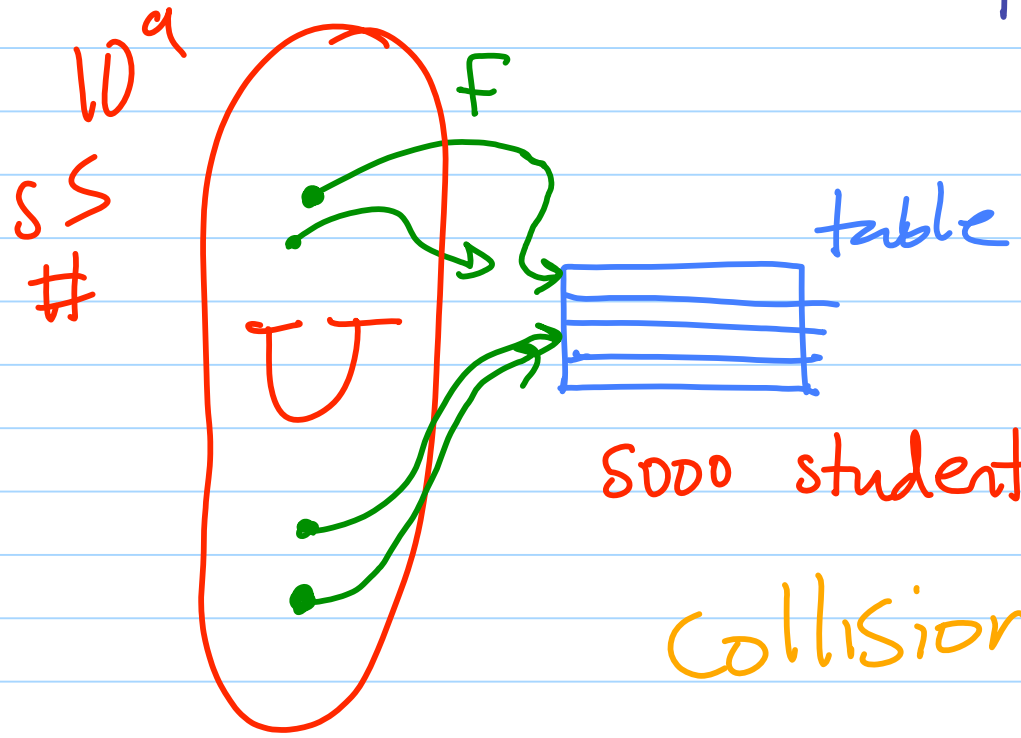
We want time efficiency of direct addressing but we can't waste so much space



Must have lots of elements in universe map to same slot

Hash function

function that maps from hashcode

to $\{0, \ldots, m-1\}$

hashcode: object → int

some integer from $0, \ldots, m-1$

m: hash table size

Desired property — for each element $x \in U$, hash(x.hashcode()) is equally likely to be any int in $\{0, \ldots, m-1\}$

prob $1/m$

Pick a hash function to be a mathematical function that maps a random integer to $0, \ldots, m-1$.

One thought: hashcode mod m (% operator) } division method

pick something that breaks natural patterns that occur

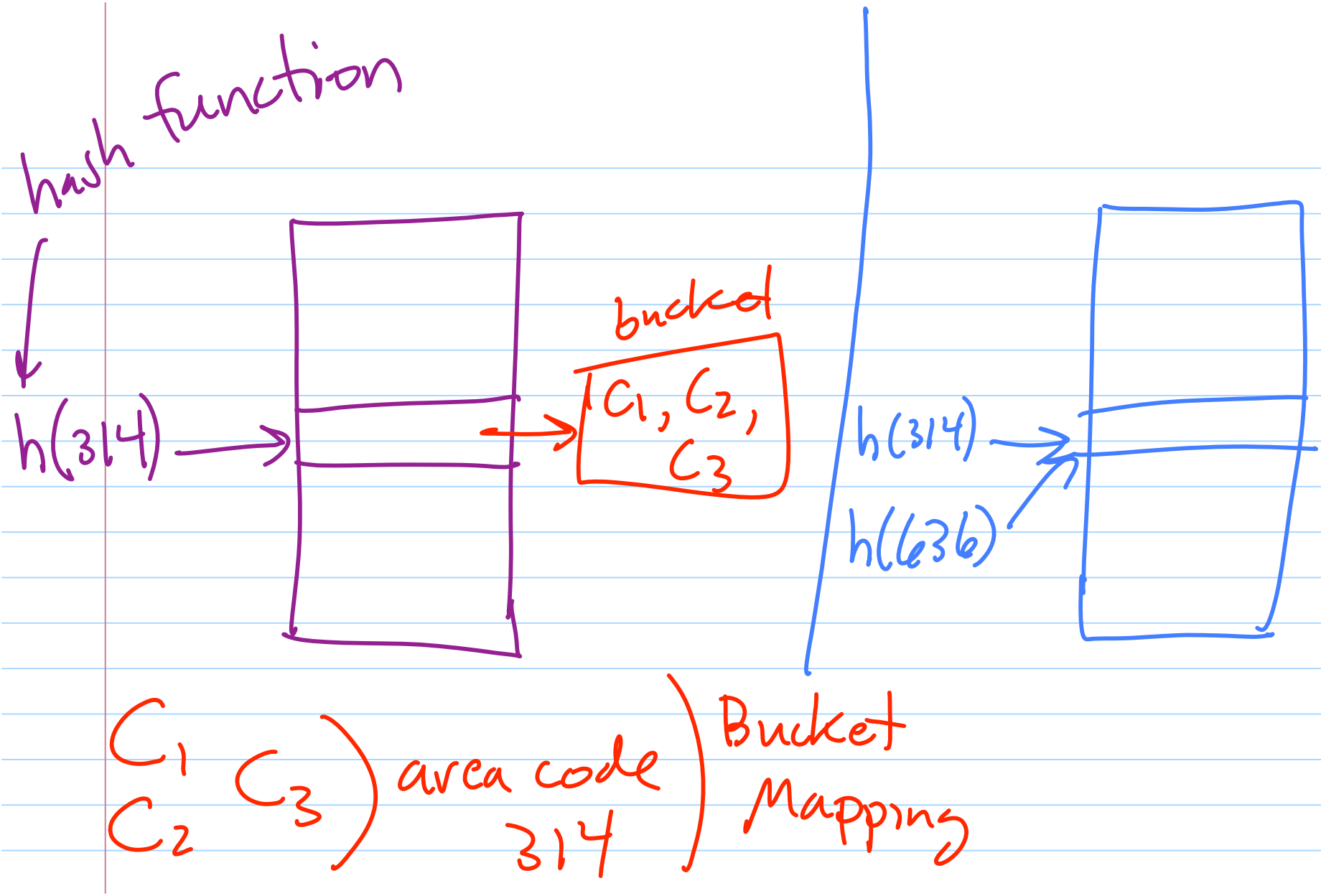
multiplication method - multiply by irrational #

Still we can have collisions -
what do we do?

Open Addressing

Separate Chaining

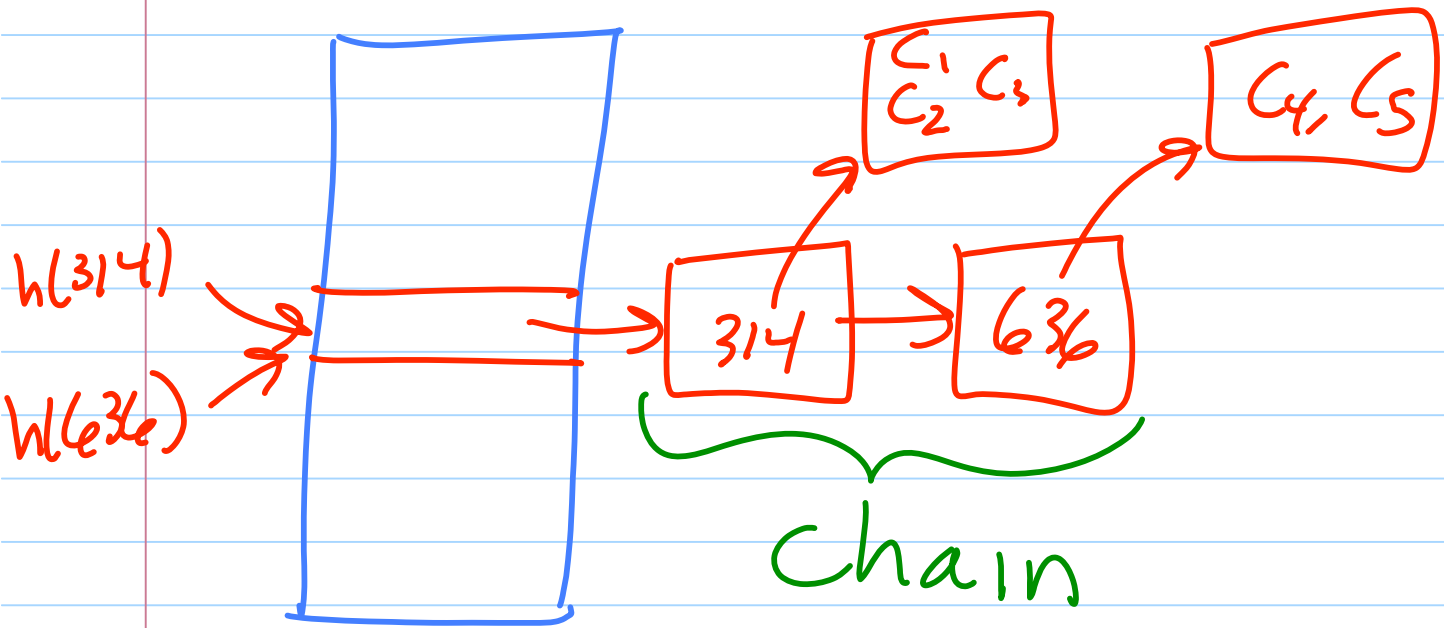
} these data structures differ in how collisions are handled



C1 C2 C3) area code 314) Bucket Mapping

Separate Chaining

One solution: keep a list of all elements (buckets) that hash to the same slot



C1, C2, C3) 314

C4, C5) 636

Open Addressing

If the slot you hash to is already occupied (there's a collision), go somewhere else.

Requires there's $\geq$ one slot per element. $(m \geq n)$

How do we decide where to go next?

Important that element e always follows the same sequence of slots as it looks for an open one.

probe sequence