Using Induction to Show Correctness

- use induction with a *loop invariant* to show correctness of a loop
- use induction to show correctness of recursion

Insertion Sort

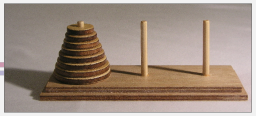
```
public static void sort(int[] arr) {
  for ( int i = 1 ; i < arr.length ; i++) {
    int elt = arr[i]; // current element to put in place

    // shift - move elements of arr[0..i-1] that are
    // greater than elt one spot to the right
    int shift = i - 1;
    for ( ; shift >= 0 && arr[shift] > elt ;
          shift = shift - 1) {
      arr[shift + 1] = arr[shift];
    }
    // put element in place
    arr[shift + 1] = elt;
  }
}
```

- define a *loop invariant*
  - a boolean statement about correctness of the solution so far that, if true when the loop exits, establishes correctness of the resulting answer

Insertion Sort

```
public static void sort(int[] arr) {
  for ( int i = 1 ; i < arr.length ; i++) {
    // arr[0..i-1] (inclusive) is sorted in increasing order
    int elt = arr[i]; // current element to put in place

    // shift - move elements of arr[0..i-1] that are
    // greater than elt one spot to the right
    int shift = i - 1;
    for ( ; shift >= 0 && arr[shift] > elt ;
          shift = shift - 1) {
      arr[shift + 1] = arr[shift];
    }
    // put element in place
    arr[shift + 1] = elt;
  }
  // arr[0..arr.length-1] (inclusive) is sorted in
  //  increasing order
}
```

Towers of Hanoi

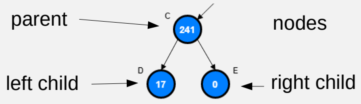


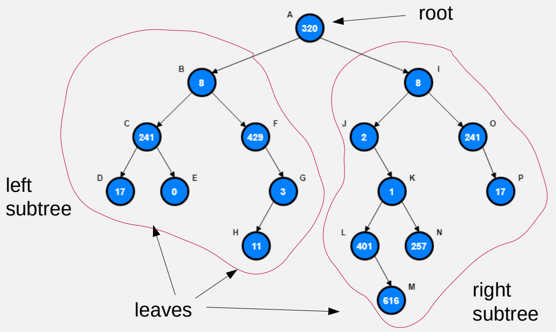
```
public static void hanoi(int n,
                         int from, int to, int spare) {
  // if there is only one disk, simply move it
  // otherwise
  // move the top n-1 disks out of the way (onto spare) so that
  // the nth disk can be moved
  // move the nth disk
  // move the n-1 disks from the spare to the final goal

  if (n == 1) {
    System.out.println("move disk from " + from + " to " + to);
  } else {
    hanoi(n - 1, from, spare, to);
    System.out.println("move disk from " + from + " to " + to);
    hanoi(n - 1, spare, to, from);
  }
}
```

- let P(n) be the statement that this code gives a valid solution for the towers of hanoi with n disks, $n \geq 1$

https://commons.wikimedia.org/wiki/File:Tower_of_Hanoi.jpeg

Binary Trees





https://emre.me/data-structures/binary-tree/

Mergesort

```
public static void sort(int[] arr) {
  int n = arr.length;

  if (n <= 1) { return; }

  int mid = n / 2;

  int[] left = new int[mid];
  int[] right = new int[n - mid];

  for (int i = 0; i < mid; i++) { left[i] = arr[i]; }
  for (int i = mid; i < n; i++) { right[i - mid] = arr[i]; }

  sort(left);
  sort(right);
  merge(arr, left, right);
}
```

Mergesort

```
public static void merge(int[] arr, int[] left, int[] right) {
  int i = 0, j = 0, k = 0;
  for ( ; i < left.length && j < right.length ; k++ ) {
    // arr[0..k-1] (inclusive) is sorted
    if (left[i] <= right[j]) {
      arr[k] = left[i];
      i++;
    } else {
      arr[k] = right[j];
      j++;
    }
  }
  for ( ; i < left.length ; i++, k++ ) {
    // arr[0..k-1] (inclusive) is sorted
    arr[k] = left[i];
  }
  for ( ; j < right.length ; j++, k++ ) {
    // arr[0..k-1] (inclusive) is sorted
    arr[k] = right[j];
  }
}
```