Chapter 1 Intro to Algorithms

Fei Tan

Chaifetz School of Business
Saint Louis University

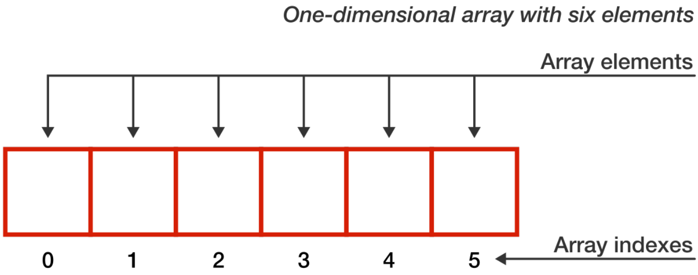
Computer Science Fundamentals
(Source: brilliant.org)

December 1, 2018

The Road Ahead...

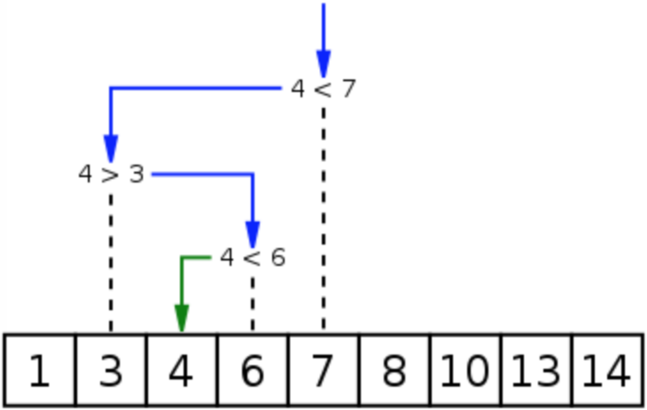
- The ability to manipulate and interpret data is essential for any programmer
- What we'll accomplish
  - discover meaning of binary search
  - evaluate performance of algorithms

Data Structure: Arrays



- Array (fixed size) holds an ordered collection of items accessible by integer index
- Data structures: array (direct access) vs. list (sequential access) due to how they are stored in memory

Algorithm: Searching



- Visualization of binary search with target $= 4$; with N elements, need at most $\lceil \log_2 N \rceil$ comparisons
- Algorithms: binary search (sorted array) vs. linear search (unsorted array)

Python Code

Iterative implementation

```python
def binarySearch(mylist, target):
    first = 0
    last = len(mylist)-1

    while first<=last:
        #note: use of // indicates floor division.
            Ex. 5//2 = 2
        midpoint = (first + last)//2
        if mylist[midpoint] == target:
            return midpoint
        else:
            if target < mylist[midpoint]:
                last = midpoint-1
            else:
                first = midpoint+1

    return None
```

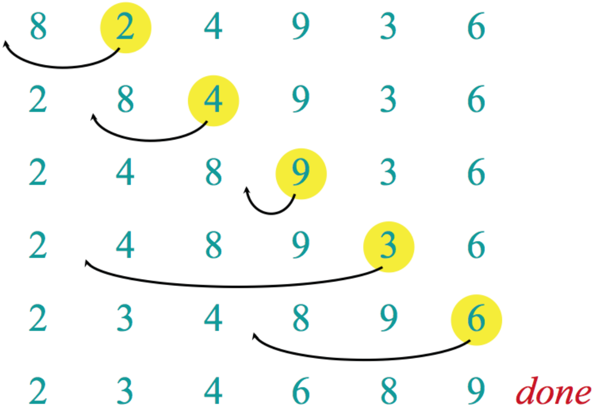
Python Code (Cont'd)

Recursive implementation

```python
def binarySearch(mylist, elem, start=0, end=None):
    if end is None:
        end = len(mylist) - 1
    if start > end:
        return 'Value not found in list'

    mid = (start + end) // 2
    if elem == mylist[mid]:
        return mid
    if elem < mylist[mid]:
        return binarySearch(mylist, elem, start,
            mid-1)
    # elem > mylist[mid]
    return binarySearch(mylist, elem, mid+1, end)
```

Algorithm: Sorting



- Visualization of insertion sort; for each $A[i]$, swap if $A[i] > A[i+1]$ until $A[i] \leq A[i+1]$
- In-place algorithm: no need to create new array to store sorted values

Python Code

```python
def insertSort(array):
    for slot in range(1, len(array)):
        value = array[slot]
        test = slot - 1
        while test > -1 and array[test] > value:
            array[test + 1] = array[test]
            test = test - 1
        array[test + 1] = value
    return array
```

Time Complexity

Big O notation

$$f = O(g) : \exists\ C \text{ s.t. } |f(x)| \leq C \cdot |g(x)| \text{ for large } x$$

- Evaluating (worst-case) algorithmic performance
  - compare *asymptotic* behavior of algorithms
  - how performance scales as function of input size
- Time complexity
  - binary search: $O(\log N)$
    - in CS, $\log$ denotes base-2 logarithm
    - $\log_2 N = O(\log N)$ since $\log_a b = \frac{\log_c b}{\log_c a}$
  - linear search: $O(N)$
  - insertion sort: $O(N^2)$