CS062

DATA STRUCTURES AND ADVANCED PROGRAMMING

24: Symbol Tables and Binary Search

Alexandra Papoutsaki
LECTURES

Mark Kampe
LABS

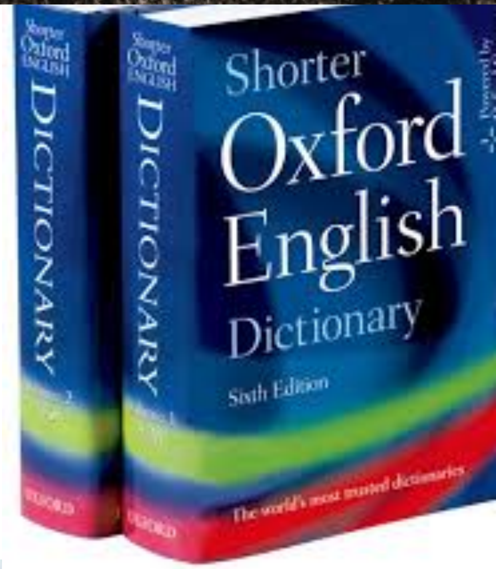
Lecture 24: Symbol Tables and Binary Search

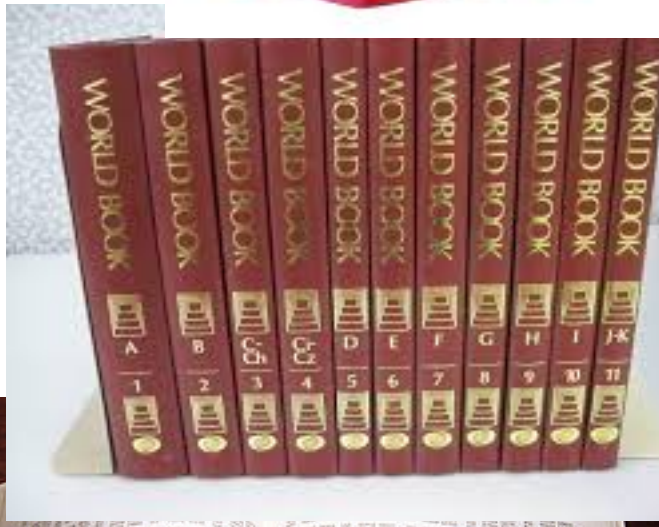
- Symbol Tables
- Binary search
- Elementary Implementations of Symbol Tables
- Ordered Operations

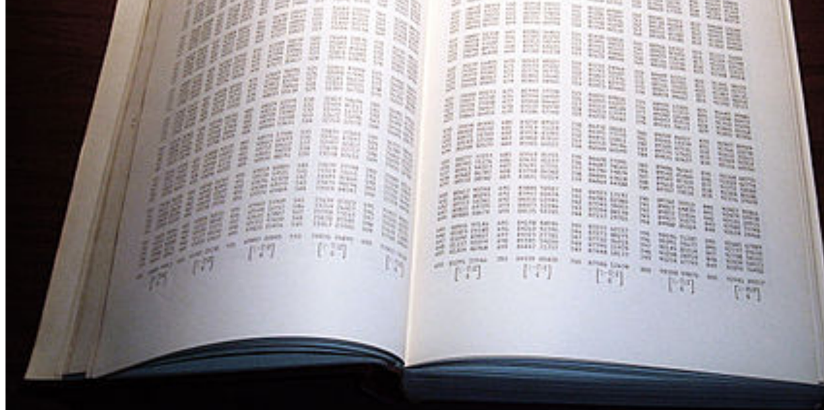
Some slides adopted from Algorithms 4th Edition or COS226

Printed symbol tables are all around us

- Dictionary: key = word, value = definition.
- Encyclopedia: key = term, value = article.
- Phonebook: key = name, value = phone number.
- Math table: key = math functions and input, value = function output.
- Unsupported operations:
  - Add a new key and associated value.
  - Remove a given key and associated value.
  - Change value associated with a given key.







Symbol tables

- Key-value pair abstractions.
  - Insert a value with a specific key.
  - Given a key, search for the corresponding value.
- Also known as: maps, dictionaries, associative arrays.
- Generalize arrays: keys not be integers between 0 and $n - 1$.
- Supported either with built-in or external libraries by the majority of programming languages.

Basic symbol table API

- `public class ST <Key extends Comparable<Key>, Value>`
- `ST()`: create an empty symbol table. By convention, values are not `null`.
- `void put(Key key, Value val)`: insert key-value pair.
    - Overwrites old value with new value if key already exists.
- `Value get(Key key)`: return value associated with key.
    - Returns `null` if key not present.
- `boolean contains(Key key)`: is there a value associated with key.
- `Iterable keys()`: all the keys in the symbol table.
- `void delete(Key key)`: delete key and associated value.
- `boolean isEmpty()`: is the symbol table empty?
- `int size()`: number of key-value pairs.

Lecture 24: Symbol Tables and Binary Search

- Symbol Tables
- Binary search
- Elementary Implementations of Symbol Tables
- Ordered Operations

Binary search

- Goal: Given a sorted array and a key, find index of the key in the array.
- Basic mechanism: Compare key against middle entry.
  - If too small, repeat in left half.
  - If too large, repeat in right half.
  - If equal, you are done.

Binary search implementation

- First binary search published in 1946.
- First bug-free one in 1962.
- Bug in Java's `Arrays.binarySearch()` discovered in 2006 https://ai.googleblog.com/2006/06/extra-extra-read-all-about-it-nearly.html

```java
public static int binarySearch(int[] a, int key) {
    int lo = 0, hi = a.length-1;
    while (lo <= hi) {
         int mid = lo + (hi - lo) / 2;
         if (key < a[mid])
              hi = mid - 1;
         else if (key > a[mid])
              lo = mid + 1;
         else return mid; }
    return -1;
}
```

- Uses at most $1 + \log n$ key compares to search in a sorted array of size n, that is it is $O(\log n)$.

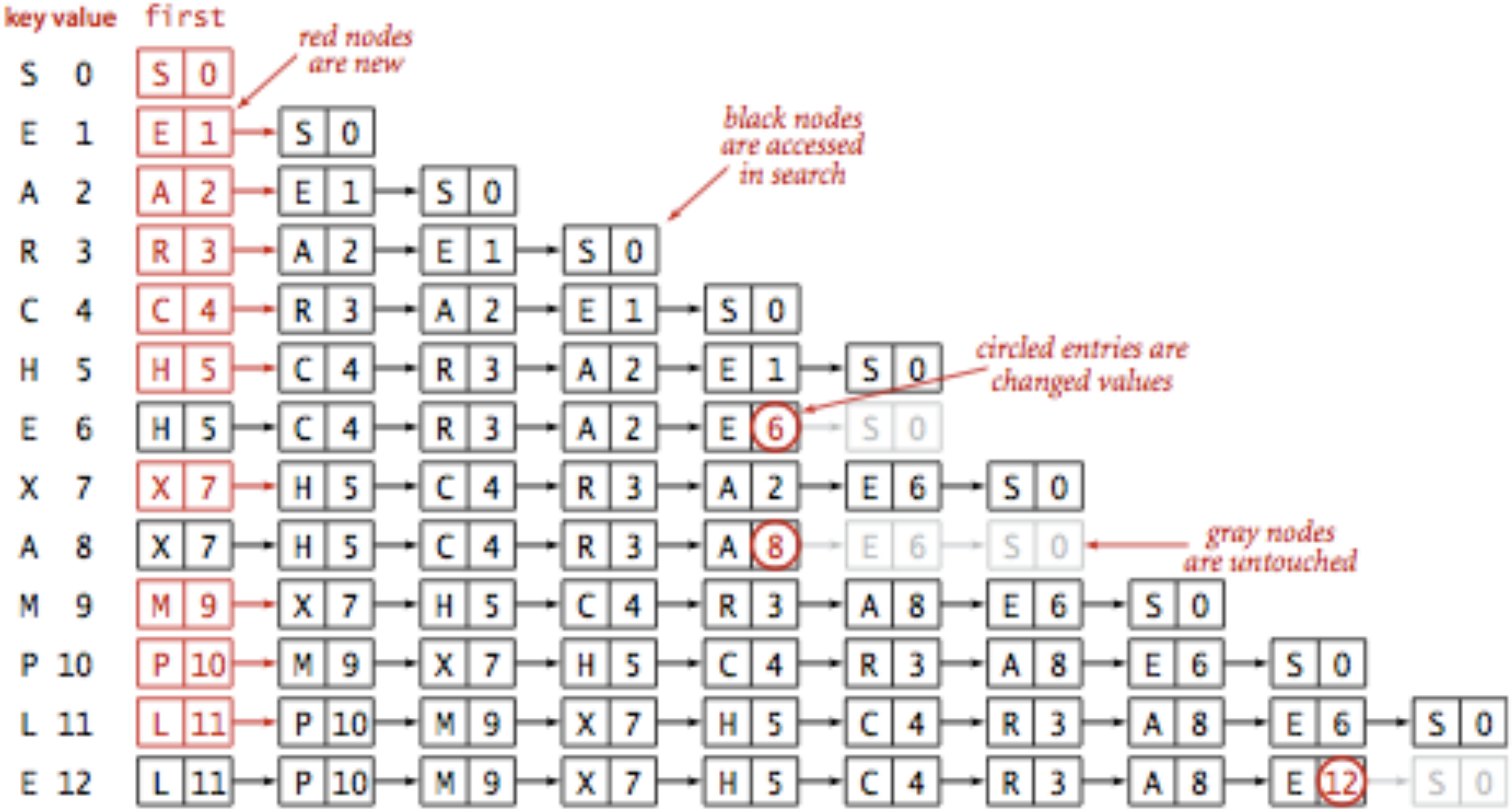
Lecture 24: Symbol Tables and Binary Search

- Symbol Tables
- Binary search
- Elementary Implementations of Symbol Tables
- Ordered Operations

Sequential search in a linked list

- Data structure: Maintain an unordered linked list of key-value pairs.
- Search: Scan through all the keys until you find a match.
- Insert: Scan through all the keys until you find a match. If you found it, update value, otherwise, add to front of list.
- If our cost model counts how many times we will compare keys, both search and insert are $O(n)$ both for worst and average case.

Sequential search in a linked list

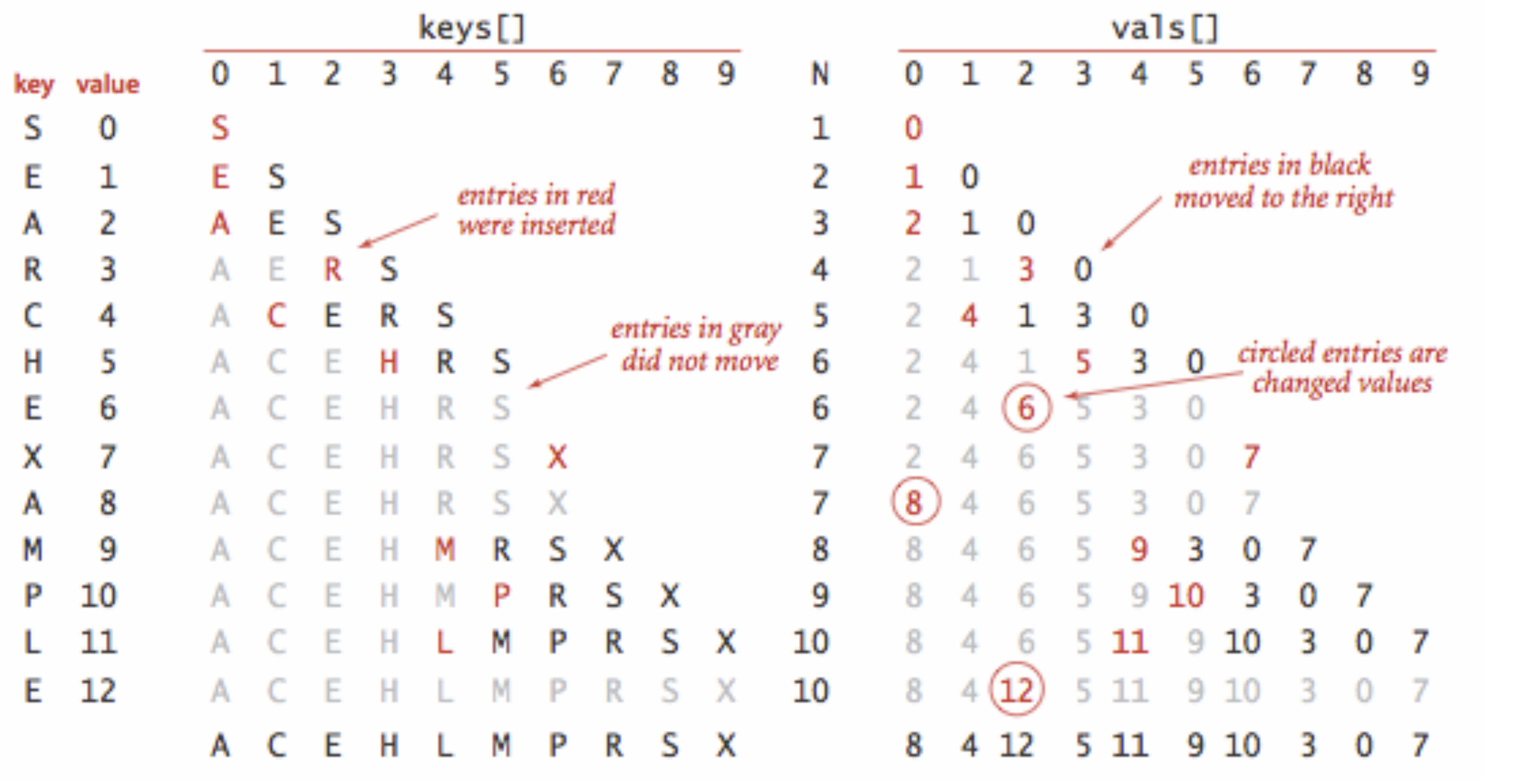

key value
first
red nodes are new
black nodes are accessed in search
circled entries are changed values
gray nodes are untouched
S 0 S 0
E 1 E 1 S 0
A 2 A 2 E 1 S 0
R 3 R 3 A 2 E 1 S 0
C 4 C 4 R 3 A 2 E 1 S 0
H 5 H 5 C 4 R 3 A 2 E 1 S 0
E 6 H 5 C 4 R 3 A 2 E 6 S 0
X 7 X 7 H 5 C 4 R 3 A 2 E 6 S 0
A 8 X 7 H 5 C 4 R 3 A 8 E 6 S 0
M 9 M 9 X 7 H 5 C 4 R 3 A 8 E 6 S 0
P 10 P 10 M 9 X 7 H 5 C 4 R 3 A 8 E 6 S 0
L 11 L 11 P 10 M 9 X 7 H 5 C 4 R 3 A 8 E 6 S 0
E 12 L 11 P 10 M 9 X 7 H 5 C 4 R 3 A 8 E 12 S 0

Trace of linked-list ST implementation for standard indexing client

Binary search in an ordered array

- Data structure: Maintain parallel arrays for keys and values, sorted by keys.
- Search: Use binary search to find key.
    - At most $O(\log n)$ compares to search a sorted array of length n.
- Insert: Use binary search to find key. If it does not exist, shift all larger keys over.
    - At most $O(n)$ time.

Binary search in an ordered array



		keys[]											vals[]									
key	value	0	1	2	3	4	5	6	7	8	9	N	0	1	2	3	4	5	6	7	8	9
S	0	S										1	0									
E	1	E	S									2	1	0								
A	2	A	E	S								3	2	1	0							
R	3	A	E	R	S							4	2	1	3	0						
C	4	A	C	E	R	S						5	2	4	1	3	0					
H	5	A	C	E	H	R	S					6	2	4	1	5	3	0				
E	6	A	C	E	H	R	S					6	2	4	6	5	3	0				
X	7	A	C	E	H	R	S	X				7	2	4	6	5	3	0	7			
A	8	A	C	E	H	R	S	X				7	8	4	6	5	3	0	7			
M	9	A	C	E	H	M	R	S	X			8	8	4	6	5	9	3	0	7		
P	10	A	C	E	H	M	P	R	S	X		9	8	4	6	5	9	10	3	0	7	
L	11	A	C	E	H	L	M	P	R	S	X	10	8	4	6	5	11	9	10	3	0	7
E	12	A	C	E	H	L	M	P	R	S	X	10	8	4	12	5	11	9	10	3	0	7
		A	C	E	H	L	M	P	R	S	X		8	4	12	5	11	9	10	3	0	7

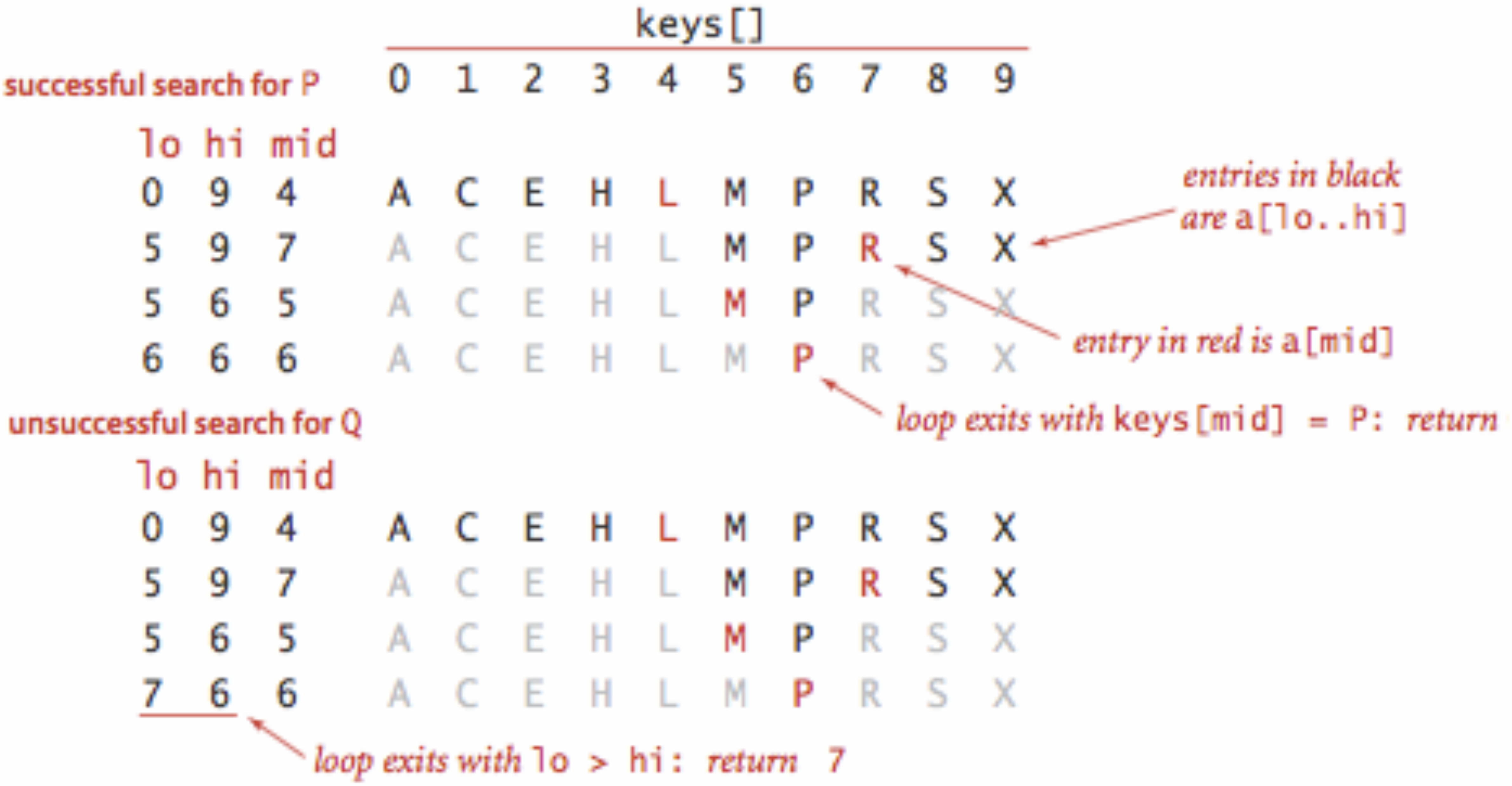
entries in red were inserted

entries in gray did not move

entries in black moved to the right

circled entries are changed values

Binary search in an ordered array



successful search for P

			keys[]									
lo	hi	mid	0	1	2	3	4	5	6	7	8	9
0	9	4	A	C	E	H	L	M	P	R	S	X
5	9	7	A	C	E	H	L	M	P	R	S	X
5	6	5	A	C	E	H	L	M	P	R	S	X
6	6	6	A	C	E	H	L	M	P	R	S	X

entries in black are a[lo..hi]

entry in red is a[mid]

loop exits with keys[mid] = P: return

unsuccessful search for Q

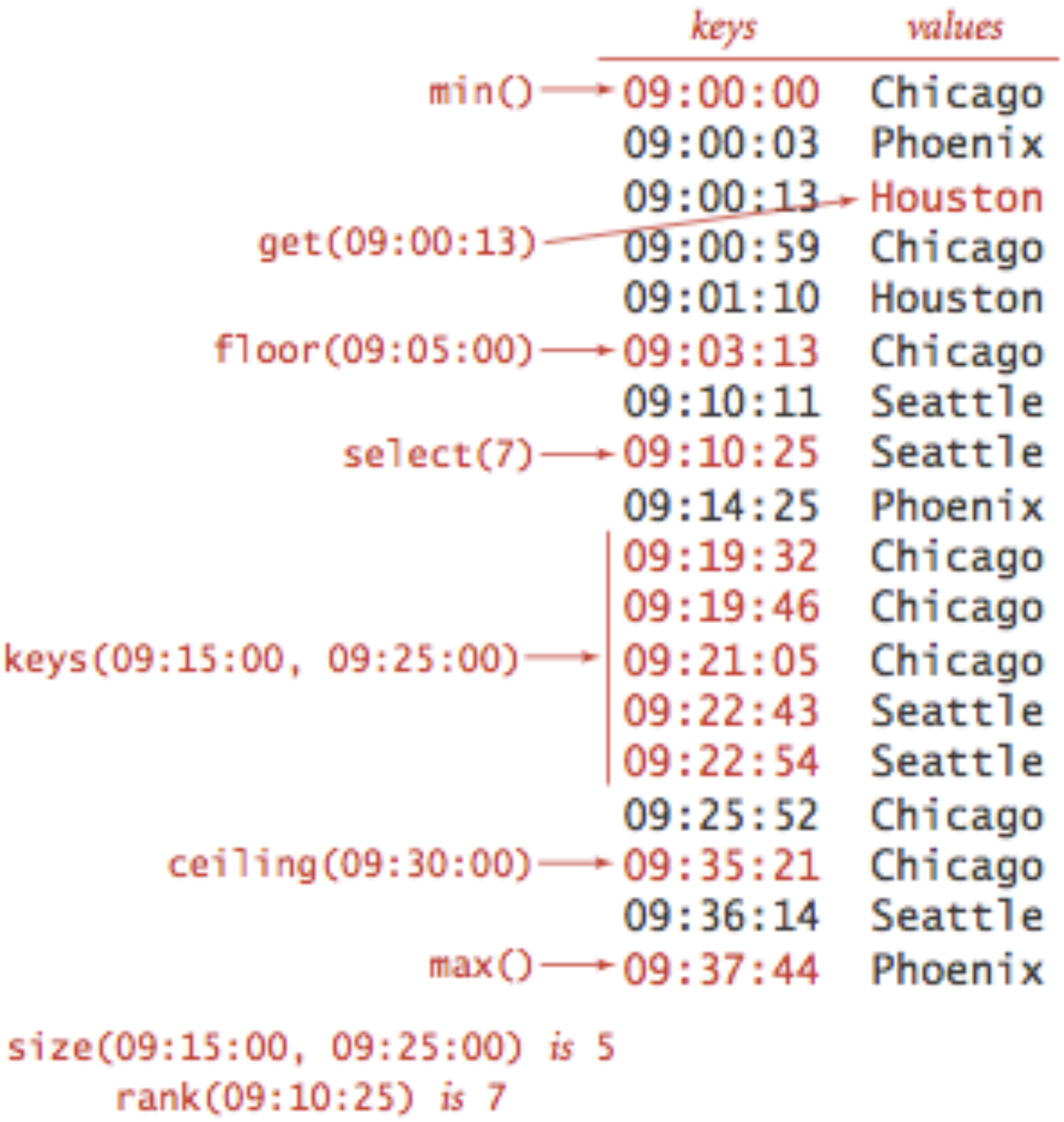
lo	hi	mid	0	1	2	3	4	5	6	7	8	9
0	9	4	A	C	E	H	L	M	P	R	S	X
5	9	7	A	C	E	H	L	M	P	R	S	X
5	6	5	A	C	E	H	L	M	P	R	S	X
7	6	6	A	C	E	H	L	M	P	R	S	X

loop exits with lo > hi: return 7

Lecture 24: Symbol Tables and Binary Search

- Symbol Tables
- Binary search
- Elementary Implementations of Symbol Tables
- Ordered Operations

Examples of ordered operations in a symbol table



	keys	*values*
min()	09:00:00	Chicago
	09:00:03	Phoenix
	09:00:13	Houston
get(09:00:13)	09:00:59	Chicago
	09:01:10	Houston
floor(09:05:00)	09:03:13	Chicago
	09:10:11	Seattle
select(7)	09:10:25	Seattle
	09:14:25	Phoenix
	09:19:32	Chicago
	09:19:46	Chicago
keys(09:15:00, 09:25:00)	09:21:05	Chicago
	09:22:43	Seattle
	09:22:54	Seattle
	09:25:52	Chicago
ceiling(09:30:00)	09:35:21	Chicago
	09:36:14	Seattle
max()	09:37:44	Phoenix

size(09:15:00, 09:25:00) *is* 5

rank(09:10:25) *is* 7

Ordered symbol table API

- `Key min()`: smallest key.
- `Key max()`: largest key.
- `Key floor(Key key)`: largest key less than or equal to given key.
- `Key ceiling(Key key)`: smallest key greater than or equal to given key.
- `int rank(Key key)`: number of keys less that given key.
- `Key select(int k)`: key with rank `k`.
- `Iterable keys()`: all keys in symbol table in sorted order.
- `Iterable keys(int lo, int hi)`: keys in `[lo, …, hi]` in sorted order.

Order of growth for ordered symbol table operations

	Sequential search	Binary search
search	n	$\log n$
insert	n	n
min/max	n	1
floor/ceiling	n	$\log n$
rank	n	$\log n$
select	n	1

Lecture 24: Symbol Tables and Binary Search

- Symbol Tables
- Binary search
- Elementary Implementations of Symbol Tables
- Ordered Operations

Readings:

- Textbook: Chapter 3.1 (Pages 362-386)
- Website:
  - https://algs4.cs.princeton.edu/31elementary/

Practice Problems:

- 3.1.1-3.1.6