31-1-3 Řazení kořenek

Složka *Minimal Sort* obsahuje řešení problému v Javě.

Tato úloha se zabývá netradičním řadícím algoritmem, kde se soustředí ne na rychlost, ale na počet posunutí v řetězci. Kvůli tomuto omezení, nemůžeme použít obvyklé postupy jako Quicksort nebo Mergesort.

Při řazení takovým to způsobem, musíme maximalizovat efektivitu jednoho posunutí. Pokud chceme posouvat co nejmenší počet *lahviček*, neměli bychom posouvat ty, co už jsou na správném místě. Z celého řetězce tak musíme vybrat nejdelší vzestupnou řadu čísel (každá *lahvička* má přiřazené svoje místo, vyjádřeno číslem; dvě lahvičky nemůžou být na stejném místě). Nejjednodušší způsob je použít LIS (Longest increasing subsequence), buď pouze pomocí Dynamic Programming ($O(n^2)$), nebo přidat Binary search a snížit tak *O* na *n log n*.

Výsledný počet kroků potom bude *délka původního řetězce – LIS řetězec.* Nyní již stačí pouze porovnat původní a LIS řetězec a přesunout neseřazený prvek tak, aby byl na správném místě vůči seřazenému řetězci a označíme ho za seřazený. Takto postupujeme, dokud řetězec nebude seřazený.

Příklad takového řešení:

- Je dán řetězec: [9, 8, 1, 4, 2, 6, 3, 7, 5, 0] (10 prvků)
- Nejdelší seřazený řetězec: [1, 2, 3, 5] (4 prvky)
    - Potřebný počet kroků k seřazení řetězce: 10–4 = 6

1. Prvek 9 posuneme za prvek 5
2. Prvek 8 posuneme za prvek 5
3. Prvek 4 posuneme za prvek 3
4. Prvek 6 posuneme za prvek 5
5. Prvek 7 posuneme za prvek 6
6. Prvek 0 posuneme na začátek

- Výsledný řetězec: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

Řešení v Javě

Za řešení úlohy zodpovídá třída Sort, která do konzole vypíše postup, jakým by měl být řetězec seřazen. První krok spočívá v spočítání LIS. Tento krok proběhne v metodě *calculateLongestSubsequence().* Pomocí cyklu *for* projede zadaný řetězec a zjistí jeho délku a poslední číslo v tomto řetězci. Metoda *makeLis()* následně LIS sestaví. Samotný návod se pak vygeneruje v metodě *howToSortArray*, kde se array přeskládá do správného pořadí a jednotlivé kroky vypíše.

```java
package minimal.sort;

import java.util.ArrayList;
import java.util.Collections;

/**
 *
 * @author Daniil
 */
public class StartUp {
    static ArrayList<Integer> array;
    public static void main(String[] args) {
        // TODO code application logic here
        fillArray(10);
        System.out.println("Původní array: " + array);

        Sort s = new Sort();

        array = s.sortArray(array);
        System.out.println("Seřazený array: " + array);
        System.out.println("Bylo potřeba" + s.getSteps() + " přesunutí.");

    }

    public static void fillArray(int n) {
        array = new ArrayList();

        for (int i = 0; i < n; i++) {
            array.add(i);
        }
        Collections.shuffle(array);
    }
}

class Sort {
    private int steps = 0;      // Počet kroků potřebných k seřazení řetězce
    ArrayList<Integer> array;   // Řetězec k seřazení
    ArrayList<Integer> lis;     // Pomocný řetězec

    public ArrayList sortArray(ArrayList<Integer> array) {
        this.array = array;
        calculateLongestSubsequence();
        howToSortArray();
        return this.array;
    }

    public int getSteps() {
        return steps;
    }

    private void howToSortArray() {
        if (steps > 0) {
            for (int i = 0; i < array.size(); i++) {
                int fromArray = array.get(i);

                if (lis.contains(fromArray)) {
                    continue;
                }
                array.remove(i);
                for (int j = array.size() - 1; j >= 0; j--) {
                    if (!lis.contains(array.get(j))) {
                        continue;
                    } else if (fromArray > array.get(j)) {
                        if (j <= array.size() - 1) {
                            array.add(j + 1, fromArray);
                            System.out.println("Prvek " + fromArray +
                                    " byl posunut za  " + array.get(j));
                        } else {
                            array.add(fromArray);
                            System.out.println(fromArray + " přidnán na konec");
                        }
                        lis.add(fromArray);
                        break;
                    } else if (j == 0) {
                        array.add(0, fromArray);
                        lis.add(fromArray);
                        System.out.println(fromArray + " přidnán na zaÄŤÄ˝tek.");
                    }
                }
                i--;
            }
        } else {
            System.out.println("Array již seřazen.");
        }
    }

    // Najde délku nejdelšího seřazeného řetězce jeho poslední prvek
    private void calculateLongestSubsequence() {
        ArrayList<Integer> tail = new ArrayList();

        for (int i = 0; i < array.size(); i++) {
            tail.add(0);
        }
        int lenght = 1;
        tail.set(0, array.get(0));

        for (int i = 1; i < array.size(); i++) {
            if (array.get(i) < tail.get(0)) {
                tail.set(0, array.get(i));
            } else if (array.get(i) > tail.get(lenght - 1)) {
                tail.set(lenght++, array.get(i));

            } else {
                tail.set(ceilIndex(tail, -1, lenght - 1, array.get(i)), array.get(i));

            }
        }
        makeLis(tail, lenght);
        System.out.println("Nejdelší LIS: " + lenght);
        System.out.println("LIS: " + lis);
        steps = array.size() - lenght;
    }

    private int ceilIndex(ArrayList<Integer> list, int l, int r, int key) {
        while (r - l > 1) {
            int m = l + (r - l) / 2;
            if (list.get(m) >= key) {
                r = m;
            } else {
                l = m;
            }
        }
        return r;
    }

    // Vytvoří array pouze s seřazenými prvky
    private void makeLis(ArrayList<Integer> a, int s) {
        lis = new ArrayList();
        lis.add(a.get(s - 1));

        for (int i = array.indexOf(a.get(s - 1)); i >= 0; i--) {
            if (array.get(i) < lis.get(lis.size() - 1)) {
                lis.add(array.get(i));
            } else if (lis.size() >= 2) {
                if (array.get(i) < lis.get(lis.size() - 2)) {
                    lis.set(lis.size() - 1, array.get(i));

                }
            }
        }

        for (int i = 0; i < lis.size() / 2; i++) {
            int temp = lis.get(i);
            lis.set(i, lis.get(lis.size() - i - 1));
            lis.set(lis.size() - i - 1, temp);
        }
    }
}
```