Reachable Cities - Comment

Hannes Guth

January 22, 2023

Comment on the functionality

This project aims to find all reachable cities from a given starting city, using only a given number k steps. Two important remarks should be made right at the beginning. The algorithm takes about 2.5 hours to run but does principally not loose remarkable performance when k is increased. It does not deliver the same results as in the sample output because it looks for the shortest path to reach a city, not any path of this length.
The program consists of 4 functions that are executed consecutively.

At first, the function `getData()` is called. It retrieves the necessary data with one single SQL query, getting name, country code, longitude, latitude, lake, river and sea of every city in the database. Missing values, *Null values*, are replaced by "XXX" to make the following steps easier. These data are stored in a list of lists and are given to the next step.

The second function which is the `neighborhood()` function. This function creates at first an array with all possible combinations of all cities. This array will later be used to calculate the shortest paths between all cities and will contain distances. In this first step, it shall only indicate if two cities are neighbors or not and is filled with the value 1000 (very big distance, representing only that one does not know the shortest path between these two cities and has nothing to do with the later calculated distance from the longitude and latitude) for all combinations of cities. In the second step, these values are updated for every combination of cities and set to 1 if this combination contains neighbors as defined in the task. If they are not neighbors, the 1000 is kept. If the two cities are the same city, a 0 is inserted. With this, the information about the immediate reachability (only 1 step necessary) is obtained. This array will be returned.

The third function contains an implementation of the Bellman-Ford algorithm and is therefore called `bellman_ford_algorithm()`. This algorithm finds the shortest path from every city to any other city in the previously calculated table. So it can be assured that no edge or city is used twice. In this example, "short" refers not to distance itself but to the necessary number of steps to reach any city in a column from any city in a row (and vice versa). In the following, the algorithm will be explained briefly. As said, one begins with an unoptimized distance matrix/array (`U1` in the code) where it is only known which cities are neighbors and which are not. In each iteration/run of the algorithm, every single value of this distance matrix is checked and possibly updated when improvement is possible. This works as follows. Every entry has a row and a column. The entry's row of the currently "best" distance matrix (`U1compare`) and the entry's column of the very first distance matrix (`U1`) are taken. Since the matrix/array has as many rows as columns, they are equal in length. The first element of the row is taken and added to the first element of the selected column. This is done for all the elements of the row/column and the results are saved in a vector. Should there be any value in the vector that is smaller than the initial entry (from which one selected the row and the column), this entry will be replaced by the smaller value (resulting in `U1new`) in the next run. Only if there has been any replacements in the current run, the matrix will differ from the one of the previous run. Only then, there will be a next round (`stp = False`). If there is no next round, the algorithm ends and no further optimization is possible. More information on this algorithm and a proof for the algorithm delivering the best possible result can be found in [Khi]. The optimized distance array is returned and taken as input for the next function. (In order to minimize runtime when not changing the database, this array can be saved and only the last function run on this array. This should save nearly 100% of computational time.)

The last function `return_reachable_cities()` takes the optimized distance array. It also gets the city name, the country code and the desired number of steps as input values. It then executes the query from the beginning again and looks for the index of the desired starting city. This index is the same as in the distance array. At this index, the program looks for distances smaller than k (for cities reachable with k or fewer steps) or equal to k (for cities reachable with exactly k steps). Since the Bellman-Ford-Algorithm gives the shortest path, there is no need to consider the threat of possibly visiting a city twice or using an edge twice on the way.

I did not collaborate with anybody in this project.

Outcome

In the following, the outcome of the program is presented.

```
Steps: 1
Reachable cities with 1 steps ( 6 ): {'Sion (CH)', 'Turin (I)', 'Besancon (F)', 'Villeurbanne
(F)', 'Lausanne (CH)', 'Bern (CH)'}

Exactly in 1 steps reachable cities ( 6 ): {'Sion (CH)', 'Turin (I)', 'Besancon (F)',
'Villeurbanne (F)', 'Lausanne (CH)', 'Bern (CH)'}


Steps: 2
Reachable cities with 2 steps ( 14 ): {'Sion (CH)', 'Lyon (F)', 'Turin (I)', 'Nice (F)',
'Besancon (F)', 'Villeurbanne (F)', 'Monaco (MC)', 'Lausanne (CH)', 'Solothurn (CH)',
'Bern (CH)', 'Nancy (F)', 'Piacenza (I)', 'Grenoble (F)', 'Aarau (CH)'}

Exactly in 2 steps reachable cities ( 8 ): {'Lyon (F)', 'Nice (F)', 'Monaco (MC)', 'Solothurn
(CH)', 'Nancy (F)', 'Piacenza (I)', 'Grenoble (F)', 'Aarau (CH)'}


Steps: 3
Reachable cities with 3 steps ( 23 ): {'Sion (CH)', 'Saarbrucken (D)', 'Ajaccio (F)',
'Nancy (F)', 'Solothurn (CH)', 'Aarau (CH)', 'Lyon (F)', 'Genua (I)', 'Monaco (MC)', 'Lausanne
(CH)', 'Koblenz (D)', 'Metz (F)', 'Bern (CH)', 'Piacenza (I)', 'Grenoble (F)', 'Strasbourg
(F)', 'Turin (I)', 'Nice (F)', 'Besancon (F)', 'Villeurbanne (F)', 'Rimini (I)', 'Toulon
(F)', 'Marseille (F)'}

Exactly in 3 steps reachable cities ( 9 ): {'Saarbrucken (D)', 'Genua (I)', 'Rimini (I)',
'Toulon (F)', 'Koblenz (D)', 'Metz (F)', 'Ajaccio (F)', 'Marseille (F)', 'Strasbourg (F)'}


Steps: 4
Reachable cities with 4 steps ( 46 ): {'Sion (CH)', 'Saarbrucken (D)', 'Koln (D)', 'Stuttgart
(D)', 'Chalons en Champagne (F)', 'Ludwigshafen (D)', 'Rotterdam (NL)', 'Mannheim (D)',
'Ajaccio (F)', 'Leverkusen (D)', 'Neuss (D)', 'Krefeld (D)', 'Nancy (F)', 'Schaffhausen
(CH)', 'Wiesbaden (D)', 'Marseille (F)', 'Arnhem (NL)', 'Solothurn (CH)', 'Basel (CH)',
'Aarau (CH)', 'Liege (B)', 'Lyon (F)', 'Genua (I)', 'Cagliari (I)', 'Duisburg (D)', 'Monaco
(MC)', 'Lausanne (CH)', 'Karlsruhe (D)', 'Koblenz (D)', 'Metz (F)', 'Bern (CH)', 'Piacenza
(I)', 'Grenoble (F)', 'Strasbourg (F)', 'La Spezia (I)', 'Chur (CH)', 'Mainz (D)', 'Nice
(F)', 'Turin (I)', 'Besancon (F)', 'Villeurbanne (F)', 'Rimini (I)', 'Nijmegen (NL)',
'Toulon (F)', 'Dusseldorf (D)', 'Bonn (D)'}
```

```
Exactly in 4 steps reachable cities ( 24 ): {'Koln (D)', 'Stuttgart (D)', 'Chalons en
Champagne (F)', 'Ludwigshafen (D)', 'Rotterdam (NL)', 'Mannheim (D)', 'Leverkusen (D)',
'Neuss (D)', 'Krefeld (D)', 'Schaffhausen (CH)', 'Wiesbaden (D)', 'Arnhem (NL)', 'Basel
(CH)', 'Liege (B)', 'Cagliari (I)', 'Duisburg (D)', 'Karlsruhe (D)', 'Koblenz (D)', 'La
Spezia (I)', 'Chur (CH)', 'Mainz (D)', 'Nijmegen (NL)', 'Dusseldorf (D)', 'Bonn (D)'}


Steps: 5
Reachable cities with 5 steps ( 57 ): {'Sion (CH)', 'Bregenz (A)', 'Saarbrucken (D)',
'Koln (D)', 'Heilbronn (D)', 'Stuttgart (D)', 'Chalons en Champagne (F)', 'Ludwigshafen
(D)', 'Rotterdam (NL)', 'Mannheim (D)', 'Ajaccio (F)', 'Leverkusen (D)', 'Neuss (D)',
'Krefeld (D)', 'Nancy (F)', 'Schaffhausen (CH)', 'Wiesbaden (D)', 'Marseille (F)', 'Arnhem
(NL)', 'Maastricht (NL)', 'Tunis (TN)', 'Solothurn (CH)', 'Wurzburg (D)', 'Basel (CH)',
'Aarau (CH)', 'Liege (B)', 'Ulm (D)', 'Lyon (F)', 'Genua (I)', 'Cagliari (I)', 'Duisburg
(D)', 'Monaco (MC)', 'Lausanne (CH)', 'Karlsruhe (D)', 'Koblenz (D)', 'Metz (F)', 'Bern
(CH)', 'Piacenza (I)', 'Grenoble (F)', 'Strasbourg (F)', 'Heidelberg (D)', 'La Spezia
(I)', 'Chur (CH)', 'Mainz (D)', 'Nice (F)', 'Turin (I)', 'Besancon (F)', 'Frankfurt am
Main (D)', 'Villeurbanne (F)', 'Rimini (I)', 'Nijmegen (NL)', 'Toulon (F)', 'Namur (B)',
'Dusseldorf (D)', 'Kassel (D)', 'Bonn (D)', 'Offenbach am Main (D)'}

Exactly in 5 steps reachable cities ( 11 ): {'Heidelberg (D)', 'Maastricht (NL)', 'Bregenz
(A)', 'Ulm (D)', 'Heilbronn (D)', 'Tunis (TN)', 'Frankfurt am Main (D)', 'Wurzburg (D)',
'Namur (B)', 'Kassel (D)', 'Offenbach am Main (D)'}


Steps: 6
Reachable cities with 6 steps ( 69 ): {'Sion (CH)', 'Bregenz (A)', 'Saarbrucken (D)',
'Koln (D)', 'Heilbronn (D)', 'Stuttgart (D)', 'Bratislava (SK)', 'Chalons en Champagne
(F)', 'Ludwigshafen (D)', 'Rotterdam (NL)', 'Mannheim (D)', 'Ajaccio (F)', 'Leverkusen
(D)', 'Neuss (D)', 'Krefeld (D)', 'Nancy (F)', 'Schaffhausen (CH)', 'Wiesbaden (D)', 'Marseille
(F)', 'Arnhem (NL)', 'Regensburg (D)', 'Maastricht (NL)', 'Tunis (TN)', 'Belgrade (SRB)',
'Solothurn (CH)', 'Innsbruck (A)', 'Budapest (H)', 'Basel (CH)', 'Wurzburg (D)', 'Aarau
(CH)', 'Liege (B)', 'Ulm (D)', 'Lyon (F)', 'Munich (D)', 'Genua (I)', 'Ingolstadt (D)',
'Cagliari (I)', 'Duisburg (D)', 'Monaco (MC)', 'Lausanne (CH)', 'Vienna (A)', 'Karlsruhe
(D)', 'Linz (A)', 'Braila (RO)', 'Koblenz (D)', 'Metz (F)', 'Bern (CH)', 'Piacenza (I)',
'Grenoble (F)', 'Strasbourg (F)', 'Heidelberg (D)', 'La Spezia (I)', 'Chur (CH)', 'Mainz
(D)', 'Nice (F)', 'Galati (RO)', 'Besancon (F)', 'Frankfurt am Main (D)', 'Turin (I)',
'Rimini (I)', 'Villeurbanne (F)', 'Nijmegen (NL)', 'Toulon (F)', 'Namur (B)', 'Augsburg
(D)', 'Dusseldorf (D)', 'Kassel (D)', 'Bonn (D)', 'Offenbach am Main (D)'}

Exactly in 6 steps reachable cities ( 12 ): {'Munich (D)', 'Ingolstadt (D)', 'Galati (RO)',
'Bratislava (SK)', 'Belgrade (SRB)', 'Vienna (A)', 'Innsbruck (A)', 'Budapest (H)', 'Linz
(A)', 'Braila (RO)', 'Augsburg (D)', 'Regensburg (D)'}


Steps: 7
Reachable cities with 7 steps ( 73 ): {'Sion (CH)', 'Bregenz (A)', 'Saarbrucken (D)',
'Koln (D)', 'Heilbronn (D)', 'Stuttgart (D)', 'Bratislava (SK)', 'Chalons en Champagne
(F)', 'Ludwigshafen (D)', 'Rotterdam (NL)', 'Mannheim (D)', 'Ajaccio (F)', 'Leverkusen
(D)', 'Neuss (D)', 'Krefeld (D)', 'Nancy (F)', 'Schaffhausen (CH)', 'Wiesbaden (D)', 'Marseille
(F)', 'Arnhem (NL)', 'Regensburg (D)', 'Maastricht (NL)', 'Tunis (TN)', 'Belgrade (SRB)',
'Solothurn (CH)', 'Innsbruck (A)', 'Budapest (H)', 'Basel (CH)', 'Wurzburg (D)', 'Graz
(A)', 'Aarau (CH)', 'Liege (B)', 'Trento (I)', 'Ulm (D)', 'Lyon (F)', 'Munich (D)', 'Genua
(I)', 'Ingolstadt (D)', 'Cagliari (I)', 'Duisburg (D)', 'Monaco (MC)', 'Lausanne (CH)',
'Vienna (A)', 'Karlsruhe (D)', 'Linz (A)', 'Braila (RO)', 'Koblenz (D)', 'Metz (F)', 'Bern
```

```
(CH)', 'Piacenza (I)', 'Grenoble (F)', 'Strasbourg (F)', 'Heidelberg (D)', 'La Spezia
(I)', 'Chur (CH)', 'Mainz (D)', 'Nice (F)', 'Galati (RO)', 'Besancon (F)', 'Frankfurt
am Main (D)', 'Salzburg (A)', 'Rimini (I)', 'Turin (I)', 'Villeurbanne (F)', 'Bolzano
(I)', 'Nijmegen (NL)', 'Namur (B)', 'Toulon (F)', 'Augsburg (D)', 'Dusseldorf (D)', 'Kassel
(D)', 'Bonn (D)', 'Offenbach am Main (D)'}

Exactly in 7 steps reachable cities ( 6 ): {'Trento (I)', 'Salzburg (A)', 'Bratislava
(SK)', 'Belgrade (SRB)', 'Bolzano (I)', 'Graz (A)'}


Steps: 8
Reachable cities with 8 steps ( 78 ): {'Saarbrucken (D)', 'Koln (D)', 'Stuttgart (D)',
'Chalons en Champagne (F)', 'Rotterdam (NL)', 'Ajaccio (F)', 'Wiesbaden (D)', 'Nancy (F)',
'Belgrade (SRB)', 'Solothurn (CH)', 'Budapest (H)', 'Cagliari (I)', 'Verona (I)', 'Braila
(RO)', 'Koblenz (D)', 'Bern (CH)', 'Piacenza (I)', 'Strasbourg (F)', 'Kassel (D)', 'Leverkusen
(D)', 'Krefeld (D)', 'Schaffhausen (CH)', 'Regensburg (D)', 'Ulm (D)', 'Trento (I)', 'Lyon
(F)', 'Linz (A)', 'Chur (CH)', 'Turin (I)', 'Besancon (F)', 'Villeurbanne (F)', 'Namur
(B)', 'Dusseldorf (D)', 'Bonn (D)', 'Bregenz (A)', 'Heilbronn (D)', 'Mannheim (D)', 'Tunis
(TN)', 'Graz (A)', 'Aarau (CH)', 'Duisburg (D)', 'Monaco (MC)', 'Lausanne (CH)', 'Vienna
(A)', 'Ljubljana (SLO)', 'La Spezia (I)', 'Mainz (D)', 'Nice (F)', 'Salzburg (A)', 'Frankfurt
am Main (D)', 'Rimini (I)', 'Zagreb (HR)', 'Bolzano (I)', 'Nijmegen (NL)', 'Olomouc (CZ)',
'Sion (CH)', 'Bratislava (SK)', 'Ludwigshafen (D)', 'Neuss (D)', 'Arnhem (NL)', 'Maastricht
(NL)', 'Innsbruck (A)', 'Wurzburg (D)', 'Basel (CH)', 'Liege (B)', 'Venice (I)', 'Munich
(D)', 'Genua (I)', 'Ingolstadt (D)', 'Karlsruhe (D)', 'Metz (F)', 'Grenoble (F)', 'Heidelberg
(D)', 'Galati (RO)', 'Toulon (F)', 'Augsburg (D)', 'Marseille (F)', 'Offenbach am Main
(D)'}

Exactly in 8 steps reachable cities ( 5 ): {'Venice (I)', 'Verona (I)', 'Zagreb (HR)',
'Ljubljana (SLO)', 'Olomouc (CZ)'}


Steps: 9
Reachable cities with 9 steps ( 81 ): {'Saarbrucken (D)', 'Koln (D)', 'Stuttgart (D)',
'Chalons en Champagne (F)', 'Ostrava (CZ)', 'Rotterdam (NL)', 'Ajaccio (F)', 'Wiesbaden
(D)', 'Nancy (F)', 'Belgrade (SRB)', 'Solothurn (CH)', 'Budapest (H)', 'Cagliari (I)',
'Verona (I)', 'Braila (RO)', 'Koblenz (D)', 'Bern (CH)', 'Piacenza (I)', 'Strasbourg (F)',
'Kassel (D)', 'Leverkusen (D)', 'Krefeld (D)', 'Schaffhausen (CH)', 'Regensburg (D)',
'Ulm (D)', 'Trento (I)', 'Lyon (F)', 'Linz (A)', 'Chur (CH)', 'Turin (I)', 'Besancon (F)',
'Villeurbanne (F)', 'Namur (B)', 'Dusseldorf (D)', 'Bonn (D)', 'Bregenz (A)', 'Heilbronn
(D)', 'Mannheim (D)', 'Tunis (TN)', 'Graz (A)', 'Aarau (CH)', 'Duisburg (D)', 'Monaco
(MC)', 'Lausanne (CH)', 'Trieste (I)', 'Vienna (A)', 'Ljubljana (SLO)', 'La Spezia (I)',
'Mainz (D)', 'Nice (F)', 'Salzburg (A)', 'Frankfurt am Main (D)', 'Rimini (I)', 'Zagreb
(HR)', 'Bolzano (I)', 'Nijmegen (NL)', 'Ancona (I)', 'Olomouc (CZ)', 'Sion (CH)', 'Bratislava
(SK)', 'Ludwigshafen (D)', 'Neuss (D)', 'Arnhem (NL)', 'Maastricht (NL)', 'Innsbruck (A)',
'Wurzburg (D)', 'Basel (CH)', 'Liege (B)', 'Venice (I)', 'Munich (D)', 'Genua (I)', 'Ingolstadt
(D)', 'Karlsruhe (D)', 'Metz (F)', 'Grenoble (F)', 'Heidelberg (D)', 'Galati (RO)', 'Toulon
(F)', 'Augsburg (D)', 'Marseille (F)', 'Offenbach am Main (D)'}

Exactly in 9 steps reachable cities ( 3 ): {'Trieste (I)', 'Ostrava (CZ)', 'Ancona (I)'}


Steps: 10
Reachable cities with 10 steps ( 85 ): {'Saarbrucken (D)', 'Koln (D)', 'Stuttgart (D)',
'Chalons en Champagne (F)', 'Ostrava (CZ)', 'Rotterdam (NL)', 'Ajaccio (F)', 'Wiesbaden
(D)', 'Nancy (F)', 'Belgrade (SRB)', 'Solothurn (CH)', 'Budapest (H)', 'Cagliari (I)',
'Verona (I)', 'Braila (RO)', 'Koblenz (D)', 'Bern (CH)', 'Piacenza (I)', 'Strasbourg (F)',
'Kassel (D)', 'Leverkusen (D)', 'Krefeld (D)', 'Schaffhausen (CH)', 'Regensburg (D)',
'Ulm (D)', 'Trento (I)', 'Wroclaw (PL)', 'Lyon (F)', 'Linz (A)', 'Chur (CH)', 'Turin (I)',
'Besancon (F)', 'Villeurbanne (F)', 'Namur (B)', 'Dusseldorf (D)', 'Bonn (D)', 'Bregenz
(A)', 'Heilbronn (D)', 'Mannheim (D)', 'Szczecin (PL)', 'Tunis (TN)', 'Graz (A)', 'Aarau
(CH)', 'Duisburg (D)', 'Monaco (MC)', 'Lausanne (CH)', 'Trieste (I)', 'Vienna (A)', 'Ljubljana
(SLO)', 'La Spezia (I)', 'Mainz (D)', 'Nice (F)', 'Salzburg (A)', 'Frankfurt am Main (D)',
'Rimini (I)', 'Zagreb (HR)', 'Bolzano (I)', 'Nijmegen (NL)', 'Ancona (I)', 'Olomouc (CZ)',
'Sion (CH)', 'Bratislava (SK)', 'Ludwigshafen (D)', 'Neuss (D)', 'Arnhem (NL)', 'Maastricht
(NL)', 'Innsbruck (A)', 'Wurzburg (D)', 'Basel (CH)', 'Napoli (I)', 'Liege (B)', 'Venice
(I)', 'Munich (D)', 'Genua (I)', 'Ingolstadt (D)', 'Karlsruhe (D)', 'Metz (F)', 'Grenoble
(F)', 'Heidelberg (D)', 'Galati (RO)', 'Toulon (F)', 'Opole (PL)', 'Augsburg (D)', 'Marseille
(F)', 'Offenbach am Main (D)'}

Exactly in 10 steps reachable cities ( 4 ): {'Wroclaw (PL)', 'Szczecin (PL)', 'Opole (PL)',
'Napoli (I)'}
```

References

[Khi] Alex Chumbley; Karleigh Moore; Eli Ross; Jimin Khim. Bellman-ford algorithm. https://brilliant.org/wiki/bellman-ford-algorithm/#algorithm-proof. Accessed on 2023-01-21.