Section #5 Overview

Agenda

- Induction - proving perfect binary tree number of leaves and height
- Graphs - computing whether there exists a path between vertices. Know different implementations of graphs
- Cycles - check if singly linked list contains cycle
- Generics - can use multiple types without casting. Important for Heap
- Selection - different selection algorithms

Pseudocode:

Is Cycle
Given a singly linked list, return true if there is a cycle (return false otherwise).

```java
public boolean isCycle (Node head){
    if (head == null) {
        return false;
    }
    Node tortoise = head;
    Node hare = head.next;
    while (hare != null){
        if (tortoise == hare) {
            return true;
        }
        if (hare.next == null) {
            return false;
        }
        hare = hare.next.next;
        tortoise= tortoise.next;
    }
    return false;
}
```

Optional Problems:

Runner Reinforcement:
Given a linked list of unknown length, find the midpoint of the list without counting how many nodes there are the list then traversing to the middle--that would be the naive solution and too boring.

Hint them to use a runner if they’re struggling.
Can assume that if the length of the list is even, then either of the “middle” nodes is OK to return.

```python
def findMiddle(node head):
    node runner = head;
    node current = head;

    while runner != null and runner.next != null:
        runner = runner.next.next
        current = current.next

    return current
```

**Is a graph 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
isTwoColorable(graph g):
    let v be a random vertex in g
    Q = [ ] //Queue
    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
```
Arranging Queens:
Write an algorithm to print all ways of arranging eight queens on a chess board so that none of them share the same row, column or diagonal.

```java
int columnForRow[] = new int [8];

boolean check(int row) {
    for (int i = 0; i < row; i++) {
        int diff = Math.abs(columnForRow[i] - columnForRow[row]);
        if (diff == 0 || diff == row - i) return false;
    }
    return true;
}

void placeQueen(int row) {
    if (row == 8) {
        printBoard();
        return;
    }
    for (int i = 0; i < 8; i++) {
        columnForRow[row] = i;
        if (check(row)) {
            placeQueen(row + 1);
        }
    }
}
```