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

We have $10^g$ students. The universe must have lots of elements in order to map to the same slot in the table, which has 5000 students. Collision
Hash Function

function that maps from \( \text{hashcode} \) to \( \{0, \ldots, m-1\}\)

hash table size

Desired property - each element \( x \in \mathbb{U} \), \( \text{hash}(x, \text{hashcode}(x)) \) is equally likely to be any int in \( \{0, \ldots, m-1\} \)
Pick a hash function to be a mathematical function that maps a random integer to 0, ..., m-1.

One thought hashcode mod m \( \% \)

pick something that breaks natural patterns that occur

multiplication method - multiply by irrational number
Still we can have collisions—what do we do?

Open Addressing
Separate Chaining

these data structures differ in how collisions are handled
Hash function

\[ h(314) \]

\[ h(314), h(636) \]

\( C_1, C_2, C_3 \) area code

Bucket

Mapping
**Separate Chaining**

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

![Diagram showing separate chaining with hash values and elements]

- $h(314)$
- $h(636)$

Nodes $C_1$, $C_2$, $C_3$, $C_4$, $C_5$ represent elements in the hash table.
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