Today: Divide and Conquer!

Algorithm Analysis

- Big-O Notation
- Model of computation

Searching

Peak Finding

- 1-dimension
- 2-dimensions

Binary Search

Sorted array: A[1..n]

2	4	4	5	6	7	8	9	11	17	23	28

Search for k in array A.

Binary Search

Sorted array: A[1..n]

2	4	4	5	6	7	8	9	11	17	23	28

Search for 17 in array A.

Binary Search

Sorted array: A[1..n]

2	4	4	5	6	7	8	9	11	17	23	28

Search for 17 in array A.

- Find middle element: 7

Binary Search

Sorted array: A[1..n]

2	4	4	5	6	7	8	9	11	17	23	28

Search for 17 in array A.

- Find middle element: 7
- Compare 17 to middle element: 17 > 7

Binary Search

Sorted array: A[1..n]

2	4	4	5	6	7	8	9	11	17	23	28

Search for 17 in array A.

- Find middle element: 7
- Compare 17 to middle element: 17 > 7

Binary Search

Sorted array: A[1..n]

2	4	4	5	6	7	8	9	11	17	23	28

Search for 17 in array A.

- Find middle element: 7
- Compare 17 to middle element: 17 > 7
- Recurse on left half

Binary Search

Sorted array: A[1..n]

2	4	4	5	6	7	8	9	11	17	23	28

Search for 17 in array A.

- Find middle element
- Compare 17 to middle element
- Recurse

Binary Search

Sorted array: A[1..n]

2	4	4	5	6	7	8	9	11	17	23	28

Search for 17 in array A.

- Find middle element
- Compare 17 to middle element
- Recurse

Binary Search

Sorted array: A[1..n]

2	4	4	5	6	7	8	9	11	17	23	28

Search for 17 in array A.

- Find middle element
- Compare 17 to middle element
- Recurse

Binary Search

Sorted array: A[1..n]

2	4	4	5	6	7	8	9	11	17	23	28

Search for 17 in array A.

- Find middle element
- Compare 17 to middle element
- Recurse

Binary Search

Sorted array: A[1..n]

2	4	4	5	6	7	8	9	11	17	23	28

Search for 17 in array A.

- Find middle element
- Compare 17 to middle element
- Recurse

Binary Search

Sorted array: A[1..n]

2	4	4	5	6	7	8	9	11	17	23	28

```
Search(A, key, n)
    begin = 0
    end = n-1
    while begin != end do:
        if  key < A[(begin+end)/2] then
            end = (begin+end)/2 – 1
        else begin = (begin+end)/2
    return A[begin]
```

Binary Search

Sorted array: A[1..n]

2	4	4	5	6	7	8	9	11	17	23	28

Search(A, key, n)

    begin = 0

    end = n-1

    while begin != end **do**:

        if key < A[(begin+end)/2] **then**

            end = (begin+end)/2 – 1

        else begin = (begin+end)/2

    return A[begin]

Does not terminate!

Round down?

A[begin] == key?

Binary Search

Specification:

- Finds element if it is in the array.
- Returns “NO” if it is not in the array

Binary Search

Sorted array: A[1..n]

2	4	4	5	6	7	8	9	11	17	23	28

```
Search(A, key, n)
    begin = 0
    end = n
    while begin < end – 1   do:
        if  key < A[(begin+end)/2] then
            end = (begin+end)/2
        else begin = (begin+end)/2
    return A[begin]
```

Precondition and Postcondition

Precondition:

– Fact that is true when the loop/method begins.

Postcondition:

– Fact that is true when the loop/method ends.

Loop Invariants

Invariant:

- relationship between variables that is always true.

Loop Invariant:

- relationship between variables that is true at the beginning (or end) of each iteration of a loop.

Binary Search

Sorted array: A[1..n]

2	4	4	5	6	7	8	9	11	17	23	28

```
Search(A, key, n)
    begin = 0
    end = n
    while begin < end – 1   do:
        if  key < A[(begin+end)/2] then
            end = (begin+end)/2
        else begin = (begin+end)/2
    return A[begin]
```

Binary Search

Functionality:

- If element is in the array, return it.

Preconditions:

- Array is of size n
- Array is sorted

Postcondition:

- A[begin] = key

Binary Search

Sorted array: A[1..n]

2	4	4	5	6	7	8	9	11	17	23	28

```
Search(A, key, n)
    begin = 0
    end = n
    while begin < end – 1   do:
        if  key < A[(begin+end)/2] then
                end = (begin+end)/2
        else begin = (begin+end)/2
    return A[begin]
```

Binary Search

Loop invariant:

- $A[begin] \le key \le A[end]$

Interpretation:

- The key is in the range of the array

Error checking:

if ((A[begin] > key) or (A[end] < key))

System.out.println("error");

Binary Search

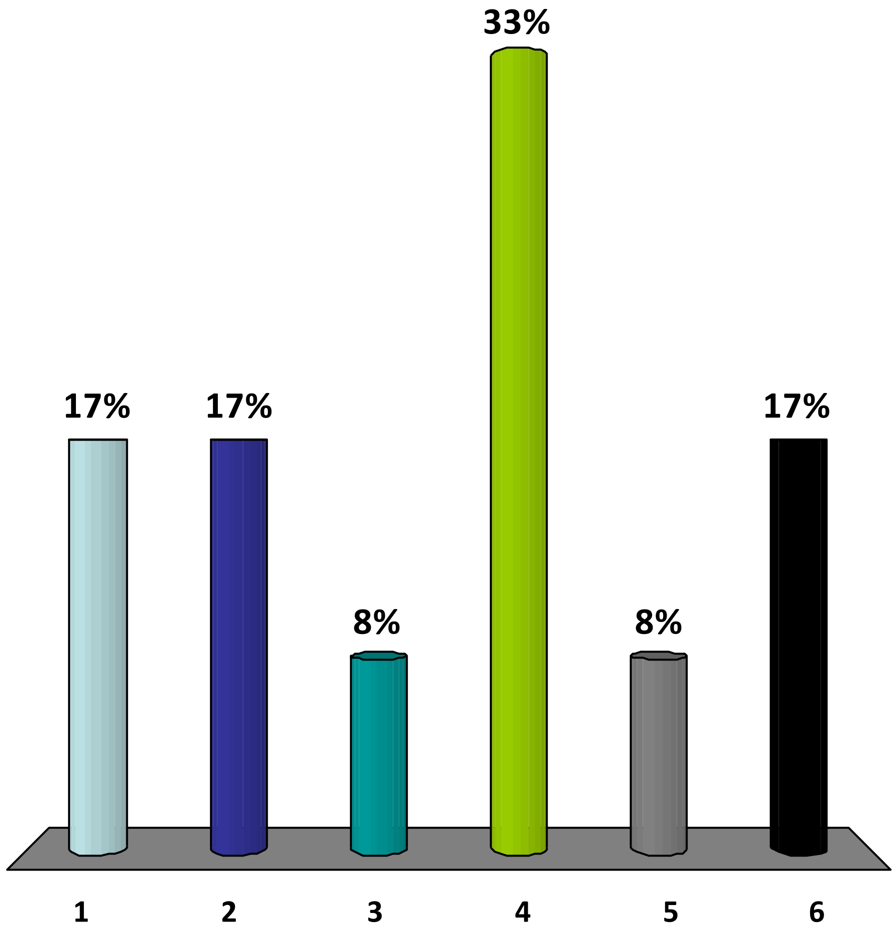
Sorted array: A[1..n]

2	4	4	5	6	7	8	9	11	17	23	28

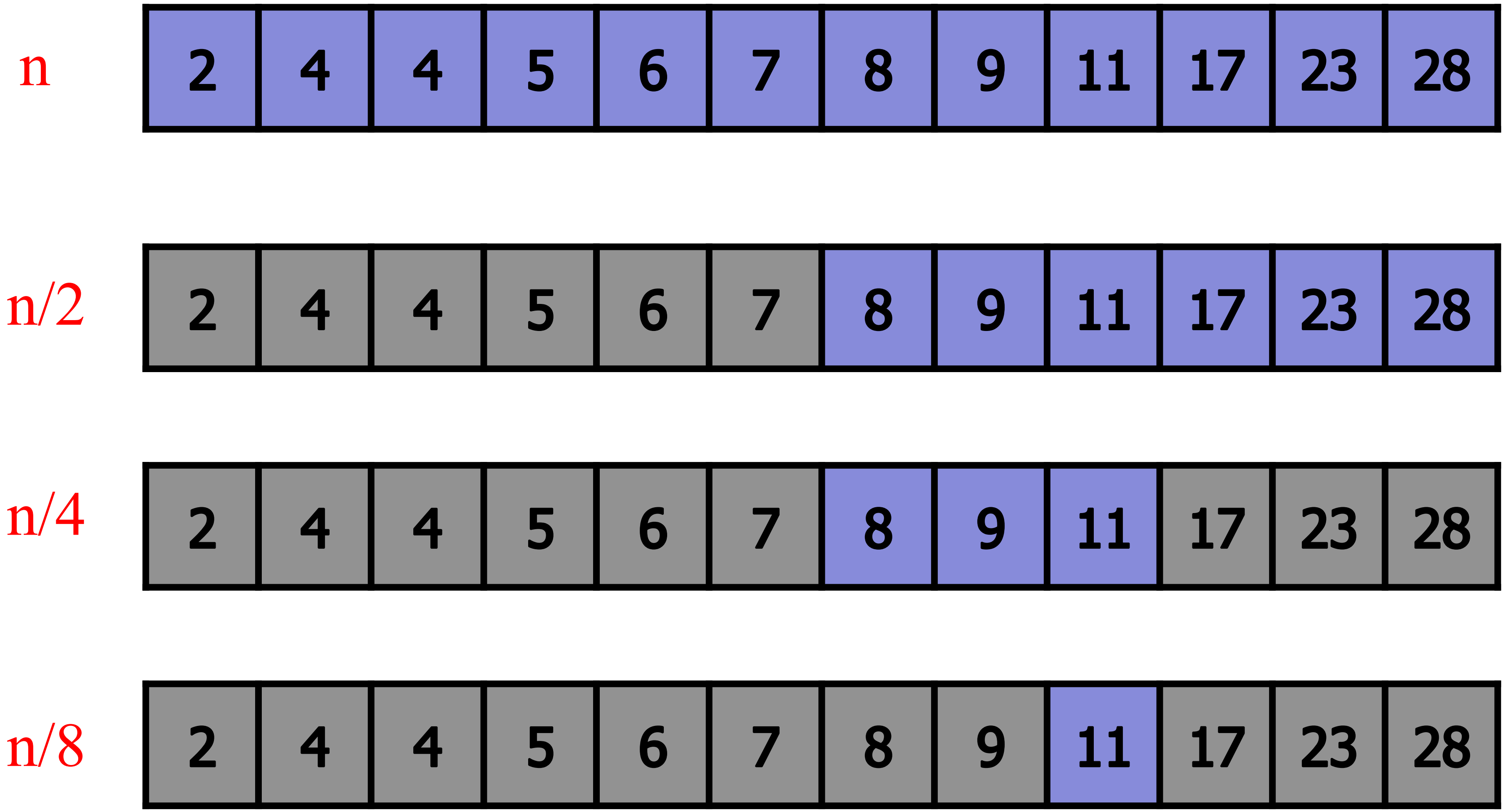
```
Search(A, key, n)
    begin = 0
    end = n
    while begin < end – 1   do:
        if  key < A[(begin+end)/2] then
            end = (begin+end)/2
        else begin = (begin+end)/2
    return A[begin]
```

What is the running time of Binary Search?

1. O(1)
2. O(log n)
3. O(n)
4. O(n log n)
5. O(n^2)
6. I'm confused…



Binary Search



Binary Search

Sorted array: A[1..n]

2	4	4	5	6	7	8	9	11	17	23	28

Iteration 1: (end – begin) = n

Iteration 2: (end – begin) = n/2

Iteration 3: (end – begin) = n/4

…

Iteration k: (end – begin) = 1 = $n/2^k$

$n/2^k = 1 \quad \rightarrow \quad k = \log(n)$