CANTHO UNIVERSITY

Chapter 3: Basic Sort and Search
Part 3.2: Searching

Lâm Hoài Bảo - FSE – CICT
Trương Minh Thái – FSE - CICT

1

CANTHO UNIVERSITY

Agenda

- Introduction
- Linear search
- Binary search
- Summary

2

CANTHO UNIVERSITY

What is searching?

- The process of finding an element with a special property in a collection
  - Search a record in a database
  - Search a word in a dictionary
  - Search an element in a list (array)
  - …
- Problem
  - Input:
    - A sequence of n elements: $<a_0, a_1, \ldots, a_{n-1}>$
    - An item x
  - Output: Position of x in the sequence

CANTHO UNIVERSITY

Agenda

- Introduction
- Linear search
- Binary search
- Summary

Search in an array

- Give an array A, find the first occurrence of an item x
- Example: A = [7, 2, 5, 3, 2]: search(x = 2, A) → 1

- Algorithm
  - Start from the first position, compare the element at that position to x.
    - If equal, return that position
    - Else, move the next position

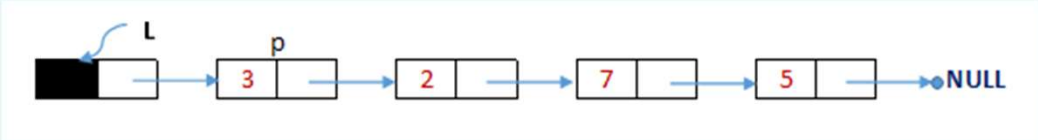
Algorithm on array

```
#Find an item in an array A of n elements
ALGORITHM linearSearch(x, A, n):
    for pos = 0 to n-1:
        if A[pos].key == x # Compare
            return pos
    return -1 #not found
```

- Complexity
  - In the worst case, there are n comparisons: T(n) = O(n)

Search in a linked list

- Give a linked list L, return pointer referring to the first node whose value is x
- Example: search(x = 2, L) → p



- Algorithm
  - Start from the first node's pointer p, compare the node referred by p->next to x.
    - If equal, return p
    - Else, move the next position

Algorithm on linked list

```
#Find an item x in a linked list
ALGORITHM linearSearch(x, L):
    p = L
    while p->next != NULL:
        if p->next->data == x: #Compare
            return p
        p = p->next

    return p #the pointer of the last node
```

- Complexity
  - In the worst case, there are n comparisons: T(n) = O(n)

CANTHO UNIVERSITY

Agenda

- Introduction
- Linear search
- Binary search
- Summary

CANTHO UNIVERSITY

Binary search

- Give an ascending sorted array of size n, find the position of a given item x.
- Intuition: Compare the target value to the middle element:
  - If equal, return the middle position
  - If not, a half of the array that the target can not lie is eliminated, repeat searching on the other side

CANTHO UNIVERSITY

Example 1

- Search 45 Result

5	9	13	16	18	28	31	35	37	38	45	48
0	1	2	3	4	5	6	7	8	9	10	11

CANTHO UNIVERSITY

Example 1

- Search 45 Result

start 0 mid end 11

5	9	13	16	18	28	31	35	37	38	45	48
0	1	2	3	4	5	6	7	8	9	10	11

Example 1

- Search 45 Result

start 0 mid 5 end 11

5	9	13	16	18	28	31	35	37	38	45	48
0	1	2	3	4	5	6	7	8	9	10	11

Example 1

- Search 45 Result

start 6 mid 5 end 11

5	9	13	16	18	28	31	35	37	38	45	48
0	1	2	3	4	5	6	7	8	9	10	11

CANTHO UNIVERSITY

Example 1

- Search 45 Result

start 6 mid 8 end 11

5	9	13	16	18	28	31	35	37	38	45	48
0	1	2	3	4	5	6	7	8	9	10	11

CANTHO UNIVERSITY

Example 1

- Search 45 Result

start 9 mid 5 end 11

5	9	13	16	18	28	31	35	37	38	45	48
0	1	2	3	4	5	6	7	8	9	10	11

Example 1

- Search 45 Result

start 9 mid 10 end 11

5	9	13	16	18	28	31	35	37	38	45	48
0	1	2	3	4	5	6	7	8	9	10	11

Example 1

- Search 45 Result 10

start 9 mid 10 end 11

5	9	13	16	18	28	31	35	37	38	45	48
0	1	2	3	4	5	6	7	8	9	10	11

CANTHO UNIVERSITY

Example 2

- Search 4 Result

5	9	13	16	18	28	31	35	37	38	45	48
0	1	2	3	4	5	6	7	8	9	10	11

CANTHO UNIVERSITY

Example 2

- Search 4 Result

start 0 mid end 11

5	9	13	16	18	28	31	35	37	38	45	48
0	1	2	3	4	5	6	7	8	9	10	11

CANTHO UNIVERSITY

Example 2

- Search 4 Result

start 0 mid 5 end 11

5	9	13	16	18	28	31	35	37	38	45	48
0	1	2	3	4	5	6	7	8	9	10	11

CANTHO UNIVERSITY

Example 2

- Search 4 Result

start 0 mid 5 end 4

5	9	13	16	18	28	31	35	37	38	45	48
0	1	2	3	4	5	6	7	8	9	10	11

CANTHO UNIVERSITY

Example 2

- Search 4 Result

start 0 mid 2 end 4

5	9	13	16	18	28	31	35	37	38	45	48
0	1	2	3	4	5	6	7	8	9	10	11

CANTHO UNIVERSITY

Example 2

- Search 4 Result

start 0 mid 2 end 1

5	9	13	16	18	28	31	35	37	38	45	48
0	1	2	3	4	5	6	7	8	9	10	11

CANTHO UNIVERSITY

Example 2

- Search 4 Result

start 0 mid 0 end 1

5	9	13	16	18	28	31	35	37	38	45	48
0	1	2	3	4	5	6	7	8	9	10	11

CANTHO UNIVERSITY

Example 2

- Search 4 Result

start 0 mid 0 end -1

5	9	13	16	18	28	31	35	37	38	45	48
0	1	2	3	4	5	6	7	8	9	10	11

Example 2

- Search 4 Result -1

start 0 mid 0 end -1

5	9	13	16	18	28	31	35	37	38	45	48
0	1	2	3	4	5	6	7	8	9	10	11

Iterative algorithm

```
ALGORITHM binarySearch(x, A, start, end):
    while start <= end:
        mid = (start + end)/2
        if A[mid].key == x:
            return mid
        else if A[mid].key < x:
            end = mid - 1
        else:
            start = mid + 1
    return -1
```

- Each iteration: one comparison, eliminate a half of elements
- In the worst case: ~ logn comparisons → T(n) = O(logn)

Recursion version

```
ALGORITHM binarySearch(x, A, start, end):
    if start <= end:
        mid = (start + end)/2
        if x == A[mid]:
            return mid
        else if x < A[mid]:
            return binarySearch(x, A, start, mid - 1)
        else:
            return binarySearch(x, A, mid + 1, end)
    else:
        return -1
```


Recursion tree

Search	45	Result	
start	0	mid	end 11

n											
5	9	13	16	18	28	31	35	37	38	45	48

Recursion tree

Search 45 Result

start 0 mid 5 end 11

n	5	9	13	16	18	28	31	35	37	38	45	48

31

Recursion tree

Search 45 Result

start 6 mid 5 end 11

n	5	9	13	16	18	28	31	35	37	38	45	48
n/2	5	9	13	16	18	28	31	35	37	38	45	48

32

Recursion tree

Search 45 Result

start 6 mid 8 end 11

n	5	9	13	16	18	28	31	35	37	38	45	48
n/2	5	9	13	16	18	28	31	35	37	38	45	48

Recursion tree

Search 45 Result

start 9 mid 8 end 11

n	5	9	13	16	18	28	31	35	37	38	45	48
n/2	5	9	13	16	18	28	31	35	37	38	45	48
n/4	5	9	13	16	18	28	31	35	37	38	45	48

Recursion tree

Search 45 Result

start 9 mid 10 end 11

n	5	9	13	16	18	28	31	35	37	38	45	48
n/2	5	9	13	16	18	28	31	35	37	38	45	48
n/4	5	9	13	16	18	28	31	35	37	38	45	48

35

Recursion tree

Search 45 Result 10

start 9 mid 10 end 11

n	5	9	13	16	18	28	31	35	37	38	45	48
n/2	5	9	13	16	18	28	31	35	37	38	45	48
n/4	5	9	13	16	18	28	31	35	37	38	45	48

36

Recursion tree

Search 45 **Result** 10

start 9 **mid** 10 **end** 11

n	5	9	13	16	18	28	31	35	37	38	45	48
n/2	5	9	13	16	18	28	31	35	37	38	45	48
n/4	5	9	13	16	18	28	31	35	37	38	45	48

~logn calls

- ~ logn function calls, each costs O(1) → T(n) = O(logn)

Agenda

- Introduction
- Linear search
- Binary search
- Summary

CANTHO UNIVERSITY

Summary

- Linear search
  - Can be done in either on array or linked list
  - Takes O(n) time
- Binary search
  - Can only be done on array
  - Takes O(logn) time

CANTHO UNIVERSITY

Q&A