CS206

Final Comments and Review

Dijsktra's shortest path

```
private class Node<H> {
    // Node content
    public H payload;
    // hold the list
    public ArrayList<Edge<G>> edges;
    // best known path to the node
    public ArrayList<Node<H>> path;
    // true if there is a path
    public boolean visited;
    // cost to traverse the path
    double cost = 0;
```

```
public void shortestWeightedPathFrom(G fr) {
    cleanPaths();
    PriorityQHeap<Double, Node<G>> pq = new PriorityQHe
    {
        Node<G> ng = nodeHash.get(fr);
        ng.visited = true;
        ng.path = new ArrayList<>();
        ng.cost = 0;
        pq.offer(0d, ng);
    }
    while (!pq.isEmpty()) {
        Node<G> curr = pq.poll();
        for (Edge<G> linked : curr.edges) {
            Node<G> ln = linked.too;
            if ((curr.cost+linked.weight) < ln.cost) {
                ln.doVisit(curr, curr.cost+linked.weigh
                pq.offer(ln.cost, ln);
            }
        }
    }
}
```

Finding Groups

- Suppose undirected links
- Question: Identify groups
  - A group is all the nodes in a graph that can be reached from each other

```
ArrayList<ArrayList<T>> allGroups() {
    change visited from boolean to integer
    mark all nodes as unvisited (==0)
    set group = 1
    while exists unvisited nodes
        a <- an unvisited node
        run dijkstra shortest path from node a
            marking nodes as visited in group
        group <- group + 1
    Collect and return groups
```

Data Structures

- Array
- ArrayList
  - it grows and shrinks
- Maps / Hashtables
  - going beyond numeric indexes
- Stacks and Queues
- Linked Lists
- Trees
- ~~Graphs~~

Programming techniques and concepts

- Object oriented programming
  - inheritance, generics, …
- Searching
- Sorting
- Recursion
- Analysis

Course Goals

1. Become a better computer scientist
2. Learn about common data structures
    1. Implementation
    2. How and when to use each
3. Understand Object Oriented program design and its implementation in Java
4. Develop an understanding of UNIX
5. Become a better Java programmer

Final

- Strong emphasis on final 1/3 of course
- Questions may involve material from first 2/3
- You will have 3 hours + 20 minutes from the time you download to the time you submit.
- Open book, open notes, open computer, open web.
- Instructions are largely unchanged from midterms.
- No discussion with anyone.