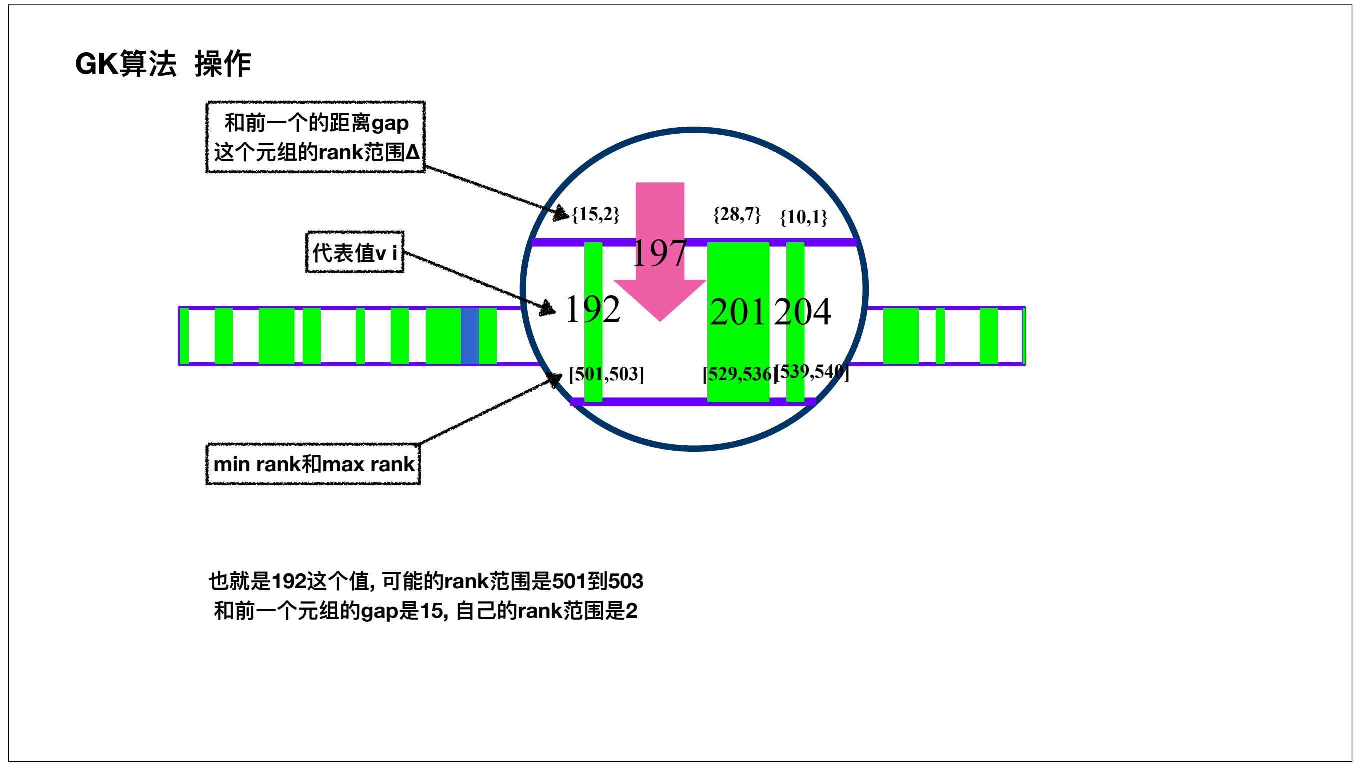
{ 267.94, 1, 6 } { 308.87, 7, 7 } { 338.44, 8, 9 } ...

$$(g_i + \Delta_i) \leq 2\epsilon N$$

接下来稍微改一下. 这个新结构也是三个值, 为了方便记忆，v就是value就是值, g就是gap两个元组间的跨度
delta是上界和下界的差.
这个结构和之前的差不多, 只不过一个插入快查询稍慢, 一个查询快插入稍慢, 本质上是等价的.
从这个结构里查询分位点, 和之前一样, 很简单
只要两个元组整体的跨度小于2eN 20就可以保证误差的要求

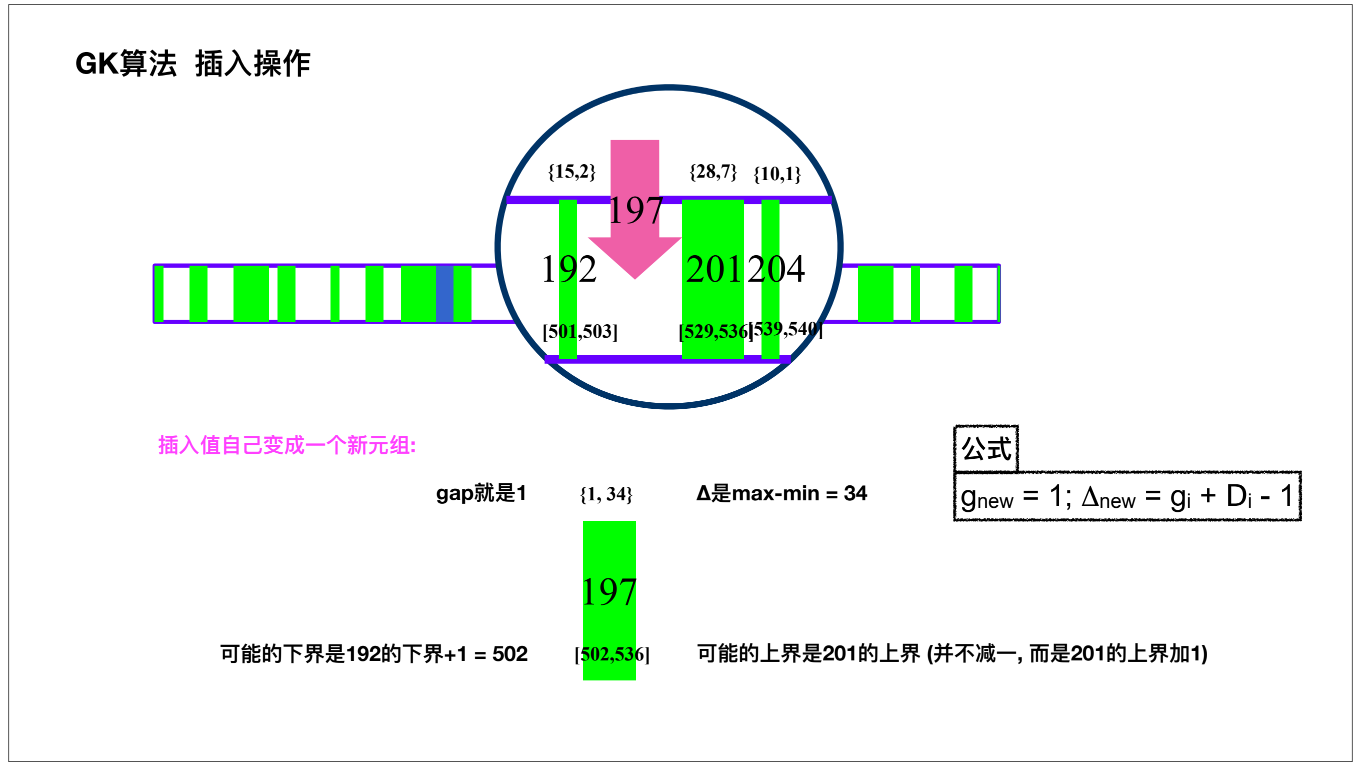


这里一共涉及两种操作, 插入和删除或者叫合并



动画效果.

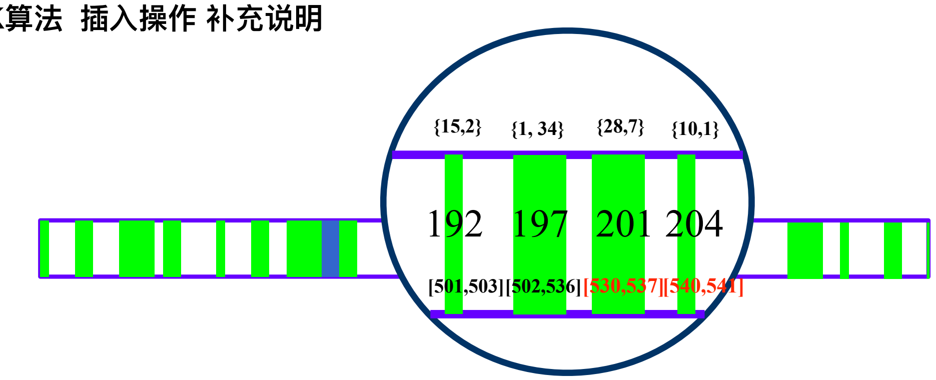
这里借用大神ppt里面的图, 图里这些绿色的就是维护的各个元组, 中间的是代表值vi, 上面俩是g和delta



插入一个值就在整个元组列表中先找到他的合适位置. 然后用他自己生成一个单独的元组.
我们先想一下197的可能排名的上界和下界

这个是总结的公式

GK算法 插入操作 补充说明

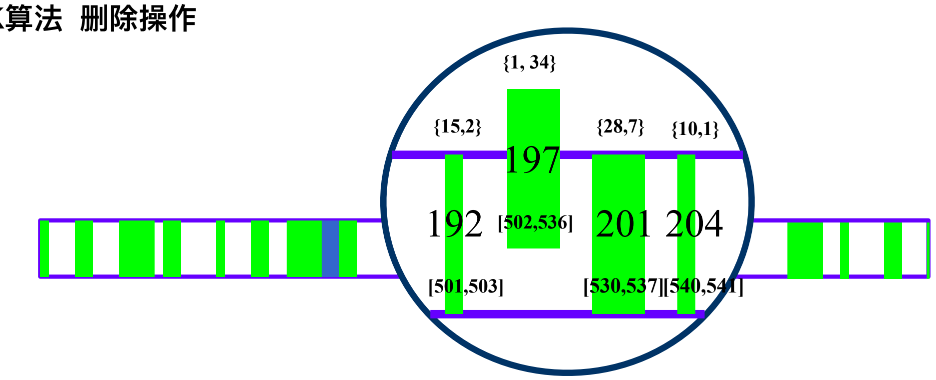


当然插入197之后:

- 排在他后面的元组所有排名都要+1
- 不要由于我们只维护gap的delta, 所有实际并不需要真的去把每个+1
- gap和delta也都不用变!

这里有同学问我, 不维护下面的min/max排名, 那图这些排名咋知道的:
把前序所有元组的gap加起来就是min
再加自己的delta就是max

GK算法 删除操作



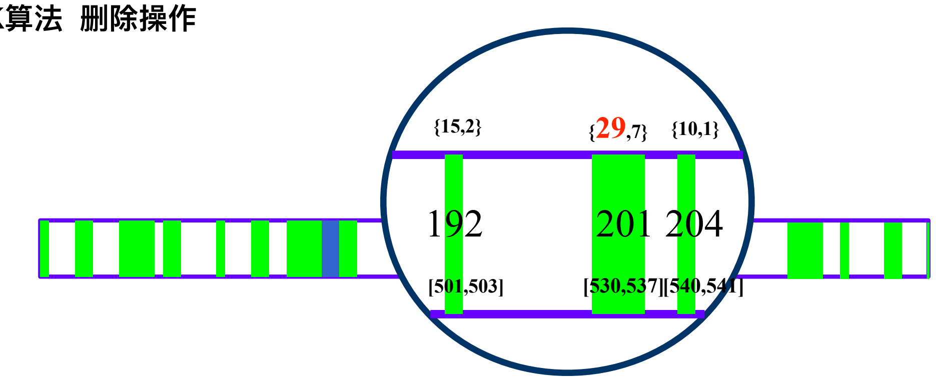
把197合并到201上:

- 对201这个值来讲, 他的可能排名的上下界并不会变, Δ也不会变
- 201这个元组的gap需要变(gap是和前一个元组的距离, 前一个元组删了)

所谓删除, 就是把一个元组合并到另一个元组上, 这里我们把197合并到201上

我们采用的是小的往的方向合并, 其实两个方向都无所谓

GK算法 删除操作



把197合并到201上:

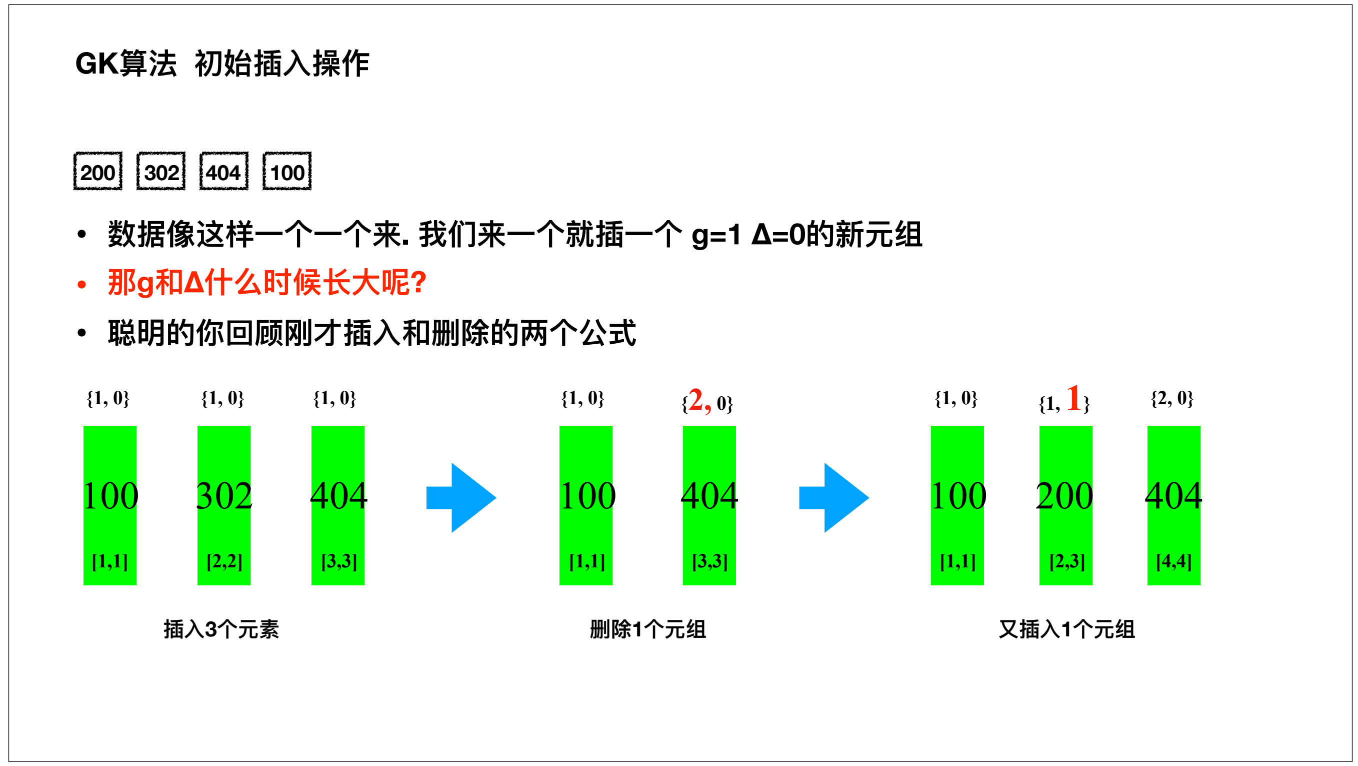
- 对201这个值来讲，他的可能排名的上下界并不会变，Δ也不会变
- 201这个元组的gap需要变
  - (gap是和前一个元组的距离，前一个元组删了，把他的gap加上)

公式

$g_i = g_i + g_{i-1}$

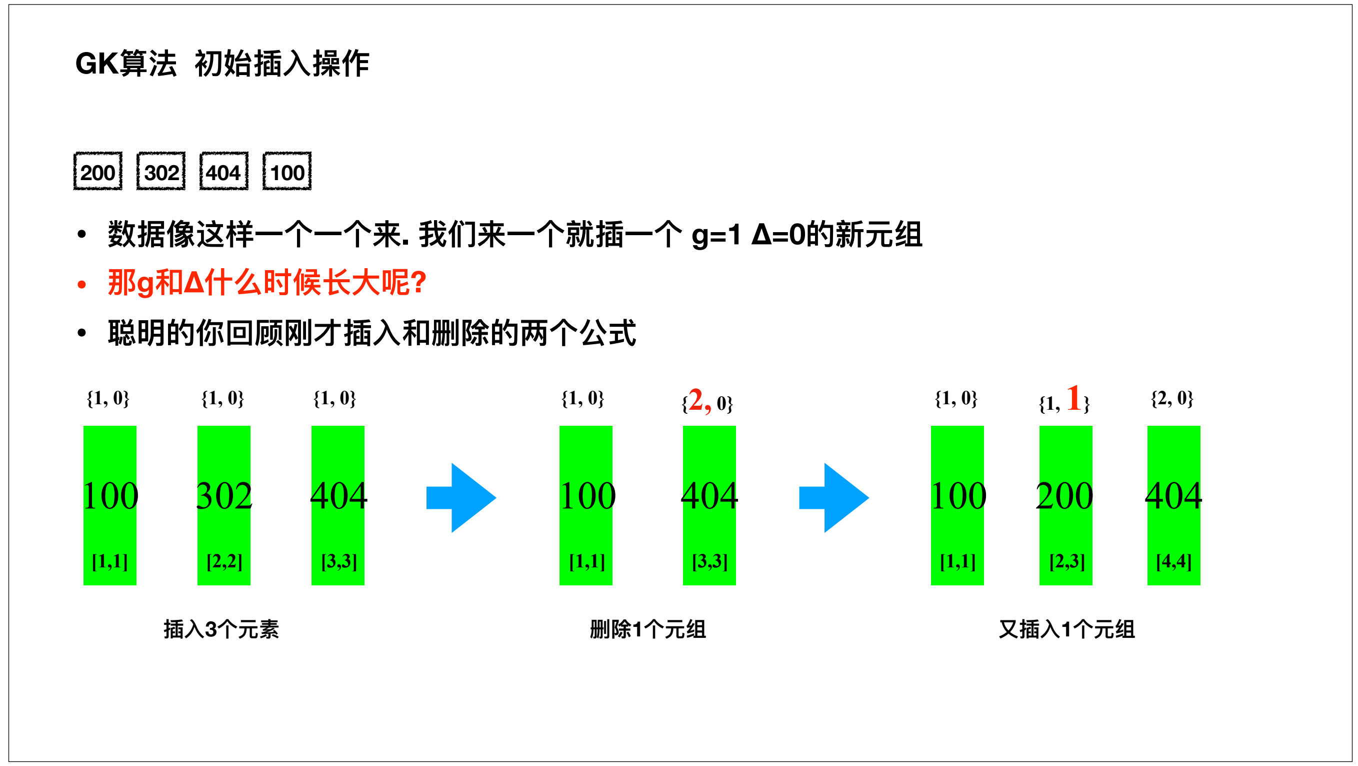
所谓删除，就是把一个元组合并到另一个元组上，这里我们把197合并到201上

我们采用的是小的往的方向合并，其实两个方向都无所谓



这里我补充一下那天没讲明白的初始化的操作.

数据像这样一个一个来. 我们来一个就插一个 g=1 Δ=0的新元组



这里我补充一下那天没讲明白的初始化的操作.

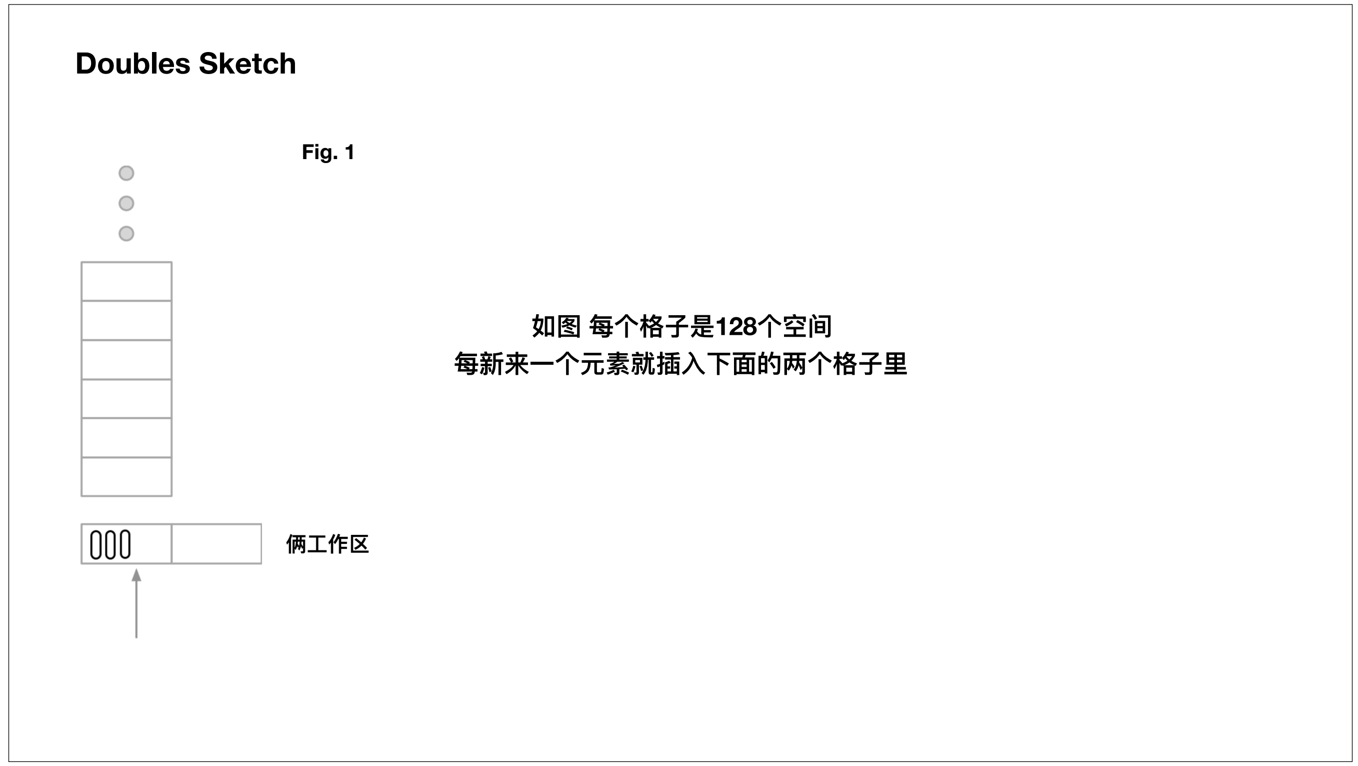
数据像这样一个一个来. 我们来一个就插一个 g=1 Δ=0的新元组

GK算法

执行时机:	每1/2ε从右到左执行一波删除, 确保g+Δ小于等于2εN
优化目标:	g尽量大 Δ尽量小
优化:	• 根据元组的delta不同分组, delta小的尽量删 • 利用树形加快操作效率
空间复杂度:	$(11/2\varepsilon)\log(2\varepsilon n)$
问题:	合并后错误率提高

这个算法整体的原理就是这样, 细节还有很大区别, 优化部分没细看

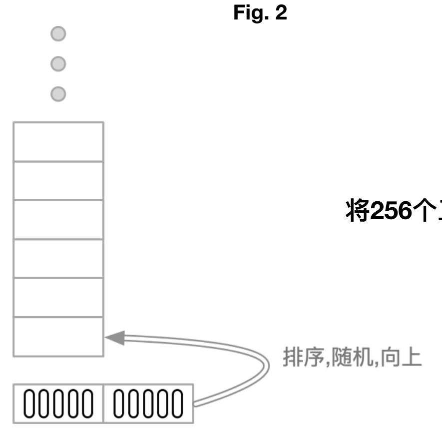
Q & A



接下来这个算法是采用随机的思想, 回顾刚才排序数据上随机抽样的想法.

Doubles Sketch

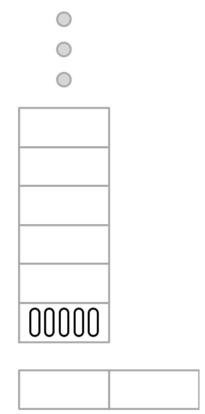
Fig. 2



当俩工作区满了(来了256个元素)
将256个工作区元素排序, 排序后每两个(奇偶)随机抽一个
抽出的128个放到上一级格子

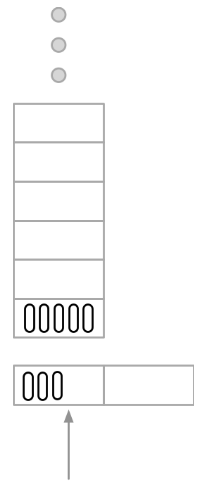
Doubles Sketch

Fig. 3



Doubles Sketch

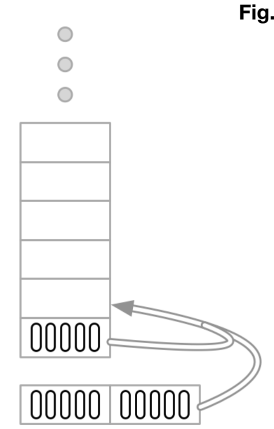
Fig. 4



新元素继续向工作区放

Doubles Sketch

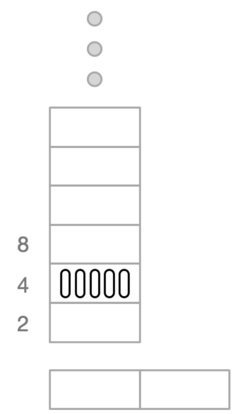
Fig. 5



工作区又放满了, 执行Fig.2里那种"排序抽样向上"的动作,
然而上一级格子已经满了.
新抽的128和上一级格子的128继续"排序抽样向上"

Doubles Sketch

Fig. 6



这样经过两次抽样，就上升到第二级了

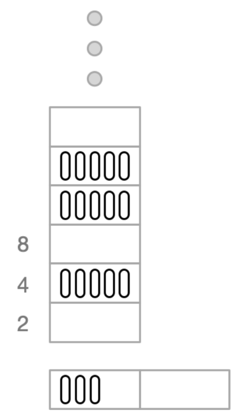
可以看出:

工作区填了2个128会上到第一级
工作区填了4个128才会上到第二级

就和知名游戏1024一样，2 2得4，
要想得8，还再再来个4 (再来2个2)

Doubles Sketch

Fig. 7



按照以上逻辑维护这样的数据结构

每个格子可能有或者没有 有效数据

格子级别不同, 权重不同

如何查询ϕ-quantile?

Doubles Sketch

	341		值	权重	rank
	496.24		217.83	64	0
	507.41		330.86	32	64
	470.68		332.19	8	96
	482.42		341	1	104
	468.1		341.12	16	105
	409.23	前16个是工作区 里面有7个有效值	361.01	4	121
	515.64		392.89	16	125
	545.43		394.29	8	141
	546.29	权重1	400.97	64	149
	570.25		407.33	32	213
	571.61		409.23	1	245
	575.82		430.29	16	246
	593.33		434.28	4	262
其实是无效的 只不过没做clear	642.41		439.31	64	266
	832.52		443.79	32	330
	434.28		459.79	8	362
	456.97		468.1	1	370
	501.64		470.68	1	371
	515.64		471.86	64	372
	545.43	2	475.15	16	436
	571.61		482.42	1	452
	593.33		485.51	4	453
	832.52		490.76	8	457
	361.01		496.24	1	465
	434.28		500	32	466
	485.51		503.49	64	498
	515.64		507.41	1	562
	546.29	4	515.24	8	563
	571.61		515.64	4	571
	593.33		523.45	32	575
	832.52		524.37	64	607
	332.19		525.04	16	671
	394.29		542.75	16	687
	459.79		546.29	4	703
	490.76		564.64	32	707
	515.24	8	569.43	64	739
	573.32		571.61	4	803

这个表格截图中:

- 左侧是999个学生分数灌入刚才那个结构之后的实际情况
- 为了方便观察, 这里的每个格子不是128, 改成了8
- 和刚才的图上下是反的, 这里上面16个是工作区
- 这整个左侧列表每行都有值, 然而有的是无效的, 只不过程序为了速度

- 右侧第一列是我将所有有效值拿出来, 从小到大排序, 一共47个
- 这些值原来所在的级别不同, 每个都有自己的权重
- 第三列则是这个值的可能的rank

- 一个值权重, 就是他在rank里站的坑数
- (或者说一个权重64的值, 是代表原来64个人站在这里的),

- 比如217的权重64, 排名0-63都是他.
- 下一个值330, rank只能从64开始, 他占32个坑.
- 由此把整个rank加了出来

排完rank, 想知道任意分位点的值, 很轻易找到

见excel

Doubles Sketch

- 空间复杂度还是logn级别的, 可以接受
- 可以merge, 两个这种结构要合并, 同级进行”排序抽样向上”操作即可
- 随机算法理解起来简单

见excel

其他算法

- Q-Digest
- T-Digest
- KLL
- Moment Sketch
- Count-Min

参考

- Emory.edu Courses584-StreamDB
- GK介绍博客
- GREENWALD, M. AND KHANNA, S. 2001. Space-efficient online computation of quantile summaries.
- Pankaj K. Agarwal, Graham Cormode, Zengfeng Huang, Jeff Phillips, Zhewei Wei, and Ke Yi. Mergeable summaries.
- DataSketches及源码

Q & A

Thanks