Technische Universität München
Institut für Informatik
Theoretical Computer Science

Fundamental Algorithms 11

Exercise 1 (Hypergraphs)

A *hypergraph* extends the concept of a graph in the sense that edges are allowed to connect an arbitrary number of vertices (instead of exactly two). Hence, a *hypergraph* is defined as a tuple (V, H), where V is a set of vertices and H is a set of *hyperedges*, where $H \subset \mathcal{P}(H) \setminus \{\emptyset\}$, with $\mathcal{P}(H)$ the power set (i.e., the set of all possible subsets) of H.

1. Give a suitable definition of the concept of a *path* in a hypergraph.
2. Now, consider the hypergraph $S = (V_S, H_S)$ of "all" scientific articles, where V_S is the set of all authors, and each hyperedge $h \in H_S$ contains all authors of a specific scientific article. The Erdös number $\mathrm{Er}(a)$ of an author $a \in V_S$ is defined as the length of the shortest path in S that connects the specific vertex $e \in V$ (e corresponds to the author Paul Erdös) to a. Write down an algorithm to determine $\mathrm{Er}(a)$. *Hint:* Think about how graph algorithms presented in the lecture might help you here.
3. Try to formulate the above problem as a graph problem, i.e. given (V, H) construct some graph G such that the solution of a particular problem on G gives you the Erdös number.

Solution:

1. A path (of length n) exists between two vertices v and w in a hypergraph (V, H), if there exists a sequence of hyperedges $h_1, h_2, h_3, \ldots, h_n$ and a sequence of vertices $v_0, v_1, v_2, \ldots, v_n$ (where $v = v_0$ and $w = v_n$), such that $(v_{k-1}, v_k) \in h_k$ for all $1 \leq k \leq n$.
2. For a (non-directed, non-weighted) graph, breadth-first traversal will find the shortest path from a given root node to any reachable node in the graph. Hence, in ComputeErd we adapt breadth-first search for hypergraphs. Instead of using an array *mark* that just labels vertices as already visited, we use an array *erd* that contains the Erdös number of a already visited vertices. Non-visited vertices will be initialized to contain -1 in the beginning.

   `HyperNode` needs to be a suitable data structure to represent a node of a hypergraph. We assume that a HyperNode represent one vertex and stores the set of hyperedges that contain this vertex (or a reference to these hyperedges). Each hyperedge is a set of vertices.
3. For the Erdös number, it is insignificant which articles are actually responsible for the connection. Hence, we define a graph V_S, E_S, where two authors $a, b \in V_S$ are connected by an edge, i.e. $(a, b) \in E_S$, iff there exists a hyperedge H_S with $a \in H_S$ and $b \in H_S$. Then we can use the regular breadth-first traversal on graphs to determine the Erdös number.

   However, scientists who are interested in their Erdös number actually do want to know which articles define the connection, so we need to set up a different graph (V, E):
   - The set of nodes is defined as $V := V_S \cup H_S$, i.e. each author and each article define a vertex.
   - E contains an edge between an author $a \in V_S$ and an article $p \in H_S$, iff a was an author of p, i.e. $a \in p$.

Now, a shortest-path search in the resulting graph (using breadth-first search) will deliver the Erdös number and the connecting papers.

Algorithm 1: ComputeErd

```
Input: e: HyperNode, correspoding to Erdös
       x: HyperNode, corresponding to the Author
       n: Integer, number of nodes in the graph
Result: the Erdös number of x (or −1 if no path exists)
for i = 1 to n do erd[i] ← −1;
erd[e.key] ← 0;
act ← [e]; // Queue of active nodes
while act ≠ [] do
    v ← remove(act); // Get next queued node
    if v = x then return erd[v.key];
    co ← {}; // Set of co-authors
    for a ∈ v.edges do co ← co ∪ a;
    for a ∈ co do
        if erd[a.key] < 0 then
            erd[a.key] ← erd[v.key] + 1;
            act ← act ∘ a;
        end
    end
end
return −1 // No path found
```