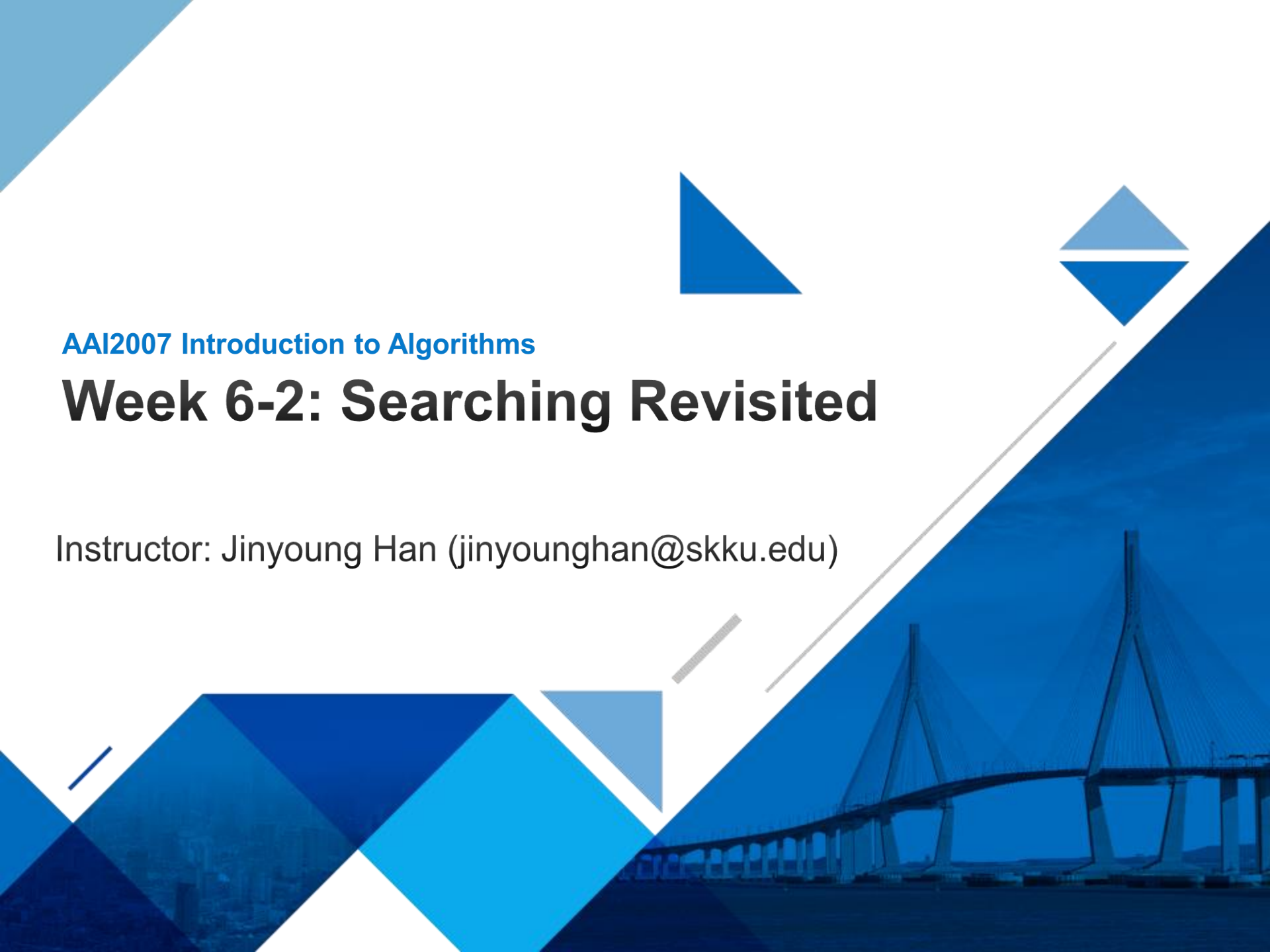
AAI2007 Introduction to Algorithms

Week 6-2: Searching Revisited

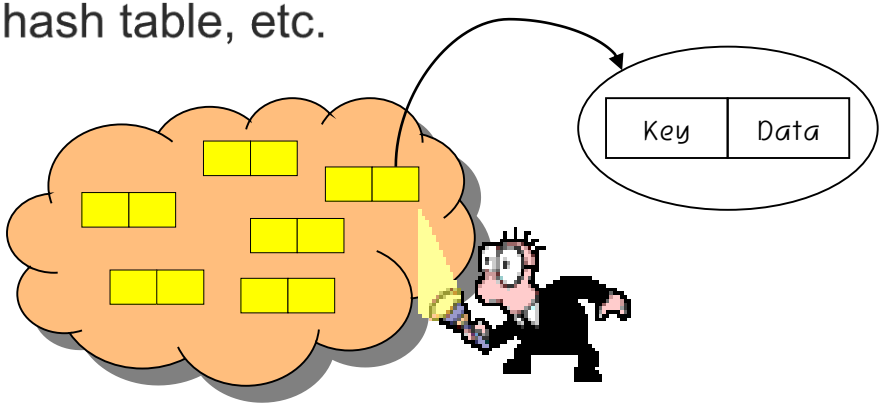
Instructor: Jinyoung Han (jinyounghan@skku.edu)



Searching Revisited

Searching

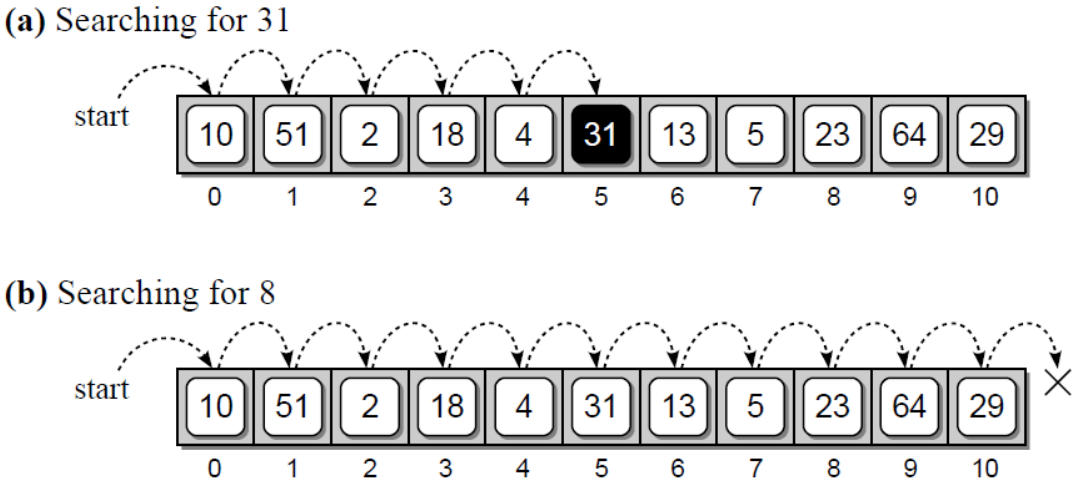
- Searching
  - Find a data from a bunch of data
  - One of the most common tasks by computer
    - Efficient searching is very important!
- Search key
  - Identifier of each data
- Data structures for searching
  - Array, linked list, tree, graph, hash table, etc.



Searching Revisited

Linear Search

- Linear search
  - Iterates over the sequence, one item at a time, until the specific item is found or all items have been examined
  - Time complexity: O(n)




```
def linearSearch( theValues, target ) :
  n = len( theValues )
  for i in range( n ) :
    if theValues[i] == target
      return True
  return False
```

Performing a linear search on an unsorted array
(a) the target item is found
(b) the item is not in the array

Searching Revisited

Binary Search

- Binary search
  - Binary search algorithm can be applied ***to a sorted sequence***
  - It starts by examining the middle item of the sorted sequence, resulting in one of three possible conditions:
    - the middle item is the target value,
    - the target value is less than the middle item,
    - or the target is larger than the middle item.
  - Since the sequence is ordered, we can eliminate half the values in the list when the target value is not found at the middle position
- Example: finding a name among 1 billion people
  - Linear search: 0.5 billion comparisons on average
  - Binary search: 30 comparisons

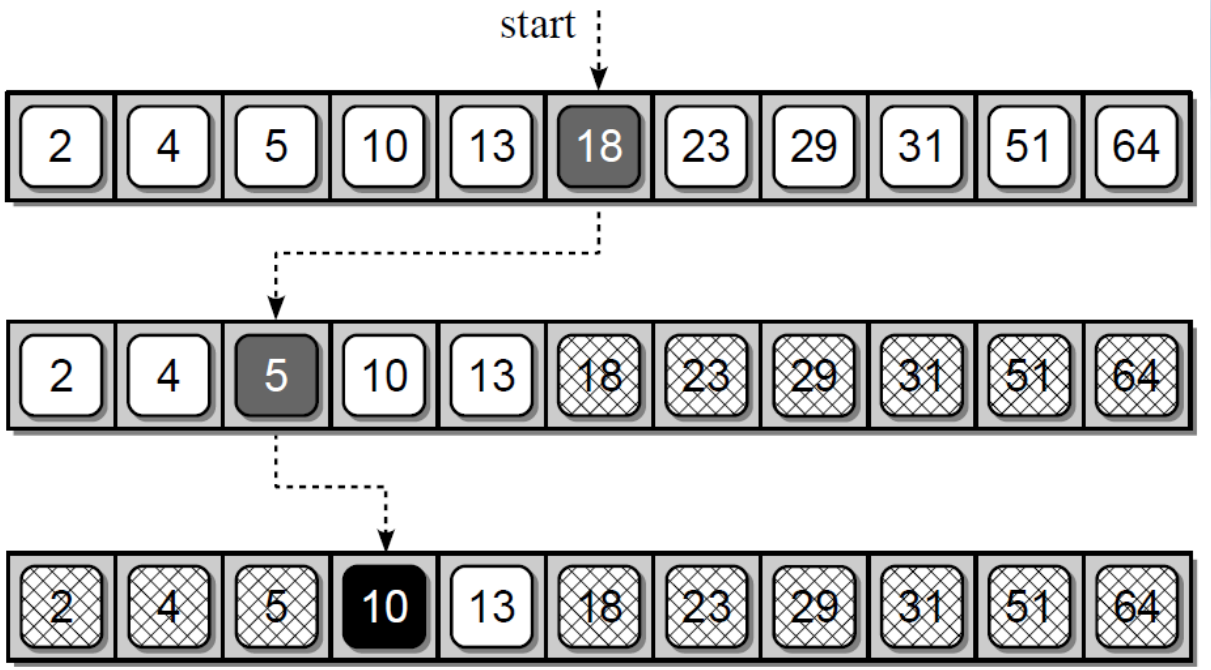
Searching Revisited

Divide-and-Conquer

- Divide and Conquer
  - Divide a larger problem into smaller parts, and conquer the smaller part
  - Recursively break down a problem into two or more sub-problems of the same or related type, until these become simple enough to be solved directly
    - The solutions to the sub-problems are then combined to give a solution to the original problem
- Examples
  - Binary search
  - Merge sort
  - Quick sort
  - Median, closest pairs, Hanoi tower, Fibonacci numbers, …

Searching Revisited

Example



Searching for 10 in a sorted array using the binary search

Searching Revisited

Example

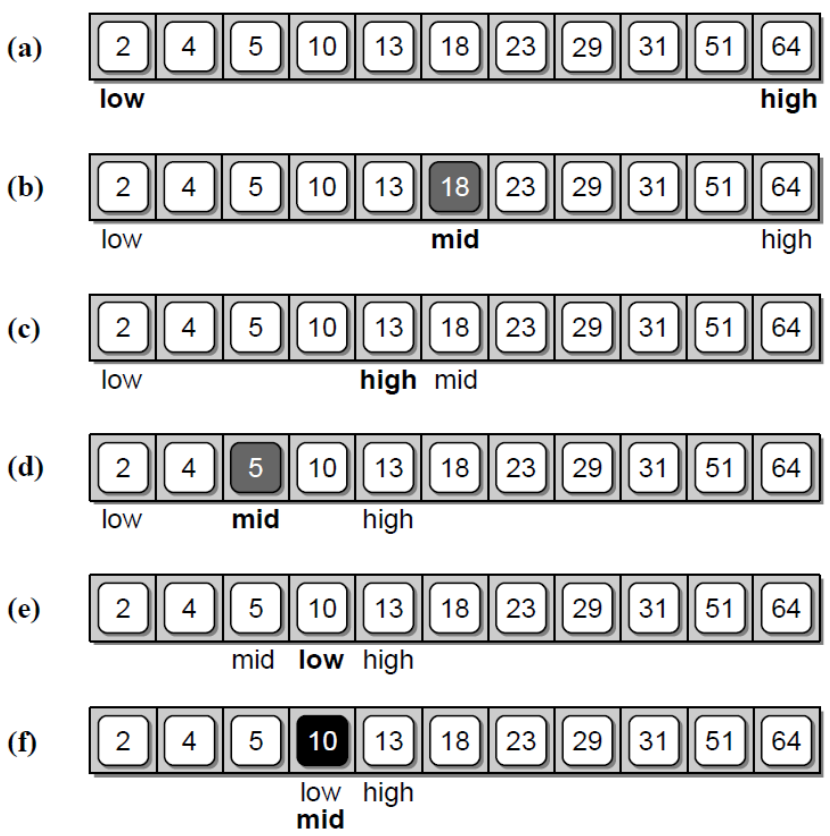
```
def binarySearch( theValues, target ) :
  low = 0
  high = len(theValues) - 1

  while low <= high :
    mid = (high + low) // 2
    if theValues[mid] == target :
      return True
    elif target < theValues[mid] :
      high = mid - 1
    else :
      low = mid + 1

 return False
```

Searching Revisited

Example



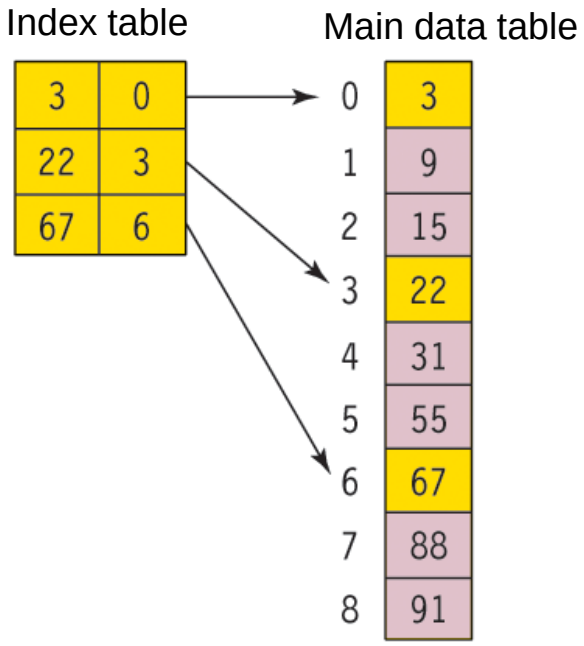
Searching for 10:

(a) initial range of items
(b) locating the midpoint
(c) eliminating the upper half
(d) Midpoint of the lower half
(e) eliminating the lower fourth
(f) finding the target item

Searching Revisited

Indexed Sequential Search

- Indexed sequential search
  - Using an index table for efficient searching
    - A main data table + index table
      - Both of them are ordered
  - Complexity
    - O(m+n/m)
      - index table size = m
      - main data table size =n



What You Need to Know

Summary

- Linear search
- Binary search
- Indexed sequential search

Thanks

Week 6-2: Searching Revisited

Instructor: Jinyoung Han (jinyounghan@skku.edu)