Homework Assignment 9

Due Date: April 29, 2022, 23:59 (extended to May 6, 2022, 23:59)

EXERCISE 1.

(i) Describe how to implement an algorithm to determine a maximum spanning tree, i.e. the sum of the edge costs shall be maximal.

(ii) What happens in the case of Prim's and Kruskal's algorithms, if negative edge costs are permitted? Is it still sensible to talk about minimum spanning trees, if negative edge costs are permitted?

total points: 12

EXERCISE 2.

(i) Kruskal's algorithm also works for graphs that are not connected, in which case the result is a *spanning forest*. Modify Prim's algorithm to work as well for non-connected graphs without referring to multiple calls of the algorithm.

Hint. Use $n - 1$ additional edges.

(ii) Show that if all edge costs are pairwise different, the minimum spanning tree is uniquely defined.

(iii) A set T of edges spans a connected graph G, if (V, T) is connected. Is a minimum cost spanning set of edges necessarily a tree? Is it a tree if all edge costs are positive?

total points: 12

EXERCISE 3. Implement a class DIGRAPH of directed graphs:

(i) Define basic operations for the insertion and deletion of edges and the determination of outgoing/incoming edges.

(ii) Implement depth-first and breadth-first search on directed graphs.

(iii) Implement a program that checks, if a given directed graph is acyclic.

Hint. Modify the classes GRAPH and GRAPHTRAVERSAL that are subject of Laboratory 10.

total points: 20

Exercise 4. For the bipartite matching problem we are given a finite bipartite graph (V, E), where the set V of vertices is partitioned into two sets *Boys* and *Girls* of equal size. Thus, the set E of edges contains sets $\{x, y\}$ with $x \in \mathit{Boys}$ and $y \in \mathit{Girls}$. A *perfect matching* is a subset $F \subseteq E$ such that every vertex is incident to exactly one edge in F. A *partial matching* is a subset $F \subseteq E$ such that every vertex is incident to at most one edge in F. So the algorithm will create larger and larger partial matchings until no more unmatched boys and girls are left, otherwise no perfect matching exists.

We use functions girls_to_boys and boys_to_girls turning sets of unordered edges into sets of ordered pairs:

$$\begin{aligned}\text{girls\_to\_boys}(X) &= \{(g, b) \mid b \in \mathit{Boys} \wedge g \in \mathit{Girls} : \{b, g\} \in X\}\\ \text{boys\_to\_girls}(X) &= \{(b, g) \mid b \in \mathit{Boys} \wedge g \in \mathit{Girls} : \{b, g\} \in X\}\end{aligned}$$

Conversely, the function unordered turns a set of ordered pairs (b, g) or (g, b) into a set of two-element sets:

$$\text{unordered}(X) = \{\{x, y\} \mid (x, y) \in X\}$$

We further use a predicate reachable and a function path. For the former one we have reachable(b, X, g) iff there is a path from b to g using the directed edges in X. For the latter one path(b, X, g) is a set of ordered pairs representing a path from b to g using the directed edges in X.

Then an algorithm for bipartite matching can be realised by iterating the following rule:

```
par if     mode = init
    then   par   mode := examine
                 partial_match := ∅
           endpar
    endif
    if     mode = examine
    then   if    ∃b ∈ Boys.∀g ∈ Girls.{b, g} ∉ partial_match
           then  mode := build-digraph
           else  par   Output := true
                       Halt := true
                       mode := final
                 endpar
           endif
    endif
    if     mode = build-digraph
    then   par   di_graph := girls_to_boys(partial_match)
                             ∪ boys_to_girls(E − partial_match)
                 mode := build-path
           endpar
    endif
    if     mode = build-path
    then choose b ∈ {x | x ∈ Boys : ∀g ∈ Girls.{b, g} ∉ partial_match}
           do    if ∃g′ ∈ Girls.∀b′ ∈ Boys.{b′, g′} ∉ partial_match
                             ∧ reachable(b, di_graph, g′)
```

then choose $g \in \{y \mid y \in \mathit{Girls}.\forall x \in \mathit{Boys}.\{x, y\}$
$\notin \mathit{partial\_match} \wedge \text{ reachable}(b, \mathit{di\_graph}, y)\}$
do **par** $\mathit{path} := \text{path}(b, \mathit{di\_graph}, g)$
$\mathit{mode} := \text{modify}$
endpar
enddo
else **par** $\mathit{Output} := \textbf{false}$
$\mathit{Halt} := \textbf{true}$
$\mathit{mode} := \text{final}$
endpar
endif
enddo
endif
if $\mathit{mode} = \text{modify}$
then **par** $\mathit{partial\_match} = (\mathit{partial\_match} - \text{unordered}(\mathit{path}))$
$\cup(\text{unordered}(\mathit{path}) - \mathit{partial\_match})$
$\mathit{mode} := \text{examine}$
endpar
endif
endpar

(i) Implement a class BiPartiteGraph of bipartite graphs covering basic operations for insertion and deletion of vertices and edges and for the determination of edges incident to a given vertex.

(ii) Using the class BiPartiteGraph from (i) implement the above algorithm for the determination of a perfect matching on a bipartite graph (provided such a matching exists).

Hint. Modify the classes Graph from Laboratory 10 and DiGraph from (i).

total points: 20