III Graph Theory

KTH/ICT:IX1500 - Discrete Mathematics, v1
Niharika Gauraha
Orginal notebook by Göran Andersson

1. Implementation and Assessment

The course includes two-three project tasks on a total of 4 hp. The projects are assessed in writing and orally. They will be given a summary grade in scale A-F. The assessment includes the number of solved project tasks. The first two projects are mandatory for grade C-E. The third project task is optional and targets higher grades A-B and requires a grade C for the first two projects.

In this project task you will work alone and solve a mathematical task, write a report in *Mathematica* and prepare a short oral presentation on your solution to the task.

Carefully read the following information so that you know which rules apply and what is expected of you.

1.1 Report

The report should be written in Mathematica and contain

- title and author of the report
- email@kth.se
- a summary containing the results of resolved parts
- separate sections for each part containing e.g.
  - mathematical formulas and equations
  - a brief discussion
  - explanatory diagrams
  - your conclusions.
- separate code section (do not mix code and text with conclusions and results)

The report will normally be uploaded a couple of days prior to the examination (see 1.5 below).

1.2 Oral Presentation

You should prepare an oral presentation of your solution to the task. The presentation will be carried out with your own laptop and *should not take more than five minutes*, effective time. A computer projector will be available at the presentation. Please carefully consider your presentation, what is important, in what order and how it will be illustrated. Practice the presentation in advance and make sure to meet the time frame.

1.3 Rules

The task is considered *individually* and *assumes that you have full knowledge of all the material you*

are presenting. In order to be approved, then you have solved the task and be able to explain the entire task and solution.

To account for the task that you do not have solved is considered cheating. It is also cheating to copy the solution or part of a solution from another. If two solutions are presented as (partially) copies they are rejected both.

If the solution contains parts, e.g. background material, which you do not have produced, you must clearly indicate this and indicate the source.

Suspicion of cheating or misleading can be reported to the Disciplinary Board.

1.4 Examination

- The examination is carried out according to the booking in canvas.
- The booking is done according to information in Canvas.
- Please notice that this project should be reported in **by individual students**.
- The report (including code section) should be uploaded according to the information in Canvas.
- The opposition report should be uploaded according to the information in Canvas.

2. Preparations

2.1 Study

Read and study the following texts:

- Chapter 9 in the course book (Böiers)
- lecture notes F14-F17
- Relevant parts of the Mathematica reference

2.2 Geographics example

The following describes how to use GeoGraphics in Mathematica

```
In[•]:= ClearAll["`*"]
```

Defining the cities and links between the cities. City number is given by the index in the first list.

```
In[•]:= city = {"Uppsala", "Stockholm", "Goteborg", "Malmo"};
        links = {{1, 2}, {2, 3}, {3, 4}};
```

Getting the geo coordinates for the cities by using the function CityData.

```
In[•]:= c = Table[CityData[city[[i]], "Coordinates"], {i, Length[city]}]
```

Describing the adjacency matrix for the links between the cities.

```
In[•]:= A = {{0, 1, 0, 0},
           {1, 0, 1, 0},
           {0, 1, 0, 1},
           {0, 0, 1, 0}};
```

```
In[•]:= 𝒢 = AdjacencyGraph[𝔸]
```

Extracting the list of vertices and edges from the Graph representation.

```
In[•]:= v = VertexList[𝒢];
        ∈ = EdgeList[𝒢];
```

Setting the options for geoGraph function

```
In[•]:= Options[geoGraph] = {
          EdgeStyle → Black, VertexStyle → GeoStyling[None],
          VertexLabels → None, VertexLabelStyle → Black
        };
```

Definition of the function geoGraph which results in a list with the geo-data for the cities and links.

```
In[•]:= geoGraph[v_, e_, c_, r_, OptionsPattern[]] :=
         Module[{
           es = OptionValue[EdgeStyle], vs = OptionValue[VertexStyle],
           vl = OptionValue[VertexLabels], vls = OptionValue[VertexLabelStyle],
           lbl, Δ = {0.1, 0.3}
          },
          If[Length[vl] ≠ Length[v], vl = None];
          lbl = If[
            vl === None, {},
            Join[{vls}, MapThread[Text[#1, GeoPosition[#2 + Δ], {-1, 0}] &, {vl, c}]]
           ];
          Join[
           Join[{vs}, (GeoDisk[c[[#1]], r] &) /@ v],
           Join[{es}, (GeoPath[{c[[#1[[1]]], c[[#1[[2]]]}] &) /@ e],
           lbl
          ]
         ]
```

Radius of disk in GeoDisk function [m], that is the size of the dots representing the cities.

```
In[•]:= r = 10 × 10^3;
```

Drawing a map of Sweden with the links between the cities.

```
In[•]:= GeoGraphics[{
          Polygon[CountryData["Sweden"]],
          geoGraph[v, ∈, c, r, EdgeStyle → Blue,
           VertexStyle → GeoStyling[Red], VertexLabels → city]
         }, GeoBackground → None, GeoProjection → "LambertAzimuthal"]
```

Tip : Study the documentation for the function WeightedAdjacencyGraph .

3. Mathematical Task

💡 Use KTH Canvas to discuss Mathematica code.

Tasks

a) For B-grade (provided you have grade C for project 1 and 2).

Study Kruskal's algorithm for a minimal spanning tree from a reliable source. Be sure to use detailed references in your explanation below. You can **choose between two different tasks**:

1. Write your own method in Mathematica using the algorithm to produce a minimal spanning tree. Illustrate the features of your method with graph examples. You are **expected to have insight** in how the algorithm works not just the code.
2. Explain **how** Kruskal's algorithm works. Use Mathematica graph-methods to illustrate your explanation with an animation. The explanation should be with the focus on how the algorithm works, not your code.

b) For A−grade (provided you have done part a).

In the guidance section 2.2, it was shown how relations between cities can be illustrated. Let us assume that each city represents a router that forwards data packets to their final destination. Your task this time is to create a tree that shows how data packets are routed so that each city can be reached by the information flow that (this time) is based on Stockholm.

```
city = {"Abisko", "Boden", "Falun", "Goteborg", "Hoganas",
   "Hudiksvall", "Jonkoping", "Kalmar", "Kiruna", "Lidkoping", "Linkoping",
   "Lulea", "Lund", "Malmo", "Mariestad", "Ostersund", "Stockholm",
   "Strangnas", "Timra", "Uppsala", "Umea", "Varberg", "Visby"};

links = {{15, 18}, {13, 17}, {3, 16}, {6, 3}, {18, 6}, {12, 2}, {15, 11}, {19, 21}, {7, 8},
   {19, 2}, {7, 4}, {21, 17}, {17, 11}, {23, 11}, {10, 7}, {16, 17}, {10, 20},
   {13, 5}, {3, 17}, {7, 22}, {15, 10}, {16, 18}, {11, 16}, {17, 23}, {18, 17},
   {20, 6}, {11, 7}, {9, 2}, {3, 20}, {4, 17}, {19, 17}, {7, 20}, {22, 4}, {14, 7},
   {15, 17}, {6, 17}, {20, 5}, {16, 15}, {11, 3}, {9, 1}, {4, 10}, {5, 14}, {6, 16},
   {16, 19}, {13, 8}, {8, 23}, {16, 10}, {4, 13}, {2, 16}, {15, 3}, {20, 18}};

city[[links[[2]]]]
```

- Weight (cost) for the use of a link between the two cities is assumed to be inversely proportional to the free capacity of the link. Suppose, for example, the following available capacity:

Stockholm	–	Göteborg	25
Stockholm	–	Lund	30
Göteborg	–	Lund	25
Stockholm	–	Falun	15
Falun	–	Östersund	15
Östersund	–	Umeå	15

  The available capacity of the other available links to random numbers between 1 and 10.

- Your solution should be optimized so that each router can be accessed with minimal weight (cost). What algorithm is used (justify your answer)?
- Illustrate your solution with the map of Sweden.
- What happens if the link Göteborg – Lund stops working?