Hashes and randomness

Ben Langmead

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

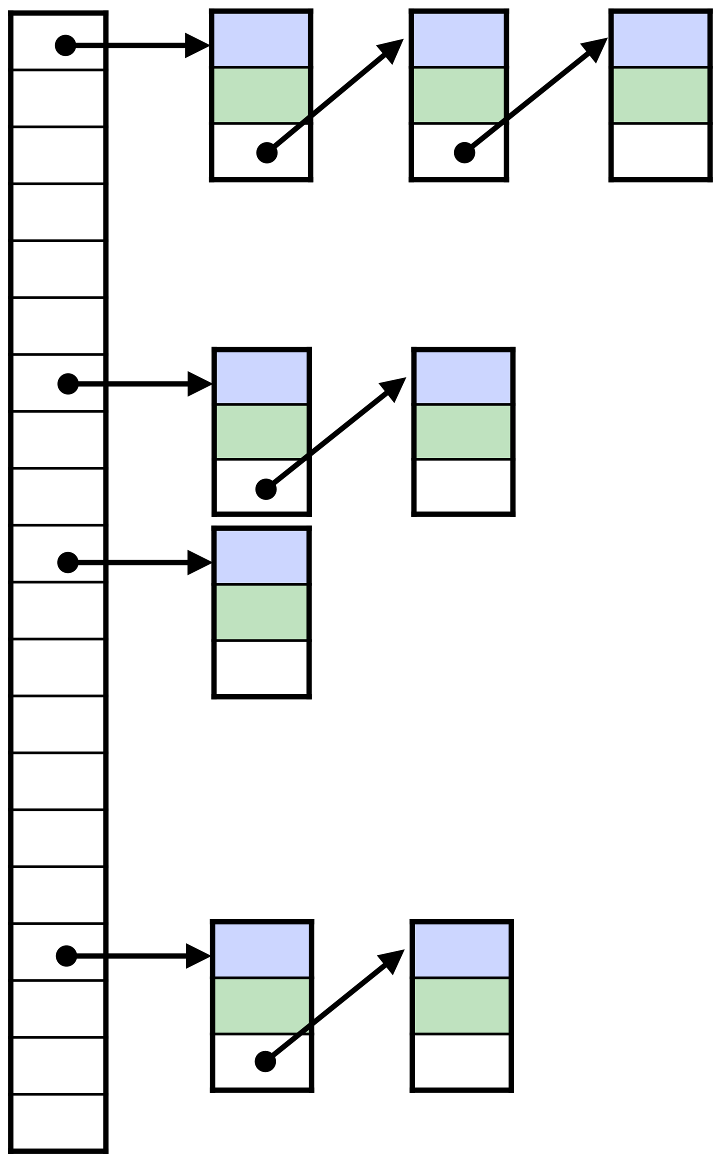
Department of Computer Science

Hash Function

Balls & Bins assumes *uniformity* & *independence*

How / in what sense do hash functions provide those?

Should mapping from keys to buckets should be "random"?



Hash Function?

```
// library function returning
// a "truly" random integer
extern int truly_random();

int hash(int x) {
    return truly_random();
}
```

Bad sign 1: non-deterministic

Bad sign 2: doesn't depend on x

Hash Function

```
int hash(int x) {
    int a = 349534879; // randomly chosen
    int b = 23479238;  // randomly chosen

    ...

    // return some function of x, a and b
}
```

E.g. The family $h_{a,b}(x) = (ax + b) \mod p$ where p is prime & a, b are uniform, independent draws from $\{0, 1, \ldots, p-1\}$

When did we choose a and b?

Algorithm phases

Phase 1

Choose algorithm

Determines *where* randomness is needed & *how much*

Phase 2

Random interlude

Make random draws.

Choose hash functions.

Phase 3

Data arrives; Execute!

Use hash functions chosen in Phase 2.

Algorithm phases

Random variables used in analysis are random over the ***choice of hash functions***

Not over the input data

We make ***no distributional assumptions*** about the input.

Phase 1

Choose algorithm

Phase 2

Random interlude

Phase 3

Data arrives; Execute!

Algorithm phases

Could remove the hash functions and instead make distributional assumptions about the input itself

But the ability to work with any input data is gone

Phase 3

Data arrives
Execute!