1.Getting started

The program aims to show the integration of the Python recursion function in Floyd's algorithm. The program has been created as part of a mid-module assessment for CSCK541 Software Development in Practice.

```
└──github.com/
    └── sgfferra/
        └── MSc-Projects/
        └── Assessment week 5_FF/
            └── Assessment week 5_FF/
                └── Program Final Version.py
```

Tool requirements and installation

For the purpose of this program, no specific tools are necessary. It has been created in Visual Studio Code for Mac and then implemented with Visual Studio. The program can be used in IDLE and other terminals that support Python language.

Development

The project has been created following Floyd's algorithm code provided below.

```
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)
```

To include the recurtion function additional lines have been included, the GIT repository contains the first two graph and the final version of the document, which reports the following code

```
def floyd_warshall(distance, n, k):
#  Implements Floyd's algorithm recursively to find     shortest paths.


for i in range(n):
 for j in range(n):
    if distance[i][k] + distance[k][j] < distance[i][j]:
        distance[i][j] = distance[i][k] + distance[k][j]

if k == n - 1:
    return distance
  else:
    return floyd_warshall(distance, n, k + 1)

#Example usage
dist = [[0, 3, float("inf"), 7],
    [float("inf", 0, 4, 1],
    [float("inf)", float("inf"), 0, 5],
    [float("inf", float("inf"), float("inf"), 0]]

result = floyd_warshall(dist, len(dist), 0)

print(result)  # Output: [[0 3 7 1], [INF 0 4 1], [INF INF 0 5], [INF IN
```

Authors

Francesco V Ferraro – students at the University of Liverpool

Additional consideration

This is the first time that I have written a code and the first time that I have used GitHub, I am very excited, and I hope the code works well.

Thank you 😊 Francesco.