Algorithms Example Sheet 4: Problems

Core graph theory

Exercise 4.P.1 [Trees]

(a) Define a *tree*.

(b) Prove that a tree with $V \geq 2$ has at least two leaves (i.e. a node with degree 1).

(c) Show that every tree with $V \geq 2$ vertices can be formed by a tree T' of $V-1$ vertices plus a vertex and an edge.

(d) Show that an undirected graph is a tree iff it is connected and $|E| = V - 1$.

(e) (Maybe only a few of these) Show that the following definitions are equivalent to the definition of a tree:

i. A tree is one component of a forest. (A forest is an acyclic graph.)

ii. A tree is a connected graph with at most $V-1$ edges.

iii. A tree is a minimally connected graph; removing any edge disconnects the graph.

iv. A tree is an acyclic graph with at least $V-1$ edges.

v. A tree is a maximally acyclic graph; adding an edge between any two vertices creates a cycle.

vi. A tree is a graph that contains a unique path between each pair of vertices.

(f) Is it true that any subgraph of a tree is a tree?

Exercise 4.P.2 Read the definitions of directed acyclic graph and tree from lecture notes. Draw an example and give its representation in adj. list format, of each of the following, or explain why no example exists:

(a) A directed acyclic graph with 8 vertices and 10 edges

(b) An undirected tree with 8 vertices and 10 edges

(c) A graph without cycles that is not a tree

Exercise 4.P.3 In an undirected graph, the degree $d(u)$ of a vertex u is the number of neighbours u. In a directed graph, we distinguish between the *indegree* $d_{in}(u)$, which is the number of edges incoming to u, and the *outdegree* $d_{out}(u)$, the number of edges outgoing from u.

(a) Show that in an undirected graph, $\sum_{u \in V} d(u) = 2|E|$.

(b) Use part **??** to show that in an undirected graph, there must be an even number of vertices whose degree is odd.

(c) Does a similar statement hold for the number of vertices with odd indegree in a directed graph?

Exercise 4.P.4 [Bipartite Graphs] An undirected graph $G = (V, E)$ is *bipartite* if the vertices V can be partitioned into two subsets L and R, such that every edge has one vertex in L and the other in R.

(a) Prove that every tree is a bipartite graph.

(b) Give an example of a graph that is non-bipartite.

(c) Design an efficient algorithm that determines whether a given undirected graph is bipartite. Argue that your algorithm works. What is the time complexity of your algorithm?

(d) Prove that a graph G is bipartite if and only if every cycle in G has an even number of edges.

Exercise 4.P.5 [Longest path in a tree]

(a) Consider a graph without edge weights, and write $d(u, v)$ for the length of the shortest path from u to v. The diameter of the graph is defined to be $\max_{u,v \in V} d(u, v)$. Give an efficient algorithm to compute the diameter of an undirected tree, and analyse its running time. *Hint:* One way to solve

this is by using breadth-first search, twice.

(b) Design a linear time algorithm for finding the diameter in a weighted tree. Argue that your algorithm is correct. *Hint:* Root the tree at some vertex r. For each vertex v, find the longest path in its subtree that ends at v. Why is it sufficient to consider only paths in its children?

BFS/DFS

Exercise 4.P.6 [Jar Problem] You are given 3 jars with capacities c_1, c_2 and c_3. Initially all jars are empty. In each step, you have the following options:

- Fill in the i-th jar.
- Empty the i-th jar.
- Move the contents of the i-th jar to that of the j-th jar.

Construct an algorithm that given the jar capacities and a target value t, finds the fewest number of moves to make the target capacity t. For example, if the capacities are $10, 7, 4$ and the target is 2, then the following moves reach the target: `(0, 0, 0) -> (10, 0, 0) -> (6, 0, 4) -> (6, 0, 0) -> (2, 0, 4)`.

Exercise 4.P.7 Consider a directed graph in which every vertex v is labelled with a real number x_v. For each vertex v, define m_v to be the minimum value of x_u among all vertices u such that either $u = v$ or u is reachable via some path from v. Give an $\mathcal{O}(E + V \log V)$-time algorithm that computes m_v for all vertices v.
Extra credit: After you have read the section on further topics in DFS, try to find an $\mathcal{O}(E + V)$ solution.

Exercise 4.P.8 [Connection to DFAs/NFAs]

(a) Given a DFA D and a string s explain how you would check if the string is accepted by the DFA (i.e. $s \in L(D)$). What is the time complexity of your algorithm?

(b) Given a DFA D, how can you check if it accepts any string (or equivalently if $L(D) = \emptyset$)? How can you check if $L(D) = \Sigma^*$?

(c) Given an NFA N and a string s explain how you would check if the string is accepted by the NFA. What is the time complexity of your algorithm?

(d) Given a DFA D, how can you efficiently construct a DFA D' that accepts the prefixes of the strings in $L(D)$? For example, if $L(D) = \{\text{abc}, yx\}$, then $L(D') = \{\text{a}, \text{ab}, \text{abc}, \text{y}, \text{yx}\}$.

Dijkstra's algorithm

Exercise 4.P.9 [Bounded edge capacities]

(a) Design an algorithm for finding the shortest path in a graph where the edges have weight 0 or 1. (Your algorithm should be more efficient than Dijkstra's)

(b) Design an algorithm for finding the shortest path in a graph where the edges have weight 1 or 2. (Your algorithm should be more efficient than Dijkstra's)

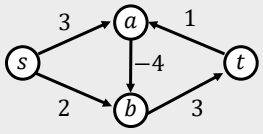
(c) Design an algorithm for finding the shortest path in a graph where all edges have integer weight in $[1, L]$. What is the time complexity of your algorithm?

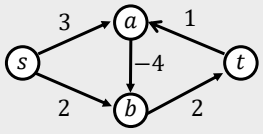
(d) (optional) Read about Dial's algorithm (or see problems 24.3.8 and 24.3.9 in CLRS).

Exercise 4.P.10 Suppose we have implemented Dijkstra's algorithm using a priority queue for which `push` and `decreaskey` have running time $\Theta(1)$, and `popmin` has running time $\Theta(\log n)$ where n is the number of items in the queue. Construct a sequence of graphs indexed by k, where the kth graph has $|V| = k$ and $|E| = \Theta(k^\alpha)$, such that Dijkstra's algorithm has running time $\Omega(k^\alpha + k \log k)$. Here α is a constant, $1 \leq \alpha \leq 2$.

Exercise 4.P.11 [Negative edges] *In the following exercises, run dijkstra as described in lectures, which loops until `toexplore` is empty. Some textbooks use different versions of Dijkstra's algorithm, that terminate once a destination vertex has been popped, or that don't allow popped vertices to re-enter `toexplore`.*

(a) By hand, run both Dijkstra's algorithm and the Bellman-Ford algorithm on each of the graphs below, starting from the vertex s. The labels indicate edge costs, and one is negative. Does Dijkstra's algorithm correctly compute minimum weights?





(b) Prove that Dijkstra's algorithm will produce the correct distance if it terminates.

(c) Prove that Dijkstra's algorithm will always terminate if the input graph has no negative-weight cycles.

(d) (+) Give a DAG, where Dijkstra's algorithm takes an exponential number of steps to compute the answer.

(e) If re-insertions are not allowed, provide an instance where the algorithm gives the wrong output. In this case could the algorithm loop forever?

(f) Given a directed graph with a single negative edge, give an efficient algorithm to find the shortest path from s to v assuming no negative cycles.

Exercise 4.P.12 [Shortest-paths modelling problems] Solve the following problem by converting the input to a suitable graph and finding the shortest path there:

(a) Given an unweighted directed graph, find the minimum number of edges that need to be reversed in order to find a path from s to t. **[CodeChef REVERSE]**

(b) Given a graph where each edge has a colour, find the path that minimises the number of colour changes. **[CodeChef YASPP]**

(c) Given a weighted graph G, find the shortest path from s to t given the option to zero out one of the edges.

(d) (Not quite modelling) Find the shortest path whose edge weights form a bitonic (increasing and then decreasing) sequence.

(e) Given a weighted graph G, an agent is placed at vertex s and their target is to reach t. Some of the vertices have a reward of y_v and the agent can get at most one of these. If the agent takes d time steps to reach the t and y_x is the reward they fetch (if any), their total reward is $d - y_x$. Design an algorithm to find the path that minimises this. **[Usaco DINING]**

Exercise 4.P.13 A contractor has written a program that she claims solves the shortest path problem, on directed graphs with edge weights ≥ 0. The program produces `v.distance` and `v.come_from` for every vertex v in a graph, reporting distances and paths from a given start vertex s. Give a $\mathcal{O}(V + E)$-time algorithm to test whether or not the output of the contractor's program is correct.
As a test of your algorithm, what will its output be for the graph above, when the contractor's program produces `s.come_from=None`, `a.come_from=b`, `b.come_from=a`, `s.distance=0`, `a.distance=1`,

`b.distance=1`?

[Exercise 24.3-4 in CLRS]

Bellman-Ford algorithm

Exercise 4.P.14

(a) Explain how the Bellman-Ford algorithm can detect if there is a negative-weight cycle in the graph.

(b) Modify the code given in the lectures to find a negative-weight cycle in the graph.

(c) (+) Let N be the number of currencies in the world. Let $e_{ij} \in \mathbb{R}$ be the exchange rate between currency i and currency j. If the exchange rate from US dollars and UK pounds is 1.05 and the exchange rate between UK pounds and US dollars is 0.98, then if you convert 100\$ you get 105£ and then if you convert back to US dollars you get 102.9\$ (so a profit of 2.9\$). You can generalise this to more than 2 currencies. Design an algorithm that given the table e determines if there is such an opportunity.

(d) (+) Design an algorithm that given a set of inequalities of the form $x_i \leq x_j + a_{ij}$, determines if there exists an assignment to x_is, such that all of these are satisfied.