ThS.Trần Lê Như Quỳnh

ALGORITHMS

DATA STUCTURE

Timing an algorithm


```
long startTime = System.currentTimeMillis( );          // record the starting time
/* (run the algorithm) */
long endTime = System.currentTimeMillis( );            // record the ending time
long elapsed = endTime − startTime;                    // compute the elapsed time
```

Code Fragment 4.1: Typical approach for timing an algorithm in Java.

The "Big-Oh" Notation

- Read in book page 168
- Let f(n) and g(n) be functions mapping positive integers to positive real numbers. We say that f(n) is O(g(n)) if there is a real constant c > 0 and an integer constant n0 ≥ 1 such that **f(n) ≤ c ·g(n),** for n ≥ n0
- The big-Oh notation allows us to ignore constant factors and lower-order terms and focus on the main components of a function that affect its growth
- Example: 5n4 +3n3 +2n2 +4n+1 is O(n4)

ThS.Trần Lê Như Quỳnh

ThS.Trần Lê Như Quỳnh

SEARCH ALGORITHMS

ALGORITHMS

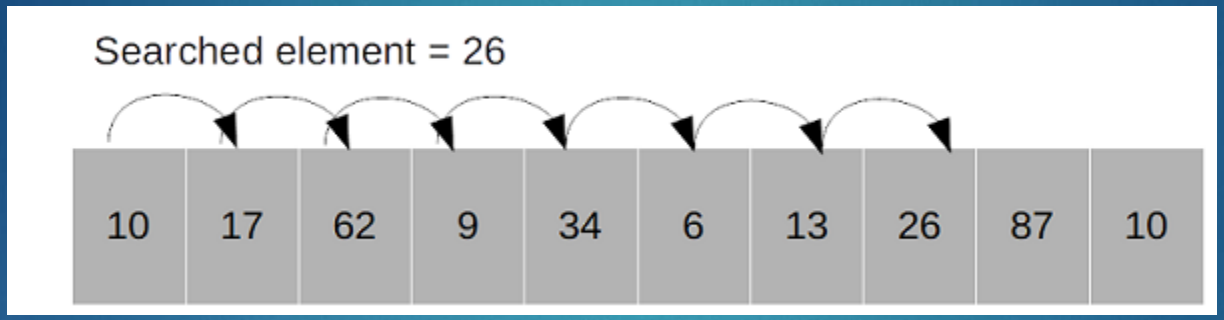
ThS.Trần Lê Như Quỳnh

Linear Search

SEARCH ALGORITHMS

LINEAR SEARCH

ThS.Trần Lê Như Quỳnh

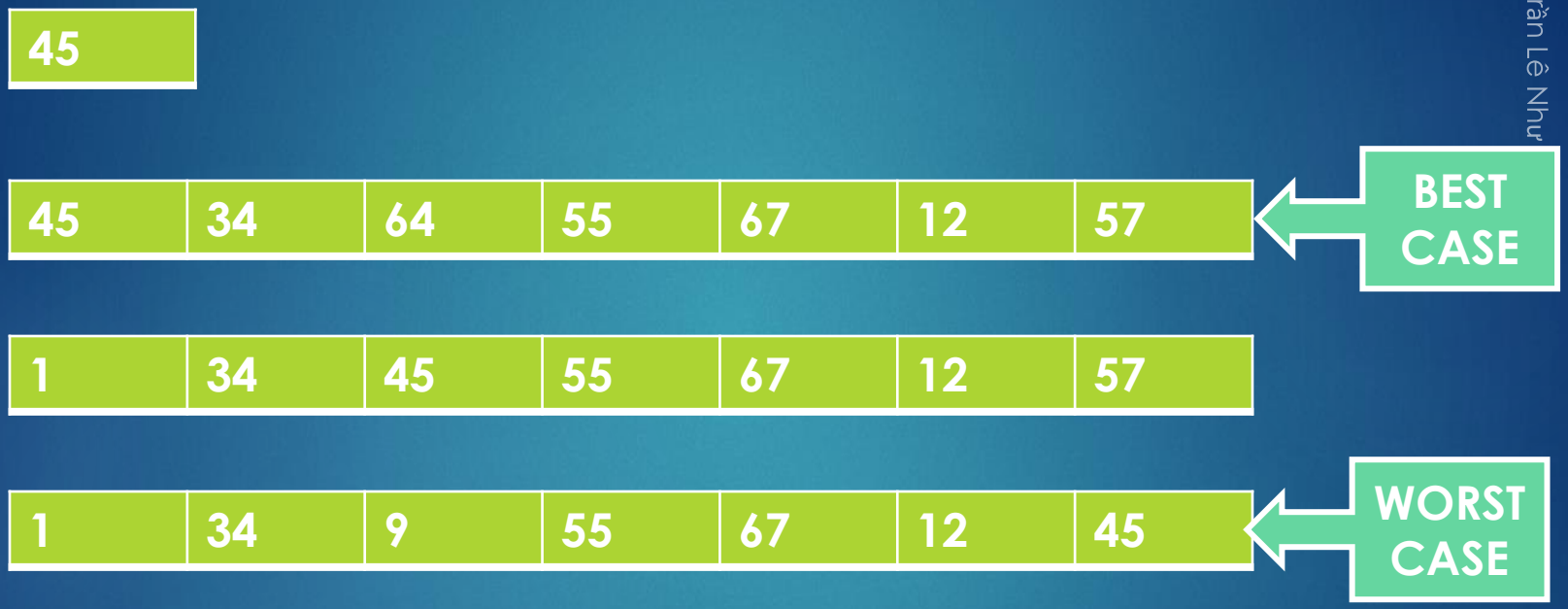


RULE

ThS.Trần Lê Như Quỳnh

- LinearSearch(A , size, target)
  - For i=0 to size -1
  - If(A[i] == target) return i
  - Else return -1

EXAMPLE



RUNNING TIME

ThS.Trần Lê Như Quỳnh

- Best case: 1 comparable time
- Worst case: n comparable times
- AVG case: (n+1) /2 comparable times
- O(n)

IMPLEMENT LINEAR SEARCH

ThS.Trần Lê Như Quỳnh

- public boolean linearSearching(int[] array, int target){
- for(….){
- If(array[i]== target){
- //TODO
- }

}

USING NON-RECURSIVE TO PROGRAM LINEAR SEARCH

ThS.Trần Lê Như Quỳnh

```
public boolean linearSearching(int[] array, int target){
  for(....){
  If(array[i]== target){
  //TODO
  }
}
```

USING RECURSIVE TO PROGRAM LINEAR SEARCH

ThS.Trần Lê Như Quỳnh

- public boolean linearSearching(int[] array, int target, int cursor)

```
  {
        if(cursor == array.length -1){
        // STOP CONDITION
        }else{
        // RECURSIVE CONDITION
        // if array[cursor] == target
        // if array[cursor] != target
        }
}
```

ThS.Trần Lê Như Quỳnh

Binary Search

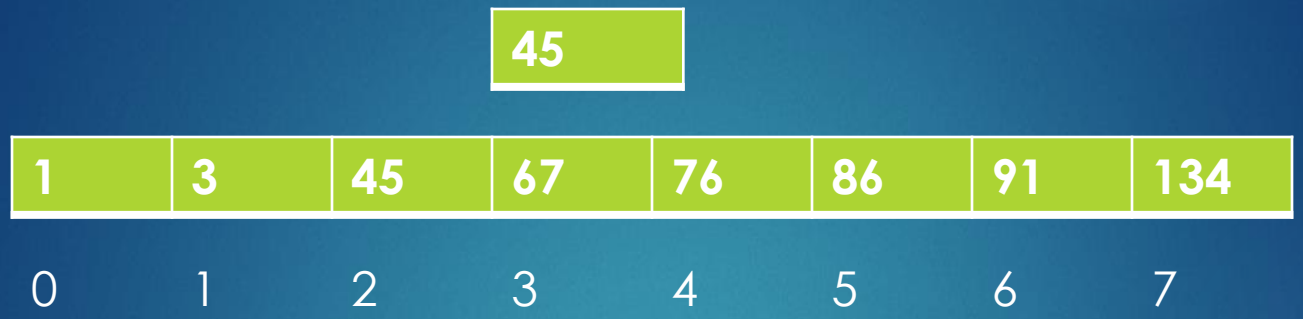
SEARCH ALGORITHM

HOW IT WORK ?

- Compare x with the middle element.
- If x matches with middle element, we return the mid index.
- Else If x is greater than the mid element, then x can only lie in right half subarray after the mid element. So we recur for right half.
- Else (x is smaller) recur for the left half.

ThS.Trần Lê Như Quỳnh

EXAMPLE



Step1 : size of array => N= 8; high = N-1 =7, low =0, mid = (low +high)/2 = 7/2 = 3
Array[3] =67 > 45 => high = mid -1 = 3-1 = 2

Step2 : high =2, low =0, mid = (low +high)/2 = 2/2 =1
Array[1] =3 <45 => low = mid +1 = 1+1 = 2

Step3: high =2,low =2 => mid =(low +high) /2 = 4/2 =2
Array[2] =45 =45 ->| stop

RUNNING TIME

ThS.Trần Lê Như Quỳnh

- Best: ceil($\log_2(n)$) +1
- Worst: floor($\log_2(n)$) +1
- AVG: approx.$\log_2(n)$ +1
- O(log(n))

USING RECURSIVE TO IMPLEMENT BINARY SEARCH

- Just using for sorted array
- Low = begin of array, high = end of array, mid = (high + low)/2
- STOP CONDITION
  - Target equals with value of element at middle
- CONTINIOUS CONDITION
  - Target larger than mid → Recursive with low= mid+1, high not change
  - Target smaller than mid → Recursive with low not change, high = mid -1

USING RECURSIVE TO IMPLEMENT BINARY SEARCH


```
/**
 * Returns true if the target value is found in the indicated portion of the data array.
 * This search only considers the array portion from data[low] to data[high] inclusive.
 */
public static boolean binarySearch(int[ ] data, int target, int low, int high) {
  if (low > high)
    return false;                                          // interval empty; no match
  else {
    int mid = (low + high) / 2;
    if (target == data[mid])
      return true;                                         // found a match
    else if (target < data[mid])
      return binarySearch(data, target, low, mid - 1);     // recur left of the middle
    else
      return binarySearch(data, target, mid + 1, high);    // recur right of the middle
  }
}
```

NO USING RECURSIVE TO IMPLEMENT BINARY SEARCH

ThS.Trần Lê Như Quỳnh

```
BinarySearch(A[0..N-1], value) {
    low = 0
    high = N - 1
    while (low <= high) {
    mid = (high + low) / 2)
    if (A[mid] = value) return value;
    else if (A[mid] > value) high = mid - 1
    else
    low = mid + 1
    }
    return -1111;// no element in array equals value
}
```

Exercise

- Coding Linear search
- Coding Binary search

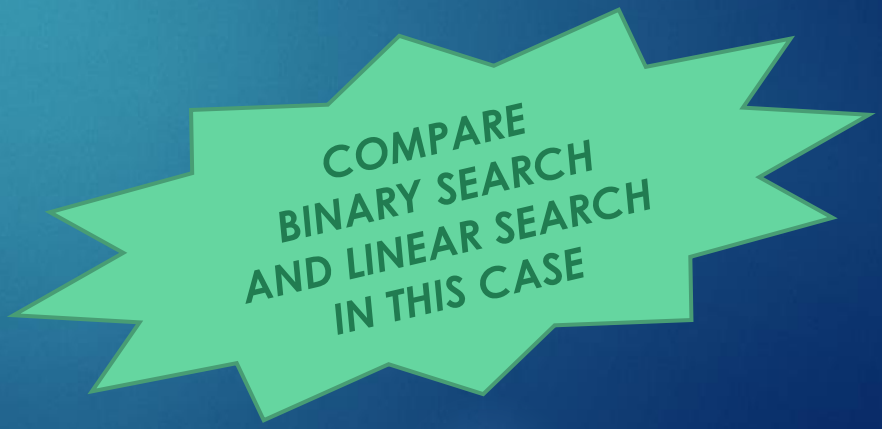
ThS.Trần Lê Như Quỳnh

Challenge 1

ThS.Trần Lê Như Quỳnh

- Using Binary search vs linear search, run by step
  - Finding target number = 7
  - Finding target number = 5

4	6	7	5	8	9	10	12	15



Challenge 2

ThS.Trần Lê Như Quỳnh

- Using Binary search vs linear search, run by step
  - Finding target number = 120
  - Finding target number = 56

23	45	56	67	78	92	101	120	135

COMPARE BINARY SEARCH AND LINEAR SEARCH IN THIS CASE