EE2T21 Data Communications Networking - 2024
Bonus Assignment # 2

Names: Sjoerd Terlouw Student ID: 5852455

1 Abstract

Implementation of the Dijkstra algorithm in Python and performance analysis using the ER random graph.

2 Example

The following plot gives the shortest path (indicated in red) of a certain graph G using the Dijkstra algorithm. The plot itself is done using the NetworkX python library. The inputs are the edges between the nodes with their weights and the starting and ending node. In the case of figure 1 the starting node is 0 and the ending node is 1. The exact format of the input can be found in the code, found in section 4.

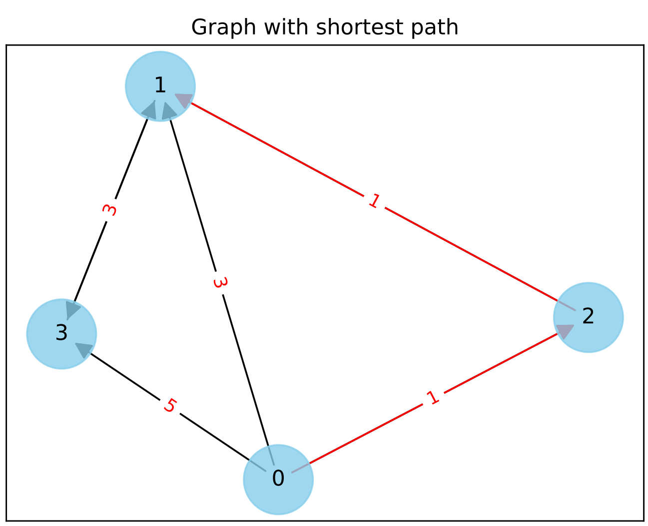

Figure 1: Visualization of the shortest path found by the Dijkstra algorithm - output of Dijkstra.run_example()

3 Performance analysis

4 Code

The code is also provided seperately and is on github.

```python
# Dijkstra.py

import networkx as nx
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

def initialize(G, A):
    """
    Find all vertices and initialize distances to infinity

    Also checks if vertices are correct and fixes them if possible
    """
    # Find all vertices
    vertices = np.unique(G['start_node'] + G['end_node'])


    # Check if nodes have correct values
    if len(vertices) == np.max(vertices)+1:
        if not np.array_equal(np.arange(np.max(vertices)+1),
         ↪ vertices):
            raise ValueError("Vertices are incorrect. Not all
             ↪ vertices are connected with edges or vertices have
             ↪ wrong names (names should be integers starting from
             ↪ 0!)")
    else:
        raise ValueError("Vertices are incorrect. Not all vertices
         ↪ are connected with edges or vertices have wrong names
         ↪ (names should be integers starting from 0!)")


    # Initialize distances
    d = [np.inf] * (np.max(vertices)+1)
    d[A] = 0

    pi = [[] for _ in range(np.max(vertices)+1)]

    return np.array(d), pi

def dijkstra(G, A, B):
    """
```

```
Description
-----------
Implementation of the Dijkstra algorithm
The nodes of G should be integers starting from 0

Parameters
----------
G : dictionary
    Contains the graph in the format
        G = {
            start_node = [0,1,2],
            end_node =   [2,3,4],
            weights =    [2,2,4]
            }
        Which indicates a graph with three weighted edges:
            - Edge going from node 0 to node 2 with weight 2
            - Edge going from node 1 to node 4 with weight 2
            - Edge going from node 2 to node 5 with weight 4
A :
    Starting node
B :
    Ending node

Returns
-------
shortest_path : list
    Shortest path from A to B starting with node A and ending
     ↪  with node B
"""
d, pi = initialize(G, A)
Q = np.array([True] * (np.max(np.array(G['start_node'] +
 ↪  G['end_node']))+1))

while Q.any() == True:
    u = np.argmin(np.where(Q, d, np.inf))
    Q[u] = False

    if u == B:
        return pi[u]+[u]
    else:
        indices = np.where(G["start_node"] == u)[0]
        for index in indices:
            w = G["weights"][index]
            v = G["end_node"][index]
            if d[u] + w < d[v]:
                d[v] = d[u] + w
                (pi[v]).append(u)
```

```python
def plot_graph(G, pi):
    """
    Description
    -----------
    Plot the graph using networkX

    Parameters
    -----------
    G : dictionary
        See dijkstra() for format
    pi : list
        Shortest path as obtained by dijkstra()
    """
    G_nx = nx.DiGraph()
    start_node = G["start_node"]
    end_node = G["end_node"]
    weights = G["weights"]

    # Add edges to the graph with weights
    for i in range(len(start_node)):
        G_nx.add_edge(start_node[i], end_node[i],
         ↪  weight=weights[i])

    pos = nx.spring_layout(G_nx)  # positions for all nodes

    # Nodes with labels
    nx.draw_networkx_nodes(G_nx, pos, node_color='skyblue',
     ↪  node_size=1500, alpha=0.8)
    nx.draw_networkx_labels(G_nx, pos, font_size=12,
     ↪  font_family='sans-serif')

    # Edges
    nx.draw_networkx_edges(G_nx, pos, edgelist=G_nx.edges(),
     ↪  edge_color='black', arrows=True, arrowsize=20)

    for i in range(len(pi) - 1):
        u = pi[i]
        v = pi[i + 1]
        nx.draw_networkx_edges(G_nx, pos, edgelist=[(u, v)],
         ↪  edge_color='red', arrows=True, arrowsize=20)

    # Edge labels
    edge_labels = {(start_node[i], end_node[i]): weights[i] for i
     ↪  in range(len(start_node))}
    nx.draw_networkx_edge_labels(G_nx, pos,
     ↪  edge_labels=edge_labels, font_color='red')
```

```python
    plt.title('Graph with shortest path')
    plt.savefig("Graph.svg")

def run_example():
    """
    Find shortest path of an example graph and save visualization
    """
    start_node = [0,0,2,1,0,3]
    end_node =   [1,2,1,3,3,1]
    weights =    [3,1,1,1,5,3]

    G = {"start_node" : start_node,
        "end_node" : end_node,
        "weights" : weights}
    A = 0
    B = 1

    pi = dijkstra(G,A,B)
    plot_graph(G, pi)
    print(pi)

run_example()
```