CS206

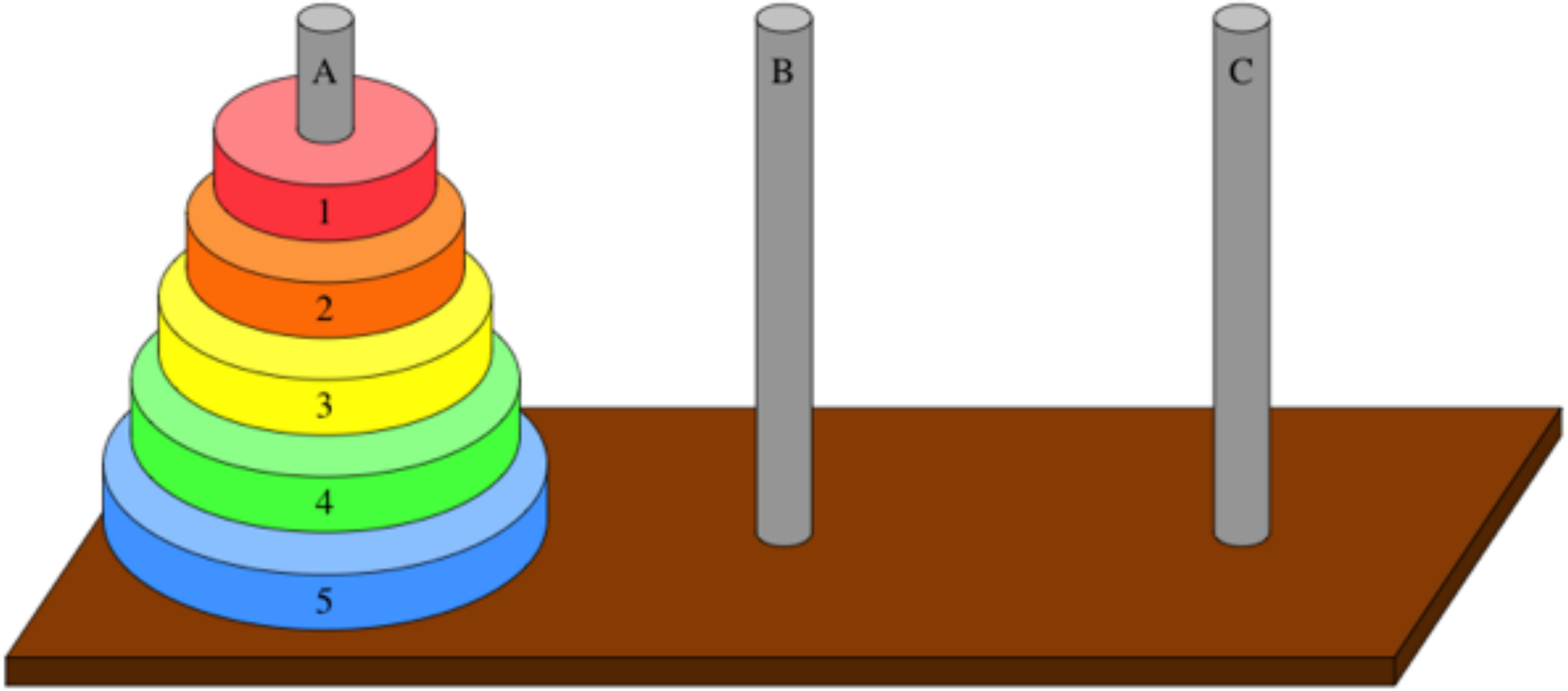
Recursion, Binary Search

recursion practice (from Tuesday)

```
/**
  * A function to add all the values in the array
  * @param array of integers
  * @return the sum of the numbers in the array  */
 public int addArray(int[] array) {
     return addArray(array, 0);
 }


/**
  * A private recursive function for adding the values in an array
  * @param array of interest to be added
  * @param the location in the array to be added next
  * @return the sum of the numbers in the array   */
 private int addArray(int[] array, int loc) {
     if (loc>=array.length)
         return 0;
     return array[loc] + addArray(array, loc+1);
 }
```

Towers of Hanoi



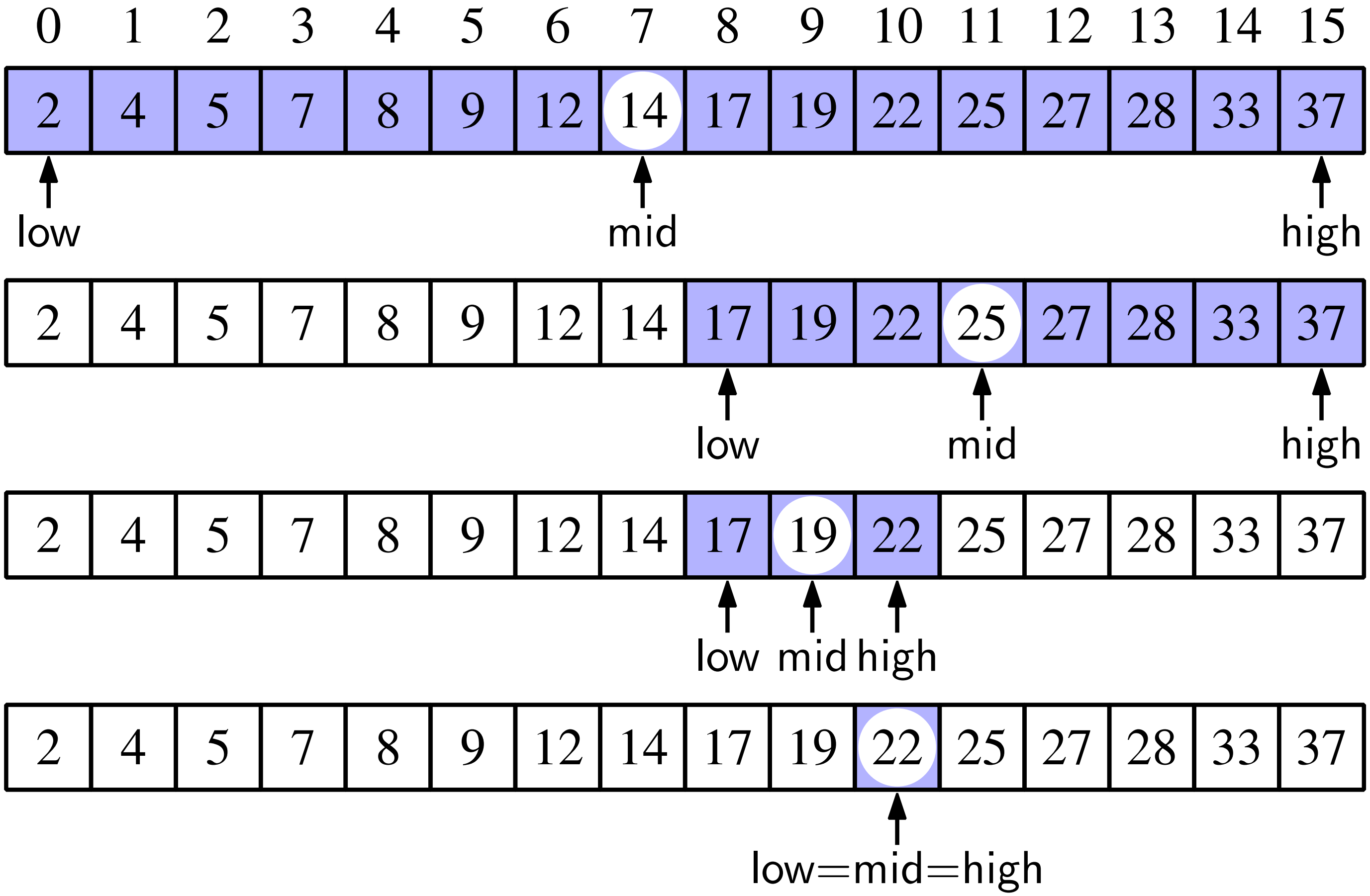
Binary Search

- Search for an integer (22) in an ordered list

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	4	5	7	8	9	12	14	17	19	22	25	27	28	33	37

- $mid = \left\lfloor \frac{low + high}{2} \right\rfloor = \left\lfloor \frac{0 + 15}{2} \right\rfloor = 7$
    - `target==data[mid]`, found
    - `target>data[mid]`, recur on second half
    - `target<data[mid]`, recur on first half

target = 22



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	4	5	7	8	9	12	14	17	19	22	25	27	28	33	37
low							mid								high
2	4	5	7	8	9	12	14	17	19	22	25	27	28	33	37
								low			mid				high
2	4	5	7	8	9	12	14	17	19	22	25	27	28	33	37
								low	mid	high					
2	4	5	7	8	9	12	14	17	19	22	25	27	28	33	37
										low=mid=high					

Binary Search Code

```
/**
* The public facing call to array search
* The array to be searched is a private instance variable
* @param target the value being searched for
* @return true if the value is in known, false otherwise
*/
public boolean contains(int target) {
    if (data==null)
        return 0;
    return iSearch(target, 0, data.length-1, 0);
}
```

Suppose change instance variable data to ArrayList?

Binary Search Code

```java
   /**
     * Binary search, recursively on sorted internal array of ints
     * @param target the item to be found
     * @param lo the bottom of the range being searched
     * @param hi the top of the range being searched
     * @param steps the number of steps the search has taken
     * @return true if the target was found
     */
private boolean iSearch(int target, int lo, int hi, int steps) {
    if (lo>hi) return false;
    int mid = (lo+hi)/2;
    System.out.println(target + " " + data[mid] + " " + lo + " " + hi
+ " " + steps);
    if (data[mid]==target) return true;
    if (data[mid]<target)
        return iSearch(target, mid+1, hi, steps+1);
    else
        return iSearch(target, lo, mid-1, steps+1);
    }
```

Binary Search Analysis

- Each recursive call divides the array in half
- If the array is of size n, it divides (and searches) at most lgn times before the current half is of size 1
- $O(lgn)$

Reimplement Binary search with iteration

What parameters does the iterative method need?
Does a separate private method even make sense?

Backtracking with Recursion

- Previous examples all progressed linearly to success/failure
- So consider doing binary like search on an unsorted array
  - Need to backtrack and try other directions on failure.
  - Backtracking is when recursion really shines

Backtracker

```
/** Binary-like search, but will work on sorted or unsorted lists
  * because it can do backtracking.
  */
private boolean iSearch(int target, int lo, int hi, int
depth)
    {
   if (lo>hi) { return false; }
   int mid = (lo+hi)/2;
   System.out.println(" " + target + " " + data[mid] + "
+ lo + " " + hi + " " + depth);
   if (data[mid]==target) return true;
   if (iSearch(target, mid+1, hi, depth+1))
      return true;
   return iSearch(target, lo, mid-1, depth+1);
}
```