# Assignment 8 – Selected Model Answers

EXERCISE 1.

**(i)** Explain how to find the minimum and the maximum key stored in a B-tree.

**(ii)** Implement operations on B-trees to return the record associated with the minimum and maximum keys, respectively.

**(iii)** Explain how to find the predecessor and the successor keys of a given key stored in a B-tree.

**(iv)** Implement operations on B-trees to return the record associated with the predecessor and the successor of a given key.

EXERCISE 2.

**(i)** Explain how to find the minimum and the maximum key stored in a B$^+$-tree.

**(ii)** Implement operations on B$^+$-trees to return the record associated with the minimum and maximum keys, respectively.

**(iii)** Explain how to find the predecessor and the successor keys of a given key stored in a B$^+$-tree.

**(iv)** Implement operations on B$^+$-trees to return the record associated with the predecessor and the successor of a given key.

EXERCISE 3. 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:

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

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 ∈ {y | y ∈ Girls.∀x ∈ Boys.{x, y}
                          ∉ partial_match ∧ reachable(b, di_graph, y)}
                            do  par path := path(b, di_graph, g)
                                    mode := modify
                                endpar
                            enddo
                    else    par Output := false
                                Halt := true
                                mode := final
                            endpar
                    endif
            enddo
    endif
    if      mode = modify
    then    par     partial_match = (partial_match − unordered(path))
                                    ∪(unordered(path) − partial_match)
                    mode := examine
            endpar
    endif
endpar
```

**(i)** Implement the above algorithm for the determination of a perfect matching on a bipartite graph (provided such a matching exists).

EXERCISE 4.

**(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)** Implement a program that determines, whether a perfect matching exists for a given bipartite graph (not the algorithm from the previous exercise).