Exercise session 5: Weights, distances, flows · 1MA020

Vilhelm Agdur[1]

[1] vilhelm.agdur@math.uu.se

6 November 2023

In the previous lecture, we learned how to count the number of spanning trees of a graph. Now, we study how to find spanning trees, and minimal spanning trees. We also think about the vertex version of Eulerian circuits, which are called Hamilton paths. Finally, we consider flows in graphs.

Weighted graphs and minimum spanning trees

We saw in our final theorem of the previous lecture that there is an effective way of computing how many spanning trees there are. Is there also a good way of finding one? The answer to this is yes – and in fact we can do something much stronger in an efficient way.

To explain the problem we are concerned with, let us introduce the notion of a *weighted* graph. We will consider simple weighted graphs, but the same notion makes perfect sense also for multigraphs and directed graphs.

Definition 1. A *weighted graph* is a simple graph $G = (V, E)$ together with a *weight function* $w : E \to \mathbb{R}$. If $H = (V', E')$ is a subgraph of G, its *weight* is defined as $w(H) = \sum_{e \in E'} w(e)$, whenever this is well defined.[2] A *minimum spanning tree* (MST) is a spanning tree T of G such that $w(T)$ is minimal among all spanning trees.

[2] If it contains infinitely many positive and infinitely many negative weights, the sum might not converge, but this case is entirely pathological and won't occur for us.

It is clear that for finite weighted graphs, there always exists at least one minimum spanning tree, but they are not necessarily unique – if all edges have the same weight, then of course every spanning tree is minimal. However, if all edges are given different weights, then the minimum spanning tree is indeed unique.

Exercise 1. The things we do here don't work at all for infinite graphs. To demonstrate this, find an infinite weighted graph without a minimum spanning tree. Can you find one with only positive weights?

Exercise 2. Prove that if all edges have different weights, then the minimum spanning tree is unique.

So we have seen that for finite directed graphs, there does exist a unique minimal spanning tree. How do we find it?

Exercise 3. Come up with a reasonable[3] method for finding a minimum spanning tree. It might be useful here to consider the result we proved two lectures ago, giving four equivalent statements, where one of them was "T is a tree".

[3] "Reasonable" here meaning something like at least not brute force, or in polynomial time. The two algorithms we'll look at in the next lecture, and which I'm fishing for with this exercise, run in time $O(|E| + |V| \log(|V|))$ and $|E| \log(|V|)$ respectively, if implemented optimally.

One approach might be to build it up by starting at one vertex and then adding in neighbours, and one might be to build it up one edge at a time, making sure never to create a cycle.

This exercise is likely one of the harder ones on this sheet, so if you get stuck, move on to the later ones.

If all the weights on our graph are positive, we can interpret the weights on our graph as being distances between vertices, or time it takes to travel between them.[4] Then it makes sense to make the following definition:

[4] If we have negative weights, this of course makes no sense: We can't have negative distances. It would also make a lot of what we are about to do just not work, as we will see in an exercise.

Definition 2. For a weighted graph G with positive edge-weights, we define the *graph distance* between two vertices $v, v' \in G$ by

$$d_G(v, v') = \min_{\text{paths } P \text{ from } v \text{ to } v'} \sum_{e \in E(P)} w(e).$$

Exercise 4. There's a reason we require positive edge-weights. What could happen if we allowed negative weights? Find an example of a weighted graph with some negative weights where the distance function isn't well-defined.

It turns out that this notion of graph distance does indeed turn a graph into a metric space, for those of you who know what that is.

Exercise 5. Convince yourselves that the following three properties of d_G hold:

1. For all $v, v' \in V$, we have $d_G(v, v') \geq 0$, and $d_G(v, v') = 0$ if and only if $v = v'$.
2. We have $d_G(v, v') = d_G(v', v)$ for all $v, v' \in V$.
3. The triangle inequality holds, that is, for all $v, u, w \in V$,

$$d_G(v, w) \leq d_G(v, u) + d_G(u, w).$$

Exercise 6. Can you come up with an algorithm to compute the distance between two vertices?

If you're feeling ambitious, it is actually possible to efficiently find the distances between a fixed vertex v_0 and every other vertex in the graph – how might you do this?