Judah Goldring
Wealth Transfer Write-Up
Professor Leff
Algorithms

I realized that the wealth transfer problem is essentially a directed graph with weights added to every edge (the percentage and squared Boolean as weights). I also realized it wasn't necessary to create an actual graph, all I needed to do was replicate, so I decided to use maps instead (maps solve everyone's problem). Every time someone intended to transfer wealth, I added the id of the person they were transferring to as the key and the id of the person transferring from as the value. I also created two other maps for this method, one that calculated the multiplier required to get from whatever stored at the child to get to the parent (multiplier calculation was just 100/percentage of wealth given away) the other map created for this method was a Boolean map that was true if the id being transferred to was going to be a squared transfer and false otherwise. Also, for required wealth I created a wealth map that mapped the id of the persons required wealth to the wealth number every time a required wealth was called. Now all I had to do in the solve it method was loop the through the key set of the required wealth map and iteratively climb up the map until the root was reached. At each step up the value stored at the current key of the required wealth map would be multiplied by the value of the multiplier map of that id and if the squared map at that id was true the value would be equal to its square root. I did this iterative approach toward the root for every key of the wealth key set and the maximum value was the minimum amount of money required such that everyone got their money.

Pseudo code

```
intendToTransferWealth(from, to, percentage, squared)

    parent.put(to, from)
multiplier.put(to, 100/percentage)
    squared.put(to, squared)

setRequiredWealth(id, wealth)
    wealth.put(id, wealth)

solveIt()
    max = 0;
 //going through every key that has a required wealth  for (j:
wealth.keyset())
        money = wealth.get(j)
        k = j
        //iteratively climbing out the "graph"
        while(parent.get(k) != null)
            if(squared(k))
                money = sqrt(money)
            money *= multiplier.get(k)
        k = parent.get(k)
        if(money > max)
  max = money  return max
```