Algorithms Example Sheet 4: Core Questions

BFS/DFS

Exercise 4.C.1 [Implementation aspects of DFS]

(a) Give pseudocode for a function `dfs_recurse_path(g, s, t)` based on `dfs_recurse`, that returns a path from s to t.

[Exercise 2 in Lecturer's handout]

(b) Modify your function from part (a) so that it does not visit any further vertices once it has reached the destination vertex t.

[Exercise 3 in Lecturer's handout]

(c) Do `dfs` and `dfs_recurse` (as given in lecture notes) always visit vertices in the same order? Either prove they do, or give an example of a graph where they do not. You may assume that there is an ordering on vertices, and that `v.neighbours` returns a sorted list of v's neighbouring vertices.

If they do not, then modify `dfs` so they do. Give pseudocode.

[Exercise 4 in Lecturer's handout]

Exercise 4.C.2 [Implementation aspects of BFS]

(a) Modify `bfs_path(g, s, t)` to find all shortest paths from s to t.

[Exercise 6 in Lecturer's handout]

(b) The breadth-first search algorithm from lecture notes uses $\mathcal{O}(1)$ storage within each vertex object (to store the seen flag), plus extra memory for `toexplore`. What is the worst case memory requirement of `toexplore`? Give your answer using Ω notation, in terms of V and E. Modify the algorithm to use $\mathcal{O}(1)$ storage within each vertex object, plus $\mathcal{O}(1)$ extra memory.

[Exercise 7 in Lecturer's handout]

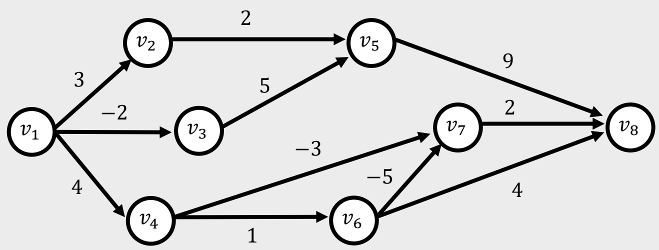
Directed Acyclic Graphs (DAGs)

Exercise 4.C.3

(a) Describe the topological sorting algorithm and argue why it works.

(b) What is its time complexity?

(c) Show its operation in the following DAG.



(d) Give a few examples of DAGs and the interpretation of the topological ordering (e.g. build systems, neural networks).

Exercise 4.C.4 Given a DAG with weights,

(a) Design an algorithm that finds the shortest path from s to t in time $\mathcal{O}(V+E)$.

(b) Design an algorithm that finds the longest path from s to t in time $\mathcal{O}(V+E)$.

(c) Design an algorithm that counts the number of paths from s to t in time $\mathcal{O}(V+E)$.

(d) (optional ++) Design an algorithm that finds the path of maximum average length (i.e. the sum of the weights in the path normalised by the number of edges in the path) from s to t in time $\mathcal{O}(V+E)$.

Exercise 4.C.5 Explain how to model a dynamic programming recurrence relation using a graph. Draw this graph for the Longest Common Subsequence (LCS) problem with $n = 5$ and $m = 3$.

Dijkstra's algorithm

Exercise 4.C.6 In a directed graph with edge weights, give a formal proof of the triangle inequality

$$d(u,v) \leq d(u,w) + c(w \to v) \text{ for all vertices } u, v, w \text{ with } w \to v$$

where $d(u,v)$ is the minimum weight of all paths from u to v (or ∞ if there are no such paths) and $c(w \to v)$ is the weight of edge $w \to v$. Make sure your proof covers the cases where no path exists.

[Exercise 8 in Lecturer's handout]

Exercise 4.C.7 [Proving shortest path properties] Read section 24.5 in CLRS and provide proofs for some of the following:

(a) Upper-bound property

(b) No-path property

(c) Convergence property

(d) Convergence property

(e) Path relaxation property

(f) Predecessor-subgraph property

Bellman-Ford algorithm

Exercise 4.C.8 In the course of running the Bellman-Ford algorithm, is the following assertion true? "Pick some vertex v, and consider the first time at which the algorithm reaches line 7 with `v.minweight` correct i.e. equal to the true minimum weight from the start vertex to v. After one subsequent pass of relaxing all the edges, `u.minweight` is correct for all $u \in$ neigbours(v)." If it is true, prove it. If not, provide a counterexample.

[Exercise 14 in Lecturer's handout]

Exercise 4.C.9 An engineer friend tells you there is a simpler way to reweight edges than the method used in Johnson's algorithm. Let w^* be the minimum weight of all edges in the graph, and just define $w'(u \to v) = w(u \to v) - w^*$ for all edges $u \to v$. What is wrong with your friend's idea?

[Exercise 25.3-4 in CLRS]

Exercise 4.C.10 [Floyd-Warshall algorithm] We are given a directed graph where each edge is labelled with a weight, and where the vertices are numbered $1, \dots, n$. Assume it contains no negative weight cycles. Define $F_{ij}(k)$ to be the minimum weight path from i to j, such that every intermediate vertex is in the set $\{1, \dots, k\}$. Give a dynamic programming equation for $F_{ij}(k)$, and a suitable definition for $F_{ij}(0)$.