Suppose we want to store a list of student records.

Student object

Object data

\[ \begin{align*}
\text{ID} & \quad \text{name} \\
\text{year} & \quad \text{courses}
\end{align*} \]  

Want searches to be quick. Which would make a good key? (unique?)

Suppose ID is 6 digits (ex: 123456)

Want to store records in a way to quickly search by ID.

ex. courses for student with ID=123456

\[
\begin{align*}
\text{array of size } 10^6 \quad \text{why } 10^6? \quad \text{highest ID} = 999999 \\
\text{index} = 123456 \quad 1000000
\end{align*}
\]

Problem? Mostly empty. Lots of wasted space!

Instead since there are closer to \(10^4\) students, use array of size \(10^4\). How to search by ID now?

\[
\begin{align*}
\text{ex. ID: 123456} \quad \text{Use last 4 digits of ID} \\
\text{index} = 3456
\end{align*}
\]

Problem? Two students may have the same last 4 digits.
Few issues:

1. How to choose key?
2. How to map keys to index?
3. (How to) deal with 2 keys that map to same index? (Collision)

Keys:
- Should be unique to each record (or close to unique)
- Should be integers
- If string, convert to int

Map key to index: $\text{table size} = n$

Always want index to be in $[0, n-1]$

How to ensure? mod!

$$\text{index} = \text{key mod } n = \text{hash(key)}$$

Hash function:

Many many variations of hash functions
We will mostly look at key mod n + some variations

Collisions:

What to do if 2 keys hash (map) to same index?

1. Separate chaining: Keep doubly linked list of all elements that hash to the same index
Suppose keys: 1, 4, 9, 16, 21, 36, 49, 24^+ 

duplicate.

\begin{align*}
\text{Time: } m &= \# \text{ of elements, } n = \text{table size}.
\end{align*}

**Worst-case:**

Find: \(O(m)\) (all elements hash to same index).

Insert: if duplicates OK (stored in separate nodes): \(O(1)\)

"want to store duplicates together: \(O(m)^2\) find().

Remove: \(O(m)\) (requires find()).

**Average-case:**

Find / Insert / Remove: \(O(1)\)

Idea: elements typically spread out evenly over all indices.

\[ m \text{ elements, } n \text{ indices } \Rightarrow O(1 + \frac{m}{n}) \]

But \(\frac{m}{n}\) will be bounded by a constant \(C\).

\[ \frac{m}{n} = \text{load} = \lambda \]

Ex: \(2 \times 10000\) elements, \(n = 1000\)

\[ C \approx 10 \]

Worst-case is very rare!!