Floyd Warshall Algorithm

Introduction

Floyd's algorithm is a shortest path algorithm that identifies the shortest path between two nodes on a graph. This document shows a simple implementation of Floyd's algorithm.

Floyd Warshall Algorithm

The solution that is documented here creates a graph matrix. The solution matrix is then updated by assuming all nodes are an intermediate node. The solution picks a node and then computes the distance between every node that includes the intermediate node as part of the path. It checks if the path is the current shortest path, if it is the graph is updated and if not the value is ignored.

Imperative Solution

The following is a distance matrix. Each list is the distance between that node and the rest of the nodes. For example graph[0] is the distance between node 0 and nodes 0,1,2 and 3. Unsurprisingly the distance between Node 0 and Node 0 is 0, whereas between Node 0 and Node 1 is 7. NO_PATH indicates that there is no direct path.

```
NO_PATH =  sys.maxsize
graph = [[0,   7,  NO_PATH, 8],
[NO_PATH,  0,  5,  NO_PATH],
[NO_PATH, NO_PATH, 0,   2],
[NO_PATH, NO_PATH, NO_PATH, 0]]
MAX_LENGTH = len(graph[0])

import sys
import itertools
```

floyd is the main function in the code on the next page. We use product to create the combinations for three loops so the code is uncollected. And use min to find a shortest path for the path combination that we have using the intermediate node.

Code

```
def floyd(distance):
    """
    A simple implementation of Floyd's algorithm
    """
    for intermediate, start_node,end_node\
    in itertools.product\
    (range(MAX_LENGTH),range(MAX_LENGTH), range(MAX_LENGTH)):

        # Assume that if start_node and end_node are the same
        # then the distance would be zero
        if start_node == end_node:
            distance[start_node][end_node] = 0
            continue

        #return all possible paths and find the minimum
        distance[start_node][end_node] = min(distance[start_node][end_node],
                        distance[start_node][intermediate] + distance[intermediate][end_node] )
    #Any value that have sys.maxsize has no path
    print (distance)
floyd(graph)
```