KIT205 Data Structures and Algorithms

Week 9 Tutorial

In this tutorial you will be implementing the page rank algorithm for the graph data structure that you implemented in the last tutorial.

Graph Data Structure

Last week you should have implemented the following data structure and written code to allow a graph to be entered. If you haven't completed that tutorial, do that first.

```
typedef struct edge{
    int to_vertex;
    int weight;
} Edge;

typedef struct edgeNode{
    Edge edge;
    struct edgeNode *next;
} *EdgeNodePtr;

typedef struct edgeList{
    EdgeNodePtr head;
} EdgeList;

typedef struct graph{
    int V;
    EdgeList *edges;
} Graph;
```

PageRank

This week you need to implement the PageRank algorithm as described in the lectures. You should use the iterative method. We will use the un-normalised version of the algorithm, so the equation will be:

$$PR(p_i) = (1-d) + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

Since our graph only stores outgoing edges, you need to calculate the sums (the sigma bit) separately. Only after the sums have been calculated for all vertices can you calculate the new PageRanks. Use the following pseudocode:

for each iteration:

1. *Set the sums for each vertex to zero*
2. *Loop through all edges, incrementing sums as you go*
3. *Calculate the new PageRanks*

Each of these steps requires looping through all of the vertices, so it's tempting to combine them, but this will give the wrong answer. Each loop must be kept separate.

You should use the following data to enter the graph (the weights are irrelevant for the PageRank calculation and the order of vertices is not important either):

```
20
4
11,0 3,0 1,0 2,0
5
12,0 0,0 2,0 3,0 11,0
6
0,0 1,0 12,0 4,0 3,0 11,0
2
1,0 11,0
2
8,0 3,0
4
3,0 4,0 0,0 11,0
4
5,0 3,0 11,0 12,0
4
5,0 12,0 3,0 13,0
5
6,0 15,0 3,0 10,0 11,0
3
7,0 3,0 11,0
4
8,0 15,0 11,0 13,0
4
3,0 10,0 0,0 2,0
4
10,0 4,0 3,0 12,0
5
11,0 12,0 19,0 0,0 3,0
3
12,0 13,0 3,0
3
12,0 3,0 11,0
3
11,0 15,0 3,0
5
7,0 11,0 19,0 5,0 3,0
4
16,0 17,0 3,0 11,0
3
17,0 3,0 1,0
```

Initially, run your code for 10 iterations, with a value of 0.5 for d, and ask your tutor to check your results. Then try using different values for d and observing the effect d and the number of iterations.