Wednesday Nov. 28

Lecture 23

Is an array sorted?

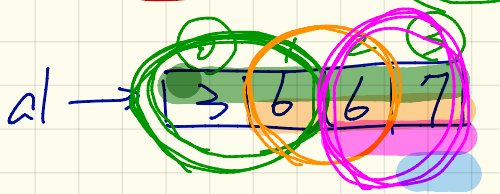
```
int[] a1 = {};
print( isSorted(a1) )
```

```
boolean isSorted(int[] a) {
  return isSortedHelper(a, 0, a.length - 1);
}

boolean isSortedHelper(int[] a, int from, int to) {
  if (from > to) { /* base case 1: empty range */
    return true;
  }
  else if (from == to) { /* base case 2: range of one element */
    return true;
  }
  else {
    return a[from] <= a[from + 1]
      && isSortedHelper(a, from + 1, to);
  }
}
```

Tracing isSorted

Say a1 = {3,6,6,7}, a2 = {3,6,5,7}



isSorted(a1)

isSH(a,0,3)

a[0]<=a[1]

3 ≤ 6

T

isSH(a,1,3)

a[1]<=a[2]

6 ≤ 6

T

isSH(a,2,3)

a[2]<=a[3]

T

6

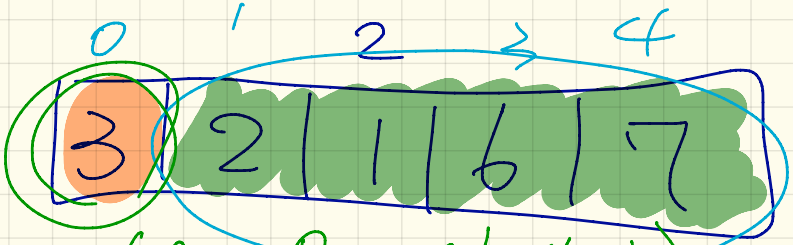
7

isSH(a,3,3)

T

base case

1



min(a, 0, a.length-1)

0 4

1 4

if(a is empty) { no min }

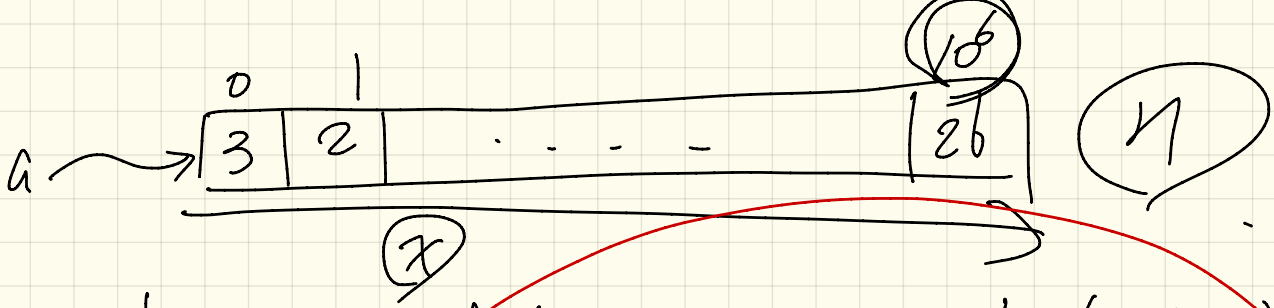
else if(a is size 1) { return a[0] }

else {

int minOfRest = min(a, from+1, to);

if(a[0] < minOfRest) { return a[0]; }

} -> else { return minOfRest; }

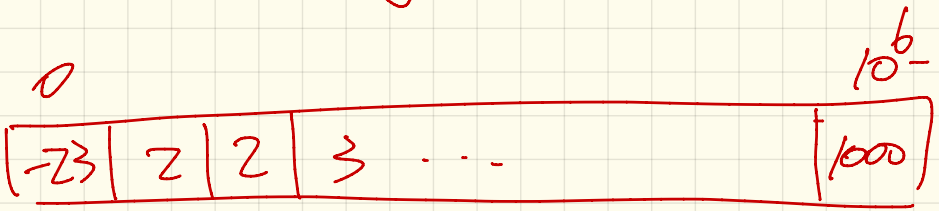


search(23)

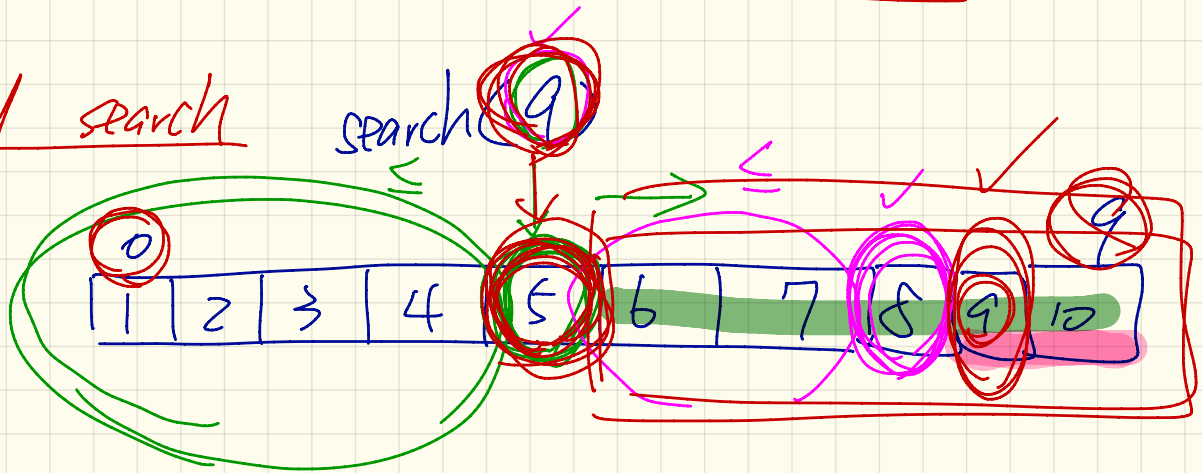
```
for(int i=0; i<a.length; i++){
    if(a[i] == 23){ return true}
}
return false;
```

$O(N)$ 10^6

Assume input array is sorted



Binary search

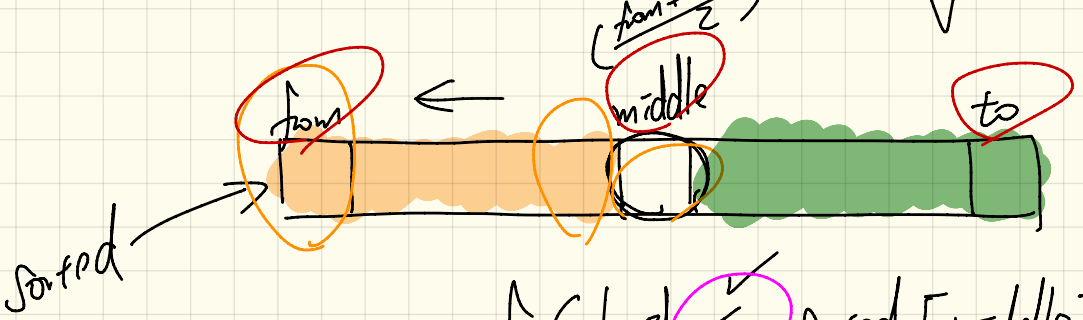


Binary Search

```
boolean binarySearch(int[] sorted, int key) {
  return binarySearchHelper(sorted, 0, sorted.length - 1, key);
}
boolean binarySearchHelper(int[] sorted, int from, int to, int key
  if (from > to) { /* base case 1: empty range */
    return false; }
  else if (from == to) { /* base case 2: range of one element */
    return sorted[from] == key; }
  else {
    int middle = (from + to) / 2;
    int middleValue = sorted[middle];
    if(key < middleValue) {
      return binarySearchHelper(sorted, from, middle - 1, key);
    }
    else if (key > middleValue) {
      return binarySearchHelper(sorted, middle + 1, to, key);
    }
    else { return true; }
  }
}
```

in ascending order

```
binSH( sorted , from , to , key )
```

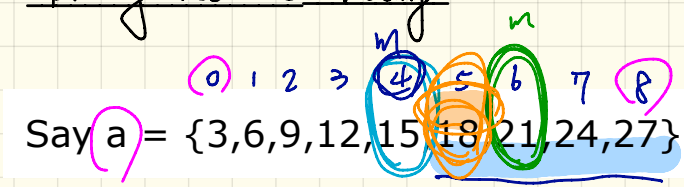



```
if( key < sorted[middle] ) {
    binSH( sorted , from , middle-1 , key );
}
else if( key > sorted[middle] ) {
    binSH( sorted , middle+1 , to , key );
}
```

Exercise

Modify the BinS. so that the input array is sorted in descending order.

Binary Search: Tracing



search(a,18)

search(a,0,8,18)

middle = 4
middV = 15
18 > 15

search(a,5,8,18)

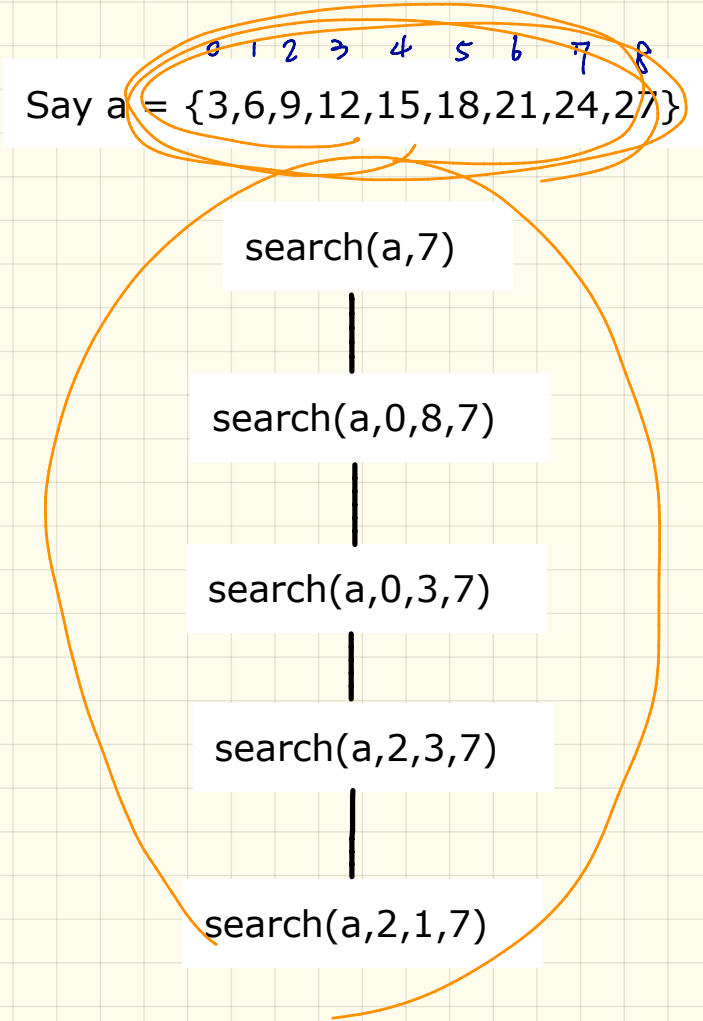
mid + 1

middle = 5+8 = 6
middleV = 21
18 < 21

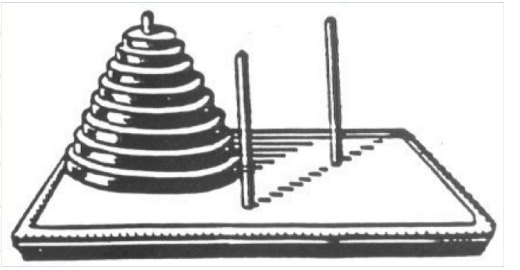
search(a,5,5,18)

top mid - 1

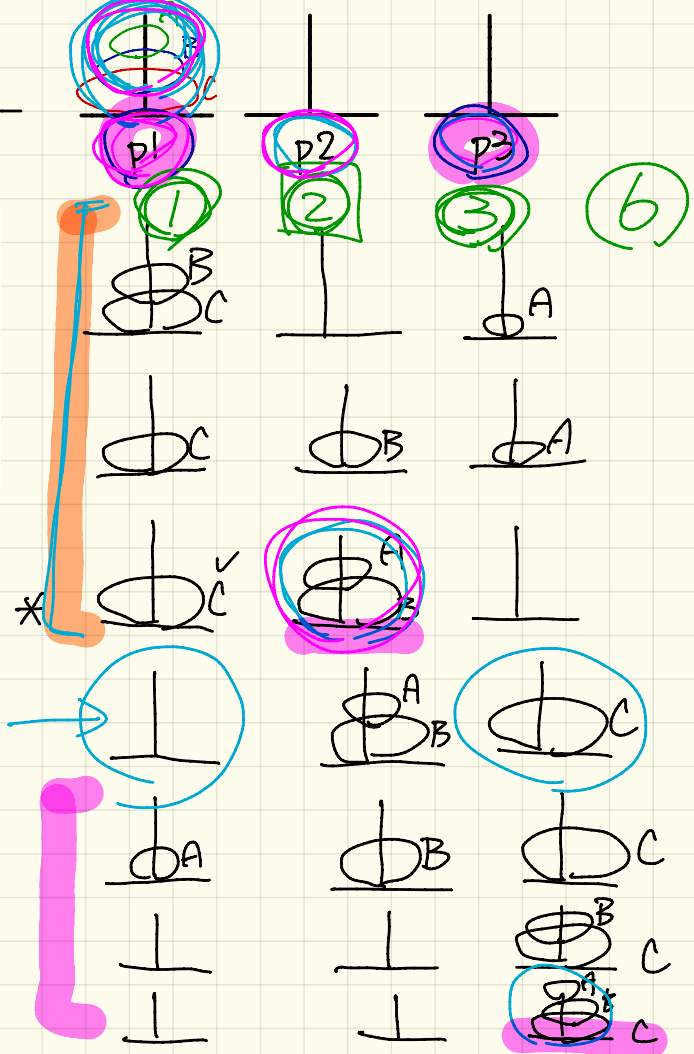
sorted[5] == 18 ✓



Tower of Hanoi : Strategy



Consider 3 disks A < B < C



Tower of Hanoi : Java

A<B<C

move from p1 to p3

```
void towerOfHanoi(String[] disks) {
  tohHelper(disks, 0, disks.length - 1, 1, 3);
}
void tohHelper(String[] disks, int from, int to, int p1, int p2) {
  if(from > to) { }
  else if(from == to) {
    print("move " + disks[to] + " from " + p1 + " to " + p2);
  }
  else {
    int intermediate = 6 - p1 - p2;
    tohHelper(disks, from, to - 1, p1, intermediate);
    print("move " + disks[to] + " from " + p1 + " to " + p2);
    tohHelper(disks, from, to - 1, intermediate, p2);
  }
}
```

move from p1 to intermediate (p2)

Say disks = {A,B,C}.
Consider towerOfHoni(disks) which calls:
tohHelper(disks, 0, disks.length - 1, 1, 3)