AI: Search, planning and machine learning
Programming task 1 (Search block)

Implement Prim's algorithm for growing a MST in Java (version 11 or later) or Python (version 3.6 or later). Take care how to represent the input graph G and its components (edges, vertices), and use an efficient priority queue library class (preferably based on a binary heap or fibonacci heap). You can find library classes for such efficient priority queues in any programming language.
Include a (possibly rough) runtime measure in your implementation, e.g. a counter for every time an edge is considered.

Experiment with your implementation:
- How does the runtime change for increasing numbers of vertices $|V|$ in G?
- And how if the number of edges $|E|$ increases from $|V|-1$ to $|V| \times (|V|-1)/2$ for different $|V|$?
- Does the runtime change in relation to the amount of equal edge-weights (i.e. for the extreme cases where all weights are equal or all weights are different, and intermediate cases)?
- Does the runtime change in relation to the (minimum, maximum, average) magnitude of the edge weights, i.e. small weights like 1, 2 vs. large weights like 2000, 15000?
- And how does the range in edge weights effect the runtime, e.g. a range of $1\ldots|V|$ vs. a wider range of $1\ldots\text{Max} > |V|$ or even $1\ldots\infty$?
- Any other interesting aspects to experiment with?

Write a short report, according to the IMRAD method in which you
- Describe the main aspects of your implementation, and argue the choices you made;
- Indicate how you measure the runtime, also answering the question "why not count each edge that's added to the evolving MST?";
- Analyze the outcomes of your experiments.

You report should match the IMRAD method for reporting experimental work, using an introduction, methods, results, discussion, and conclusion section. Here, you write respectively "why" you did the research (including research questions and some context), "how" you did it, "what" you found, "how" you interpret the raw results, and "what" the interpreted results contribute to the research questions. Your report must be submitted as .pdf document.

Bonus points can be obtained when you are able to read in a graph from a file (such as GML: see https://en.wikipedia.org/wiki/Graph_Modelling_Language) and/or when you can generate random graphs. At the very least your implementation should hard-code several graphs and return the results per graph. Please hand in your report (pdf format!), the source code (zipped), and an execution log (.txt format) on Brightspace.

Also check out the assessment form on Brightspace!

The **programming language** for the task is:

1. Java (version 11 or later), or
2. Python (version 3.6 or later)

To hand in:

Both report (in .pdf) and (zipped) code. Do not forget your name(s), student number(s), course, number of the task, date, etc. Hand in via Brightspace. If you submit in a couple, let one of the students submit report and code and the other just a text file with the name of the other student for easy reference.

Deadline:

Friday 2-10-2020 23.59 via Brightspace.