- Quick Selection
  - Idea: similar to quick sort
    
    pick a pivot
  
  split the array into two parts
  
  suppose pivot point a[p] is the i-th smallest number.
  
  if k < i
    \[\text{recursively on the left}\]
  
  else if k = i
    \[\text{return a[p]}\]
  
  else
    \[\text{recursively on the right}\]

- Hash tables
  - Goal: maintain a set of numbers in 0, 1, 2, ..., N-1
    
    think of \[\sqrt{N} = 2^q\]
  
  set \(\{0, 1, 2, \ldots, N-1\}\) (set of all possible values)
  
  is called the universe.

  in a set, there will be \(n\) numbers from the universe.

  1) Efficient look-up: checking whether a number is in the set takes \(O(1)\)

  2) Space: data structure should take \(O(n)\) space.

- hash table
  
  1) choose size \(m\) of the hash table.
  
  2) choose a hash function \(f(x)\)
    
    \[f(x): \{0, 1, \ldots, N-1\} \rightarrow \{0, 1, 2, \ldots, m-1\}\]

  3) allocate an array size \(m\), at each location, have a pointer to a linked list.
4 operations on $X$ will be performed on the linked list at location $f(x)$.

$123 \quad f(123) = 5$

$234 \quad f(234) = 11$

$345 \quad f(345) = 5$

Best case: Every number in the set is in its own location (all linked lists have size $\leq 1$).

Worst case: Every number in the set has the same hash function value.

Running time = $O(n)$ essentially a linked list.

- Choosing a random hash function

# Possible hash functions? For each number in universe, there are $m$ choices

$\frac{m}{N}$

Storing a totally random hash function takes $\log_2 m^N$ bits

$= N \cdot \log_2 m$

- In practice: Choose a pairwise independent hash family $f$

$\forall x, y \in \{0, 1, \ldots, N-1\}, \text{ if } x \neq y$

$D \rightarrow f(x) = f(y) \Rightarrow \boxed{1}$
\[ a, b, y \leq 0, 1, \ldots, N-1, \quad i \leftrightarrow x \leftrightarrow y \]

\[ \Pr_{f \sim F} [f(x) = f(y)] = \frac{1}{m} \quad \text{size of hash table} \]

there are pairwise independent hash families of size \( N^2 \).

storing a random function in this family

takes \( \log_2 N^2 = 2 \log_2 N \) bits.

- Example: hash table has \( n \) elements \( \{x_1, x_2, \ldots, x_n\} \)
  
  query \( y \) (want to find out whether \( y \in \{x_1, x_2, \ldots, x_n\} \))

  what is the expected running time of the query?

  idea: running time = length of the linked list at \( f(y) \)

  let \( X \) be the length of the linked list.

  try to compute \( \mathbb{E}[X] \)

  let \( X_i \) be a random variable

  \[ X_i = \begin{cases} 0, & f(x_i) = f(y) \\ 1, & f(x_i) \neq f(y) \end{cases} \]

  \[ X = X_1 + X_2 + \cdots + X_n \]

  \[ \mathbb{E}[X] = \mathbb{E}[X_1] + \mathbb{E}[X_2] + \cdots + \mathbb{E}[X_n] \]

  \[ \mathbb{E}[X_i] = \begin{cases} 1, & X_i = y \\ \frac{1}{m}, & X_i \neq y \end{cases} \quad \text{pairwise independent hash family.} \]

  \[ \leq 1 + \frac{n-1}{m} \leq O(1 + \frac{n}{m}) \]

  if we choose \( m = \Theta(n) \)

  \[ \mathbb{E}[X] = O(1) \]

  \[ \square \]