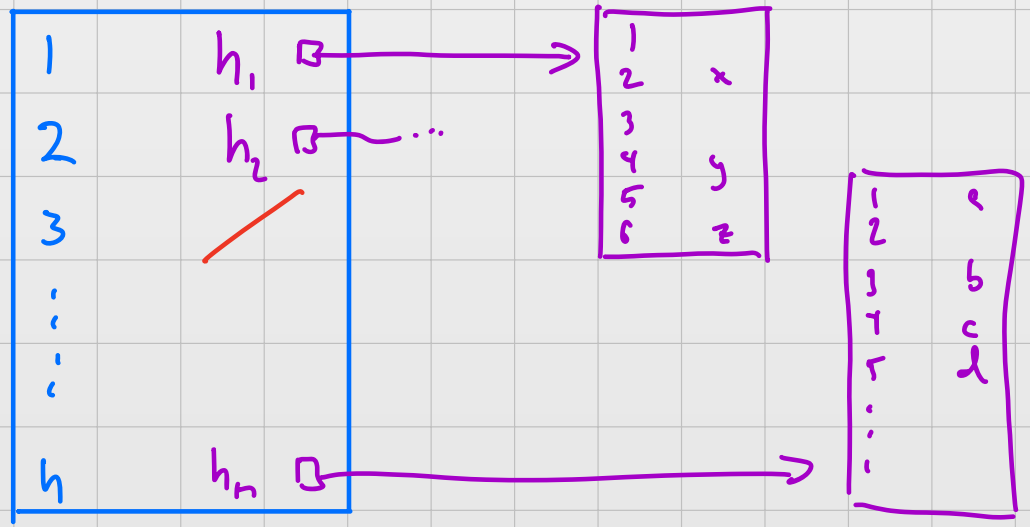
CS 5112

Streaming Algorithms

FKS Perfect Hashing

Level 1:

Level 2: Each bucket is a really big hash table (i.e. collision free)

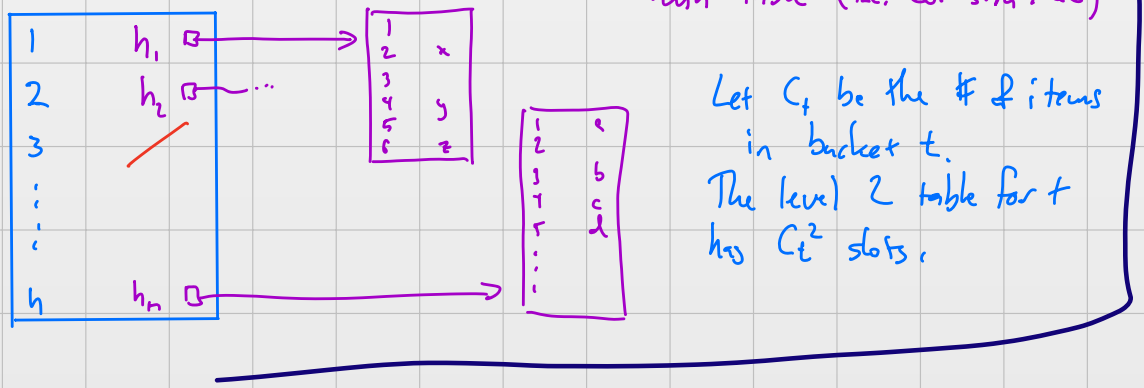


Let C_t be the # of items in bucket t.
The level 2 table for t has C_t^2 slots.

FKS Perfect Hashing

Level 1:

Level 2: Each bucket is a really big hash table (i.e. collision free)



Let C_t be the # of items in bucket t.
The level 2 table for t has C_t^2 slots.

$O(n)$

Construct: 1. Loop through all items and hash them to buckets (lists)

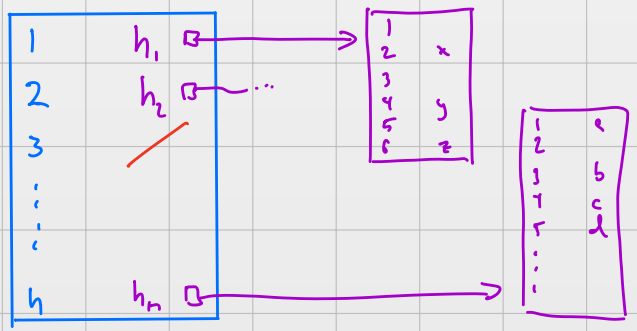
Expected $O(1)$ actually 2

2. For each bucket: $O(C_t)$
   a. Choose a random h_t. $O(1)$
   b. Try to assign items to level 2 slots using h_t.
   c. If there's a collision, go back to a.

FKS Perfect Hashing

Level 1:

Level 2: Each bucket is a really big hash table (i.e. collision free)



Let C_t be the # of items in bucket t.
The level 2 table for t has C_t^2 slots.

Construct: 1. Loop through all items and hash them to buckets (lists) $O(n)$

Expected $O(1)$ (actually 2) { 2. For each bucket: $O(C_t)$
a. Choose a random h_t. $O(1)$
b. Try to assign items to level 2 slots using h_t.
c. If there's a collision, go back to a.

expected

$$O(n) + O(1) \cdot \sum_{t=1}^{n} C_t$$

$$= O(n) + O(n)$$

in expectation

FKS Perfect Hashing

What about space?

Level 1: Uses $O(n)$

Level 2: $\sum_{i=1}^{n} C_i^2$ space $(2n+1 = O(n))$.

$E\{C_i^2\}$

C_t = RV for # of items in level-1 bucket t.

$I_{i,t}$ = RV for whether $h(x_i) = t$. $\begin{cases} 1 & \text{if } h(x_i) = t \\ 0 & \text{otherwise} \end{cases}$

$$C_t = \sum_{i=1}^{n} I_{i,t} \qquad E[C_t] = \sum_{i=1}^{n} E[I_{i,t}] = n/n = 1$$

Want something similar for C_t^2.

$J_{i,j,t}$ = RV indicating whether $h(x_i) = t \wedge h(x_j) = t$

$$= \begin{cases} 1 & \text{if } h(x_i) = t \text{ and } h(x_j) = t \\ 0 & \text{otherwise} \end{cases}$$

$$C_t^2 = \sum_{i,j} J_{i,j,t}$$

$$E\{C_t^2\} = \sum_{i,j} J_{i,j,t}$$

i's \ j's	a	b	c	d	...
a	.	.	.	.	
b	.	.	.	.	
c	.	.	.	.	
d	.	.	.	.	
⋮					

$$E\left\{\sum_{t=1}^{m} C_t^2\right\} = \sum_{i,j,t} E\{J_{i,j,t}\} = \sum_{i,j} E\left\{\sum_{t=1}^{n} J_{i,j,t}\right\} = \sum_{i,j} Pr(h(x_i) = h(x_j))$$

$$E\left\{\sum_{t=1}^{m} C_t^2\right\} = \sum_{i,j,t} E\{T_{i,j,t}\} = \sum_{i,j} E\left\{\sum_{t=1}^{m} T_{i,j,t}\right\} = \sum_{i,j} Pr(h(x_i)=h(x_j))$$

$$= \sum_{i=j} 1 + \sum_{i \neq j} Pr(h(x_i) = h(x_j))$$

$$= n + \frac{1}{n} \sum_{i \neq j} 1 = n + (n^2 + n)\frac{1}{n}$$

$$= 2n + 1.$$

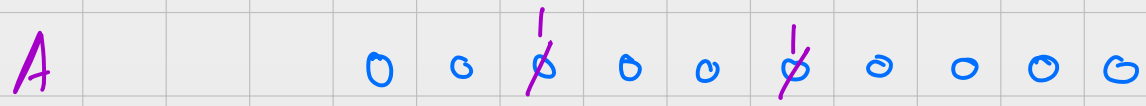
Streaming Algorithms

	1	2	3	...									
Stream →	A	C	R	L	A	B	X	Z	D	A	A	A	B

The stream is really big, so we can't record it.
Still want to answer questions: 1. What's the most common item?
2. A list of frequent items
3. # unique items

Distinct Element Problem

First attempt: Use a little hash table:



Distinct Element Problem

Flajolet-Martin

Clever idea: Hash each item to $\{1, \ldots, m\}$, keep track of the smallest hash, s.

If we know what the smallest hash is, we can estimate the # of unique items.

Return cm/s.

Problem: lots of variance.

Majority Problem

Input: A stream
Output: The majority element if it exists. (and whether one exists)

Problem: Can't determine if a majority exist with $< \frac{n}{2}$ space.

Boyer-Moore

Store: candidate item a
counter c

On ith item:

Majority Problem

Boyer-Moore

Store: candidate item a

counter c

On ith item x_i:

$a \neq \phi$

if $a = x_i$, c++

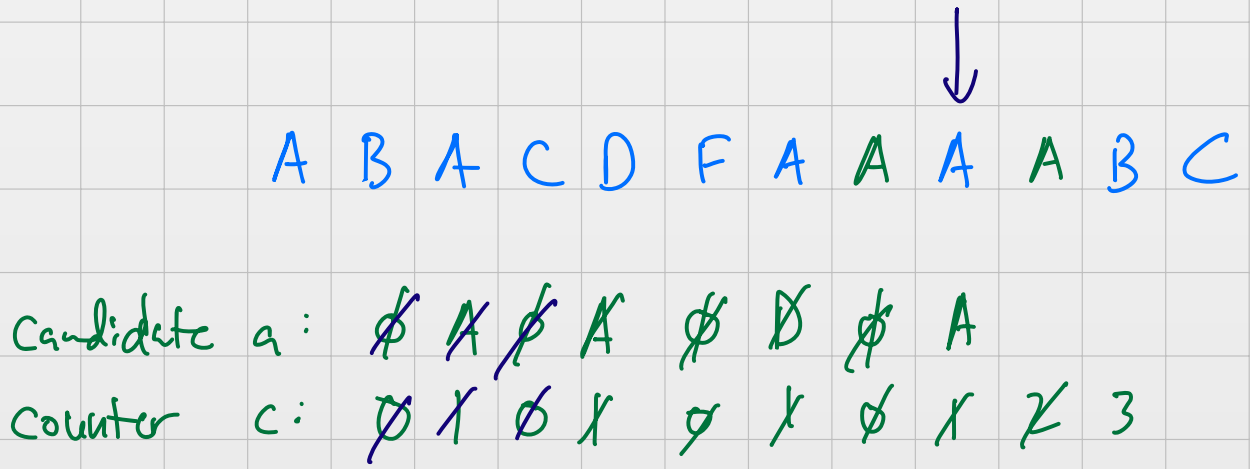
else c--

if $c = 0$, set $a = \phi$

$a = \phi$

set $a = x_i$

c++


A B A C D F A A A A B C
candidate a:
counter c:
3

Boyer-Moore

Suppose there is a majority element, B.

Case 1: C never goes back to 0. → we see first item more than all other items combined
⇒ first item is B.

Case 2: C does go back to 0.

↳ Let's look at the last time C goes to 0.