MATH1058 coursework [40 marks]

Coursework summary

In this coursework, you are asked to code in Python the $O(n^3)$ and $O(n^2)$ versions of Dijkstra's algorithm that we've studied in the lectures. You are then asked to run your code (using a script we supply) to produce data on how the two variants behave computationally as the size of the graph increases, analyze this data in Excel, and, finally, write up your findings in LaTeX.

Part A: Python

(Make sure the Python packages `networkx`, `numpy`, and `xwlt` are installed and accessible on your machine.)

Open the (incomplete) Python script `dijkstra.py`.

The script contains two functions: `dijkstra1()` (which should implement the Dijkstra algorithm in its $O(n^3)$ version), and `dijkstra2()`(which should implement it in its $O(n^2)$ version). Comparing the practical efficiency of these two versions of the algorithm is the main aim of this activity.

The two functions must take as input:

- the successors list of the graph (call it `successors`; see further)
- the starting node (what we called s in the lecture slides)

Assume that, given a directed graph $G = (V, A)$, `successors` is implemented as a list of dictionaries where, for each $i \in V$, `successors[i]` is a dictionary whose keys are the indices of the nodes adjacent to node `i` and whose values are the lengths of the arcs from `i` to such nodes.

Notes and examples on the implementation

Consider the following graph:

Its successors list is:

```
successors = [{1 : 2, 2 : 5}, {2 : 3}, {3 : 4}, {0 : 21, 1: 8}]
```

Your implementations of `dijkstra1` and `dijkstra2` should be very similar to the pseudocode we have seen in the lectures. The only (technical) part where they may differ a bit is in the way the graph is navigated. Consider the following snippet, where `S` is a list containing a subset of the vertex indices:

```
#We loop over all nodes i in S
for i in S:
```

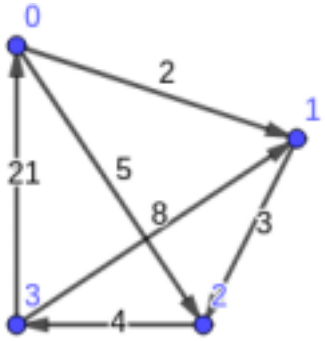

Figure 1: Graph represented by `successors`

```
#We loop over all nodes j which are successors of i
for j, length_ij in successors[i].items():
    #Do something
```

The code loops over all `i in S` and, for each such `i`, it then loops over its forward star by looping over `successors[i]`. In it, `j` is a successor of `i` and while `length_ij` is the length of the `(i,j)` arc.

Suggestion: once you have coded the first version of the algorithm, run `dijkstra.py` in Python. The script will run the two implementations you have provided on the small graph we discussed here. Print the two Python lists (`L` and `P`) that your functions return and verify that the values they contain are correct. The output should be:

```
0
({0: 0, 1: 2, 2: 5, 3: 9}, {0: '-', 1: 0, 2: 0, 3: 2})
({0: 0, 1: 2, 2: 5, 3: 9}, {0: '-', 1: 0, 2: 0, 3: 2})
1
({1: 0, 0: 28, 2: 3, 3: 7}, {1: '-', 0: 3, 2: 1, 3: 2})
({1: 0, 0: 28, 2: 3, 3: 7}, {1: '-', 0: 3, 2: 1, 3: 2})
2
({2: 0, 0: 25, 1: 12, 3: 4}, {2: '-', 0: 3, 1: 3, 3: 2})
({2: 0, 0: 25, 1: 12, 3: 4}, {2: '-', 0: 3, 1: 3, 3: 2})
3
({3: 0, 0: 21, 1: 8, 2: 11}, {3: '-', 0: 3, 1: 3, 2: 1})
({3: 0, 0: 21, 1: 8, 2: 11}, {3: '-', 0: 3, 1: 3, 2: 1})
```

where `0`, `1`, `2`, and `3` are the starting nodes and the first (respectively, second) line after each of them contains `L` and `P` for `dijkstra1` (respectively, `dijkstra2`).

Data generation

As you can see, the script `dijkstra.py` imports the script `math1058_cwk.py`. Opening it, you'll see that it contains the function `run_experiments()`, which takes as input:

- the name of a function (written by you) containing an implementation of Dijkstra's algorithm (either `dijkstra1` or `dijkstra1` in this coursework)
- your (8 digit) student ID (it contains mine; make sure you change it!)

`run_experiments()` generates a set of graphs (with random lengths depending on your ID) and runs the implementation you supplied on each of them. It also measures the amount of microseconds taken to execute the algorithm on each graph, and records it in an Excel file (see further).

Running `dijkstra.py` will, in turn, run import `math1058_cwk.py` and run `run_experiments()`. You should see an output similar to this one:

```
Your implementation named dijkstra1 will be run on a total of 12 graphs
Working on graph number 1 of size n = |V| = 8 and m = |A| = 12
Working on graph number 2 of size n = |V| = 16 and m = |A| = 56
Working on graph number 3 of size n = |V| = 24 and m = |A| = 132
Working on graph number 4 of size n = |V| = 16 and m = |A| = 24
Working on graph number 5 of size n = |V| = 32 and m = |A| = 112
Working on graph number 6 of size n = |V| = 48 and m = |A| = 264
Working on graph number 7 of size n = |V| = 24 and m = |A| = 36
Working on graph number 8 of size n = |V| = 48 and m = |A| = 168
Working on graph number 9 of size n = |V| = 72 and m = |A| = 396
Working on graph number 10 of size n = |V| = 32 and m = |A| = 48
Working on graph number 11 of size n = |V| = 64 and m = |A| = 224
Working on graph number 12 of size n = |V| = 96 and m = |A| = 528
...
Your implementation named dijkstra2 will be run on a total of 12 graphs
Working on graph number 1 of size n = |V| = 8 and m = |A| = 12
Working on graph number 2 of size n = |V| = 16 and m = |A| = 56
Working on graph number 3 of size n = |V| = 24 and m = |A| = 132
Working on graph number 4 of size n = |V| = 16 and m = |A| = 24
Working on graph number 5 of size n = |V| = 32 and m = |A| = 112
Working on graph number 6 of size n = |V| = 48 and m = |A| = 264
Working on graph number 7 of size n = |V| = 24 and m = |A| = 36
Working on graph number 8 of size n = |V| = 48 and m = |A| = 168
Working on graph number 9 of size n = |V| = 72 and m = |A| = 396
Working on graph number 10 of size n = |V| = 32 and m = |A| = 48
Working on graph number 11 of size n = |V| = 64 and m = |A| = 224
Working on graph number 12 of size n = |V| = 96 and m = |A| = 528
```

Part B: Excel

When called as specified before in `dijkstra.py`, `run_experiments()` will produce two .xls files: `dijkstra1_data.xls` and `dijkstra2_data.xls`. Each of them features a column for each graph on which the experiments have been run, containing:

- the number of nodes in the graph `n = |V|`
- the number of arcs in the graph `m = |A|`
- exactly `n` rows reporting, for each vertex in the graph, the computing time (in microseconds) measured when running the implementation you supplied adopting that vertex as s.

The two files should contain 12 columns with data, one for each of the graph the experiments have been run on. As you can see, the 12 graphs have 8 distinct values of n.

You are asked to create a new spreadsheet called `analysis.xls`.

Part B.I

For each value of n, `analysis.xls` should contain a column reporting:

- the number of nodes in the graph (n)
- the average computing time of `dijkstra1` (y_1)
- the average computing time of `dijkstra2` (y_2)

Note: as some of the graphs on which the experiments have been run feature the same number of vertices, some of the columns of `dijkstra1.xls` and `dijkstra2.xls` should be merged before computing the averages!

Part B.II

Draw a chart plotting y_1 versus n and y_2 versus n.

Part B.III

Estimate two power-law functions to capture the relationship between y_1 and n and y_2 and n. The two functions should be of the following nature:

- $y_1 = c_1 n^{k_1}$
- $y_2 = c_2 n^{k_2}$

You are asked to determine the coefficients of these functions via linear (or affine, to be more precise) regression.

For $i = 1, 2$, the relationship

$$y_i = c_i n^{k_i}$$

implies

$$\log y_i = \log c_i + k_i \log n.$$

(This holds since $n, k_i, c_i > 0$).

Letting now $\bar{y_i} := \log y_i$ and $\bar{n} := \log n$, we deduce the affine relationship:

$$\bar{y_i} = \log c_i + k_i \bar{n}$$

where $\log c_i$ and k_i are, respectively, the slope and intercept of an affine function which maps $\bar{n}$ into $\bar{y_i}$.

Part B.III.A

You are asked to report in `analysis.xls` a copy of the previous table whose values are all equal to the logarithms of those in the previous table. Namely, the table should contain:

- the logarithm of the number of nodes in the graph ($\log(n)$)
- the logarithm of the average computing time of `dijkstra1` ($\log(y_1)$)
- the logarithm of the average computing time of `dijkstra2` ($\log(y_2)$)

Part B.III.B

Draw a chart plotting $\log(y_1)$ versus $\log(n)$ and $\log(y_2)$ versus $\log(n)$.

Part B.III.C

Report the values of $\log c_i$ and k_i for $i = 1, 2$ by affine regression; the slope c_i can be found using the function `SLOPE`; the intercept using the function `INTERCEPT`.

Part C: LaTeX report

Complete the the supplied latex file `report.tex` by answering the questions it contains, and then compile it into a PDF. *No other format will be accepted.*

Caveat: marks are earned based uniquely on what you wrote in your report!

What to submit

- `dijkstra.py`
- `analysis.xls`
- `report.zip` (a compressed archive containing all the files necessary to compile your report—do not forget to include the figures!)
- `report.pdf`: your report as a PDF file