Section 9 Overview

Agenda

- Mini Assignment
- Review on Spanning Trees
  - Spanning Trees vs. Minimum Spanning Trees
  - Prim-Jarnik’s Algorithm
  - Kruskal’s Algorithm
- Code Review
  - Peer review code style using radix sort code
- Functional Programming
- Hand Back Exams
- Interview Questions

Optional Problems

Detect if there is a cycle in a singly linked list

```python
def isCycle(Node node):
    set = HashSet()
    while node != null:
        if set.contains(node):
            return true
        set.insert(node):
        node = node.next
    return false
```

```python
def isCycle(Node node):
    Node fast = node
    while fast != null and fast.next() !=null:
        node = node.next()
        fast = fast.next().next()
    if fast == node:
        return true
    return false
```

Reverse a string in Java without stringBuilder

In place solution:

```java
public String reverse(String s) {
    char[] str = s.toCharArray();
    int mid = str.length / 2;
```
// If we went from i=0:str.length;, we’d end up un-reversing everything
for (int i = 0; i < mid; i++) {
    char tmp = str[i];
    str[i] = str[str.length - i];
    str[str.length - i] = tmp;
}
return new String(str);

Reverse a singly linked list
http://www.geeksforgeeks.org/write-a-function-to-reverse-the-nodes-of-a-linked-list/

public void reverse(Node head) {
    prev = None
    current = head
    while(current is not None):
        next = current.next
        current.next = prev
        prev = current
        current = next
    head = prev
}

Reverse a doubly linked list
http://www.geeksforgeeks.org/reverse-a-doubly-linked-list/

public void reverse(Node head) {
    Node tmp = null;
    Node curr = head;

    while (curr != null) {
        tmp = curr.prev
        curr.prev = curr.next
        curr.next = tmp
        curr = curr.prev
    }

    // Edge cases like empty list and list with one node
    if (tmp != null) {
        head = tmp.prev
    }
Are two trees equal?

```java
public boolean treeEqual(TreeNode a, TreeNode b) {
    if(a.value != b.value)
        return false
    if(a.hasLeft() && b.hasLeft())
        if(!treeEqual(a.left(), b.left()))
            return false
    else
        if(a.hasLeft() || b.hasLeft())
            return false
    if(a.hasRight() && b.hasRight())
        if(!treeEqual(a.right(), b.right()))
            return false
    else
        if(a.hasRight() || b.hasRight())
            return false

    return true
}
```

Is a graph a 2-colorable?

A graph is k-colorable if every node can be assigned one of k colors such that no node has a neighbor that is the same color as itself.

```python
//all nodes in g are initially uncolored
//assume g is connected
def is2Colorable(Graph g):
    pick any v in g
    Q = [ ] //Queue
    give v color A
    Q.enqueue(v)
    while Q not empty:
        vertex cur = Q.dequeue;
        for all edges e of cur:
            w = node on opposite side of e
            if cur and w are the same color
                return false
            if w is uncolored
                give w the opposite color of cur
                Q.enqueue(w)

    return true
```
Write an algorithm that takes in a list of numbers and a goal number, and returns whether or not a pair in the list sums to the goal number.

// Inputs: list $L$ of numbers, and integer $sum$
// Output: whether or not there exists a pair in $L$ that sums to $sum$

def findNumSums(L, sum):
    Hashset $hash$ is an empty hashset of integers
    for each number $n$ in $L$:
        if $hash$ contains $sum - n$:
            return true
        insert $n$ into $hash$
    return false