Algorithms SV worksheet 4

Bálint Molnár

September 28, 2025

1 Graph Traversal

Let $G = (V, E)$ be a connected graph. A tree $T = (V', E')$ is a spanning tree of G, if

- $V = V'$
- $E' \subseteq E$

Every graph traversal generates a spanning tree: it consists of the edges that have been used to explore the graph (i.e. if node v is visited from 'parent' node u, the edge (u, v) is part of the spanning tree).

1. Take the following 20-node graph: Nodes are $\{0, \dots 19\}$, and the edges are:

   ```
   [(0, 15), (0, 14), (0, 10), (0, 17), (0, 5), (0, 19), (1, 17),
   (1, 16), (1, 5), (1, 14), (1, 4), (2, 3), (2, 9), (3, 15),
   (3, 18), (3, 9), (3, 6), (4, 12), (5, 19), (5, 6), (6, 8),
   (6, 13), (6, 11), (7, 12), (7, 11), (7, 14), (8, 17), (8, 19),
   (8, 9), (9, 11), (9, 15), (9, 12), (9, 17), (10, 16), (11, 13),
   (11, 17), (12, 17), (14, 18), (15, 17), (18, 19)]
   ```

   Write a DFS algorithm that starts from node 0 and always visits its neighbours based on their numerical order (starting from the smallest).

   Calculate and visualise the algorithm's spanning tree on the graph above.

   Feel free to use AI tools, StackOverflow, etc. for the graph preprocessing and the visualisation, but please write the main code by yourself.

2. For the graph in the previous exercise, verify that for any edge (u, v), that is NOT in the spanning tree, either u is an ancestor of v, or v is an ancestor of u in the spanning tree. Prove that any DFS spanning tree has this property.

3. Prove that the height of some BFS spanning tree is minimal (among all spanning tree heights).

4. *(Difficult)* An edge e in a connected graph G is called a *bridge*, if removing it disconnects the graph. Design an algorithm that finds all the bridges in linear time.
   Hint: use the property of DFS spanning trees from Q2.

2 Shortest Paths

1. In a directed graph with edge weights, give a formal proof of the triangle inequality distance(u to v) $\leq$ distance(u to w) + cost($w \to$v) for all vertices u, v, w with $w \to v$. Make sure your proof covers the cases where no path exists.

2. Prove a more general variant: distance(u to v) $\leq$ distance(u to w) + distance (w to v).

3. Assume there exist values $D_{u,v}$ for any pair of nodes, u, v, which satisfy the following properties:

(a) $D_{u,u} \leq 0$ for all u

(b) $D_{u,v} \leq \text{cost}(u \to v)$ for all u, v.

(c) $D_{u,v} \leq D_{u,w} + D_{w,v}$ for all u, v, w

Prove that $D_{u,v} \leq \text{distance}(u \,\text{to}\, v)$ for all u, v

4. The notes from 2024 use a different Dijkstra's implementation. Please read the relevant part of the notes, and explain how the implementation differs from the one you saw on your lecture. What do you think, which one is faster in better?

5. Show that the old note's Dijkstra's algorithm eventually terminates with the correct result if the graph has negative-weight edges, but no negative cycles. What is its worst-case complexity for these graphs?

6. Briefly explain four different algorithms to solve the All-Pairs Shortest Path problem. Compare their time complexities, and explain the benefits of each approach.