Time Based Key-Value Store

Leetcode #981

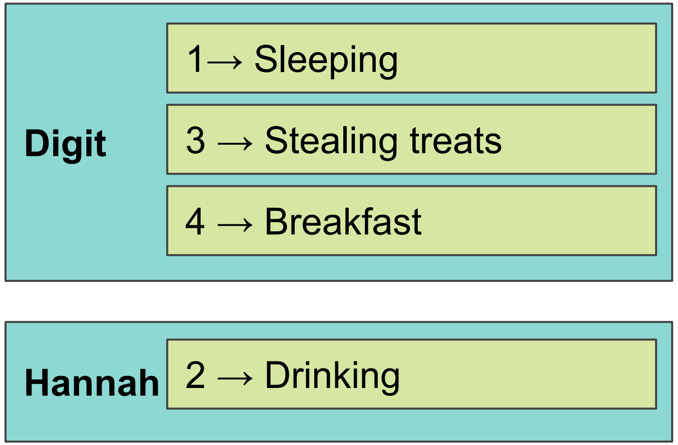
Problem:

- Design a time-based key-value data structure that can store multiple values for the same key at different time stamps and retrieve the key's value at a certain timestamp.
  - SET: Stores the key key with the value value at the given time timestamp.
  - GET: Returns a value such that set was called previously, with timestamp_prev <= timestamp. If there are multiple such values, it returns the value associated with the largest timestamp_prev. If there are no values, it returns "".
- All the timestamps timestamp of set are strictly increasing
- A lot of calls made to set and get.

Examples:

Digit	1→ Sleeping
	3 → Stealing treats
	4 → Breakfast
Hannah	2 → Drinking

Examples:



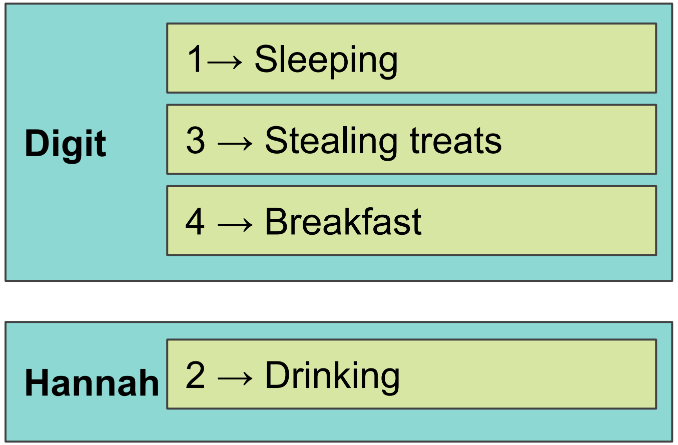
get(Digit, 3) → Stealing treats

get(Digit, 2) → Sleeping

get(Hannah, 0) → “”

get(Bobo, 3) → “”

Examples:



get(Digit, 3) → Stealing treats

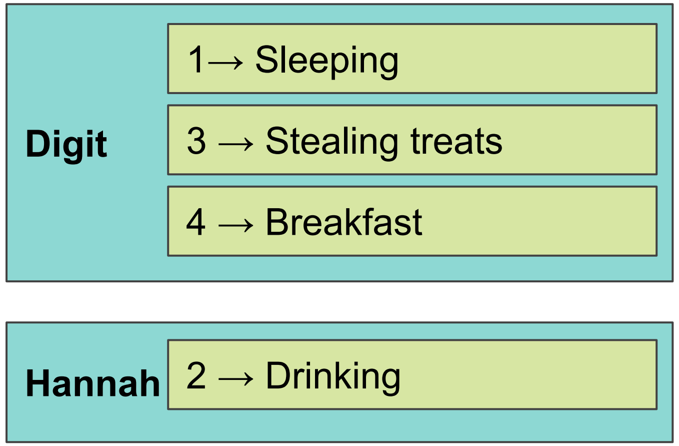
get(Digit, 2) → Sleeping

get(Hannah, 0) → “”

get(Bobo, 3) → “”

- Key exists
- Timestamp <= exists

Examples:



get(Digit, 3) → Stealing treats

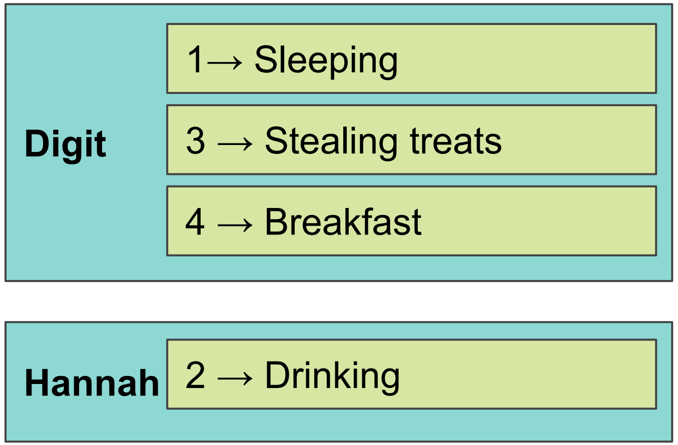
get(Digit, 2) → Sleeping

get(Hannah, 0) → “”

get(Bobo, 3) → “”

- Key exists
- Timestamp <= exists so choose max

Examples:



get(Digit, 3) → Stealing treats

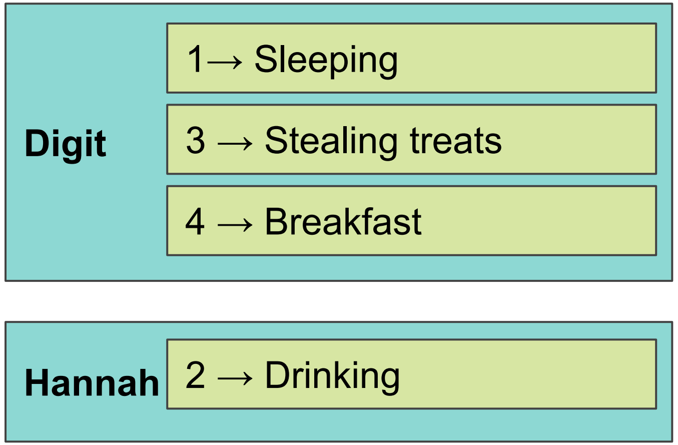
get(Digit, 2) → Sleeping

get(Hannah, 0) → “”

get(Bobo, 3) → “”

- Key exists
- Timestamp <= doesn’t exists

Examples:



get(Digit, 3) → Stealing treats

get(Digit, 2) → Sleeping

get(Hannah, 0) → “”

get(Bobo, 3) → “”

- Key doesn’t exist

Needs:

- Quick insert for **key**
  - “Put Digit in as a key”
- Quick lookup on **key** (no sorting required)
  - “What timestamps does Digit have?”
- Quick insert for a new **timestamp for a key**
  - “Give Digit the timestamp 3 with a value ‘Sleeping’”
- Quick sorted lookup on **timestamps**
  - “Given these timestamps for Digit, what timestamp is less than or equal to 3?”

Needs:

- Quick insert for **key**
  - "Put Digit in as a key"
- Quick lookup on **key** (no sorting required)
  - "What timestamps does Digit have?"
- Quick insert for a new **timestamp for a key**
  - "Give Digit the timestamp 3 with a value 'Sleeping'"
- Quick sorted lookup on **timestamps**
  - "Given these timestamps for Digit, what timestamp is less than or equal to 3?"

- HashMap has O(1) average insert and lookup
  - O(n) worst case with bad hash function

Needs:

- Quick insert for **key**
  - “Put Digit in as a key”
- Quick lookup on **key** (no sorting required)
  - “What timestamps does Digit have?”
- Quick insert for a new **timestamp for a key**
  - “Give Digit the timestamp 3 with a value ‘Sleeping’”
- Quick sorted lookup on **timestamps**
  - “Given these timestamps for Digit, what timestamp is less than or equal to 3?”

- HashMap has O(1) average insert
  - O(n) worst case with bad hash function
- But data isn’t sorted so won’t be optimal for lookup

Needs:

- Quick insert for **key**
    - "Put Digit in as a key"
- Quick lookup on **key** (no sorting required)
    - "What timestamps does Digit have?"
- Quick insert for a new **timestamp for a key**
    - "Give Digit the timestamp 3 with a value 'Sleeping'"
- Quick sorted lookup on **timestamps**
    - "Given these timestamps for Digit, what timestamp is less than or equal to 3?"

- Instead, we can use an array to store values
    - O(1) insert
- Perform binary search on array for O(log n) since we know it is sorted

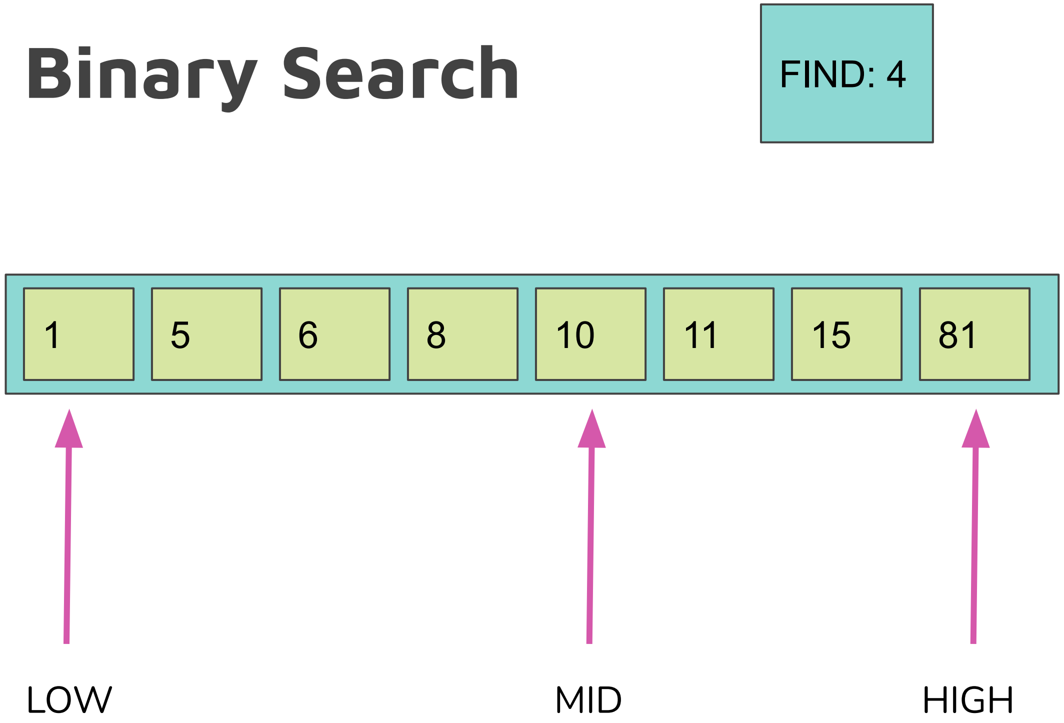
Algorithm: Set

- If the key already exists, append the new timestamp and value to the current list associated with the key
- Otherwise, add the key and add a new list that contains the timestamp and value

Algorithm: Get

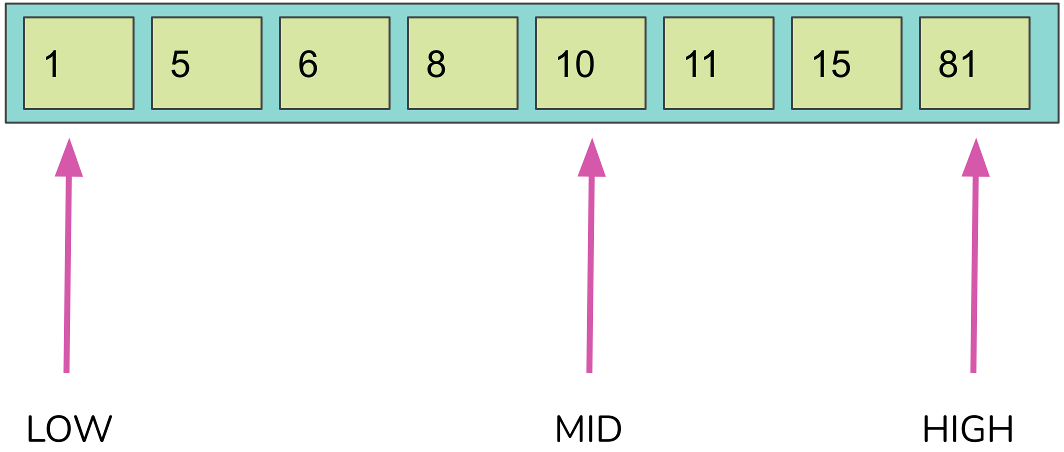
- If the key doesn't exist, return “”
- If the key exists but the smallest timestamp associated with it is larger than the timestamp we are looking for, return “”
- Run a binary search on the list associated with the key and find the largest timestamp that is <= the timestamp. Return its value

Binary Search



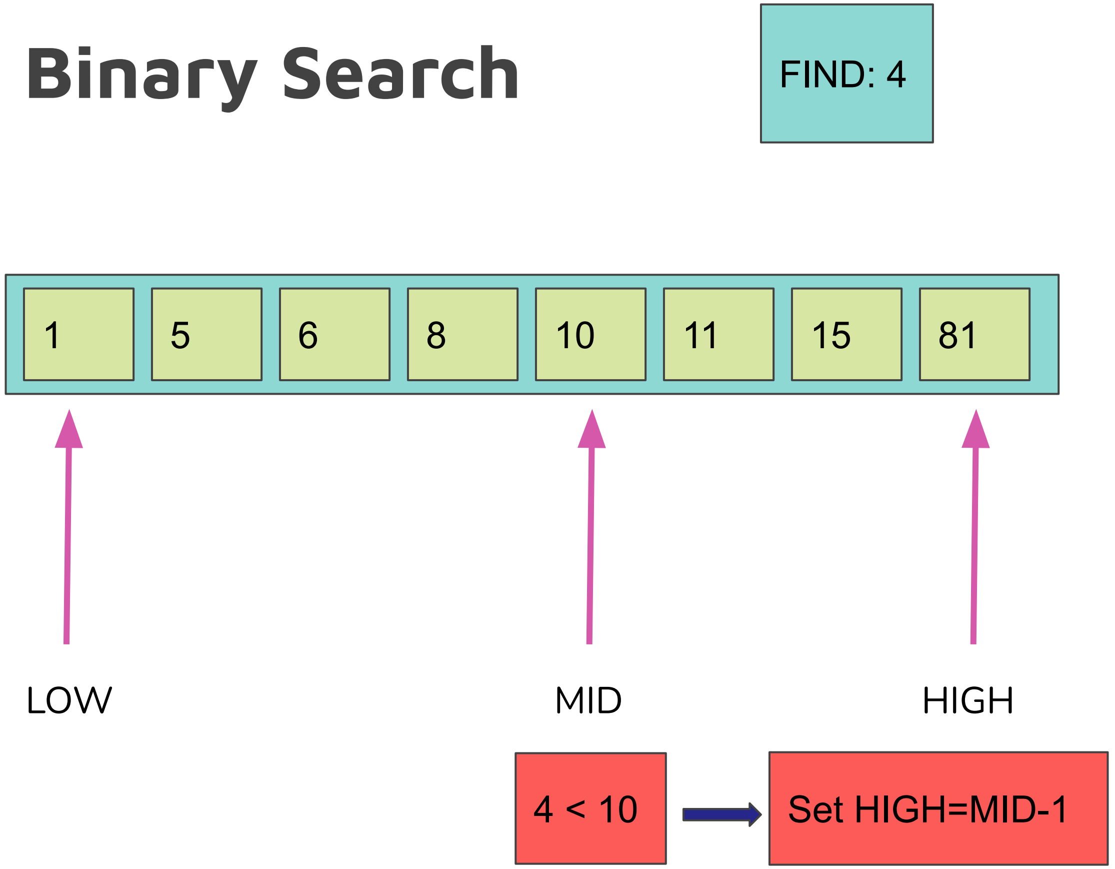
Binary Search

FIND: 4

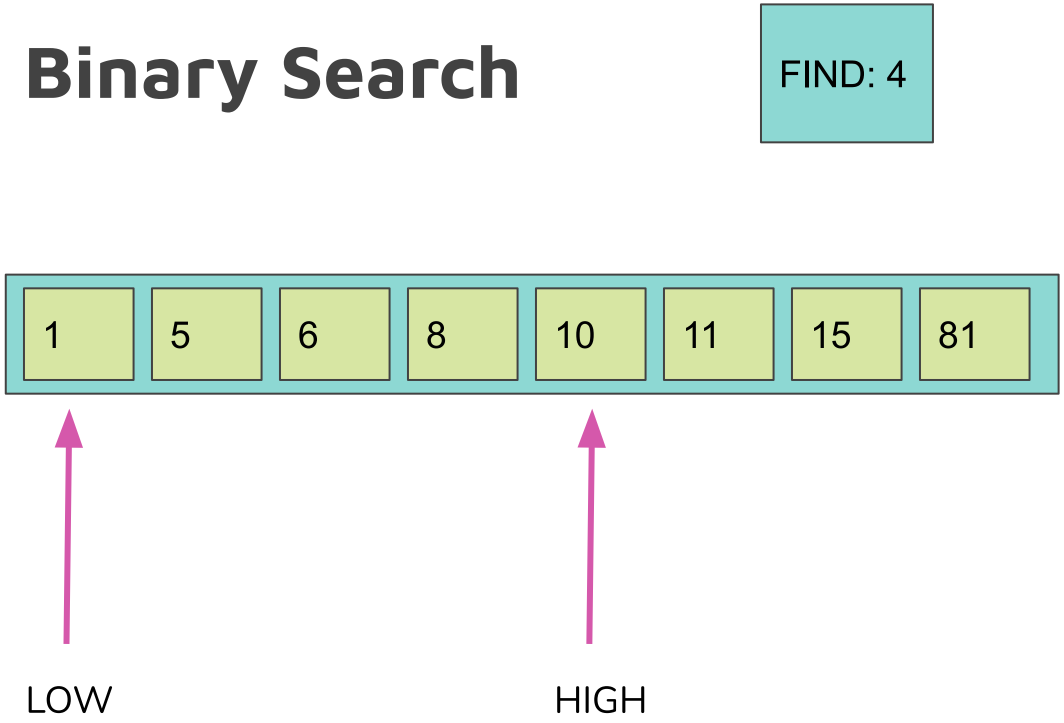


Our special case: is 4 >= lowest number? 4>= 1? Yes -> continue

Binary Search

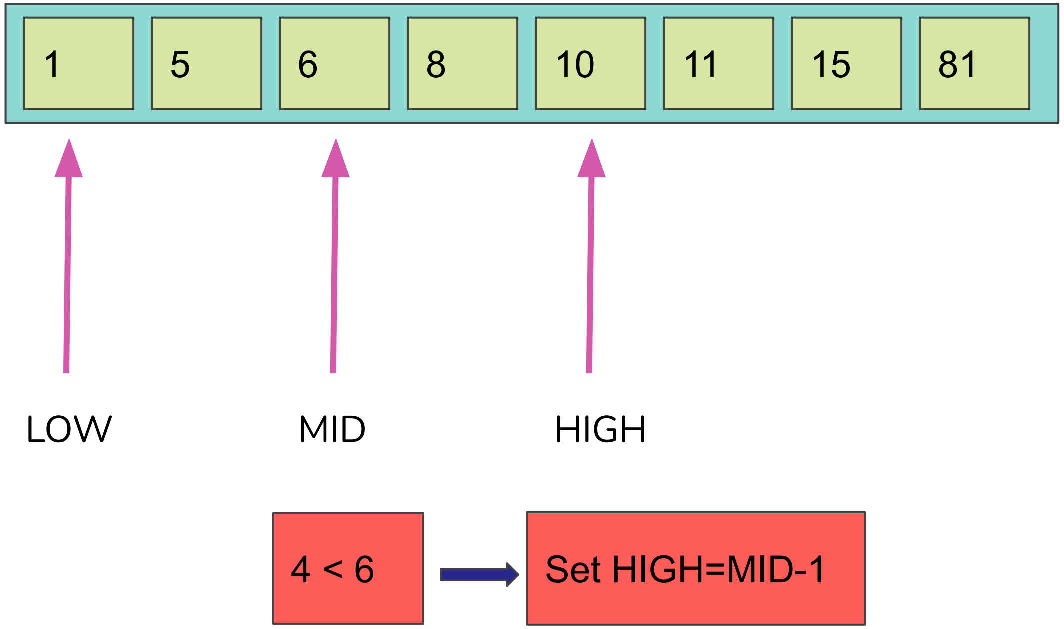


Binary Search

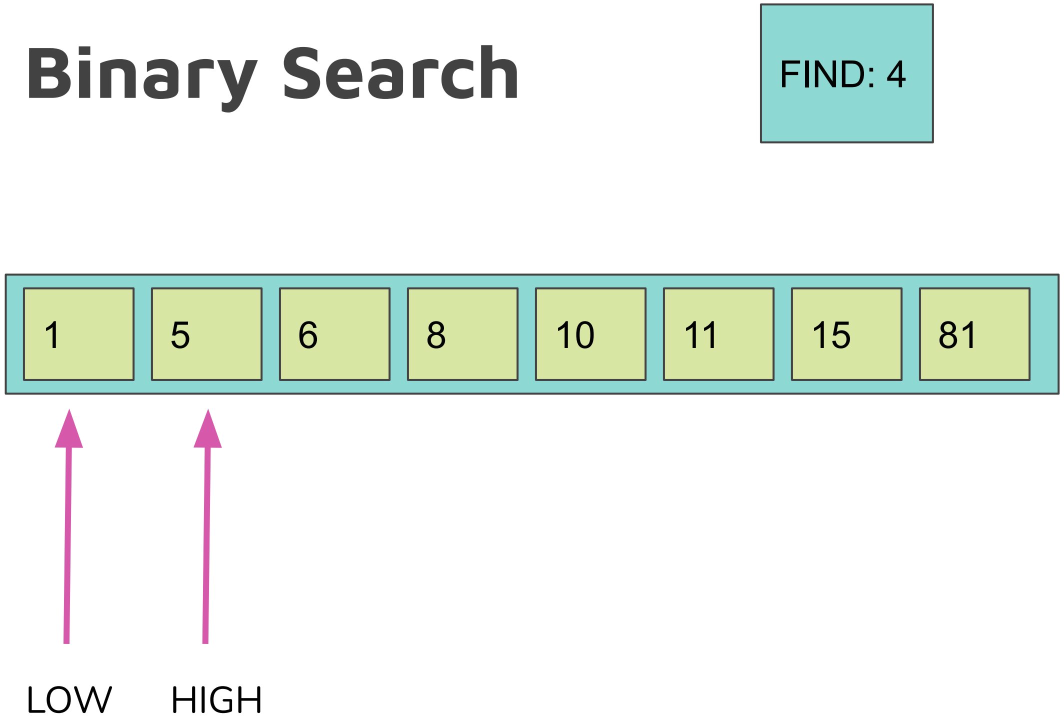


Binary Search

FIND: 4

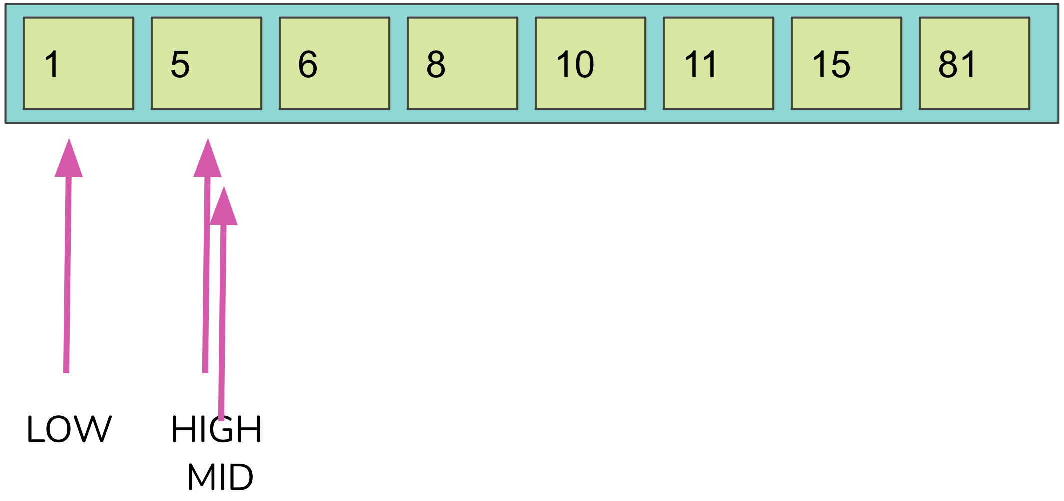


Binary Search



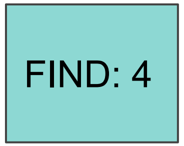
Binary Search

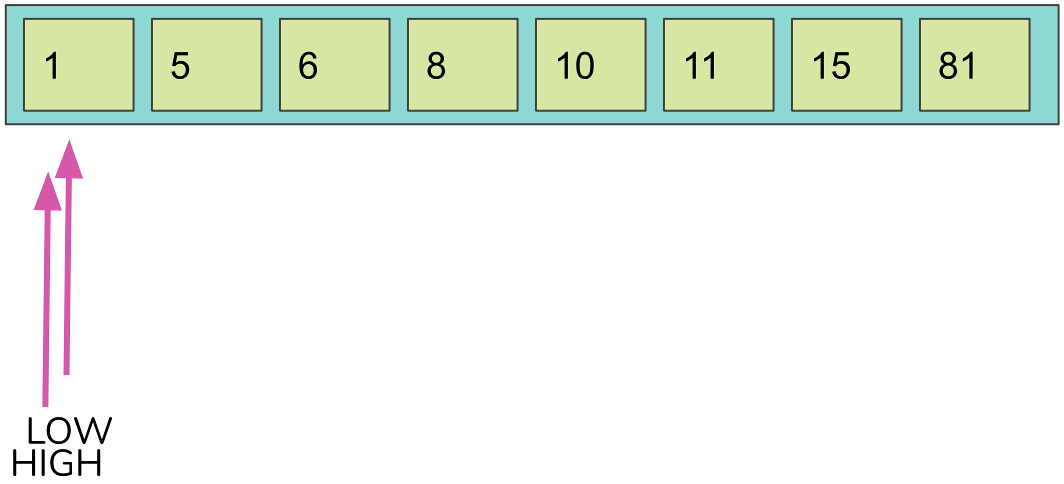
FIND: 4



4 < 5 → Set HIGH=MID-1

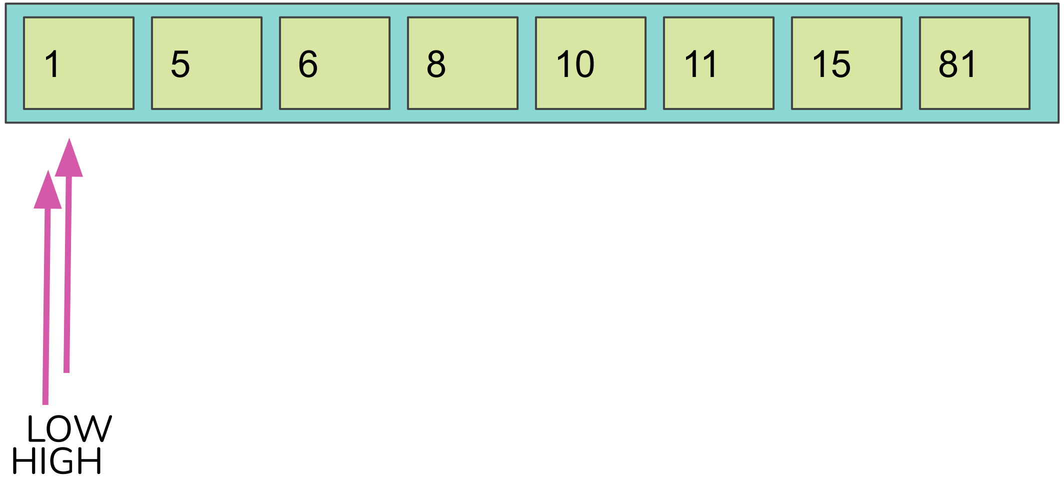
Binary Search





Binary Search

FIND: 4



LOW=HIGH so do this must be the max number less than 4