1 Lista 5, Zadanie 3

Mamy zaproponować strukturę Q dla dynamicznych zbiorów liczb, na kórych będzie można wykonac operacje $Min-Luka$ oraz, na której będzie można elegancko zaimplementować operacje $Insert, Delete, Search$.

1.1 Pomysł główny

Naturalnym kandydatem do zadania, są oczywiście, RB-drzewa, które jeszcze odrobine wzmocnimy. Standardową wartością w węźlie będzie liczba, oprócz tego dodamy jeszcze 3 pola:

- x.min-wart - minimalna wartość poddrzewa zakorzenionego w x.
- x.max-wart - maksymalna wartość poddrzewa zakorzenionego w x.
- x.min-luka - minimalna różnica wartości poddrzewa zakorzenionego w x. Gdy x jest liściem x.min-luka $= \infty$.

Po co są min-wart i max-wart? Przydadzą się przy obliczaniu min-luka po operacjach przetwarzających drzewo.
Operacje $Insert, Delete$ na RB-drzewach nie są banalne. Nie wiem czy jest cel przepisywać delikatnie zmienione procedury z podręcznika Cormen'a, $Insert$ z fixupem mają razem 34 linijki, a dodanie powyższych pól niedużo zmienia w samym algorytmie.
Do procedur z podręcznika wystarczy dodać implementacje poniższego zachowania.

$$\text{x.min-wart}= \begin{cases} \text{x.left.min-wart} & \text{gdy x.left != nil,} \\ \text{x.wart} & \text{wpw} \end{cases}$$

$$\text{x.max-wart}= \begin{cases} \text{x.right.max-wart} & \text{gdy x.right != nil,} \\ \text{x.wart} & \text{wpw} \end{cases}$$

Teraz kolos:

$$\text{x.min-luka} = \min \begin{cases} \text{x.right.min-luka} & \text{gdy x.right} == \text{nil to x.right.min-luka} = \infty, \\ \text{x.left.min-luka} & \text{gdy x.left} == \text{nil to x.left.min-luka} = \infty, \\ \text{x.right.min-wart - x.wart} & \text{gdy x.right} == \text{nil to x.right.min-wart - x.wart} = \infty, \\ \text{x.wart - x.left.max-wart} & \text{gdy x.left} == \text{nil to x.wart - x.left.max-wart} = \infty, \end{cases}$$

Pozostaje jeszcze $Search$, który jest zwykłym algorytmem z RB-drzew, tu nic się nie zmienia.