README

Nume: Gogea

Prenume: Mihail

Grupa: 324CC

Tema 1 Paradigme de programare (Racket) Arbori binari

Dificultate: 8.5/10

Timpul alocat aceste teme: 14-18 ore

Observatii:

Pentru rezolvarea temei documentatia a fost luata din laborator, curs, wikipedia si youtube.

Unele explicatii ale unor functii sunt comentarii desupra functilor in fisierul .rkt .

Pentru majoritatea functilor am definit eu proprile functii iar apoi le-am apelat in functile antet din scheletul temei.

Modul de implementare:

Task 0:

Structura nodului este urmatoarea : '(valoare left right) unde valoarea este radacina nodului iar left este nodul stang iar right este nodul drept care aceastea la randul lor pot fi noduri sau frunze.

Pentru alegerea minimului sau maximului am definit o functie minmax in care unul dintre argumente este tipul pe care vreau sa il parcurg (stanga sau dreapta) pentru a nu scrie aceasta functie de 2 ori pentru minim si maxim.

Pentru inorder am definit functia inordine unde cu ajutorul functiei append am apelat recursiv stanga valoare-nod dreapta , urmand ca rezultatul sa fie o lista cu elementele arborelui inordine.

La succesor si predecesor am luat cele 2 cazuri : primul caz in care are fiul necesar pentru a lua valoarea , iar apoi daca nu am acel fiu a trebuit sa caut in sus spre radacina valoarea cautata , pentru a face asta am luat un acumulator unde retineam valoarea cautata ,iar apoi updatata la valoarea finala.

Task 1:

Inserarea am facuto astfel : am cautat locul unde trebuie sa inserez noul nod , iar pe parcursul acestei cautari nu am schimbat nimic lasand totul la fel cum era inainte, iar apoi cand am gasit locul unde trebuie sa inserez noul nod am introdus nodul iar apoi am facut recursiv balance.

La functia balance am folosit cele 4 cazuri:leftleft, leftright, rightright, rightleft ale balansului unui arbore avl.

La stergere am cautat nodul si in functie de tipul lui am executat comenzile necesare: daca era frunza pur si simplu parintele lui nu mai avea acel fiu , daca avea doar un fiu am introdus ca fiu al parintelui lui fiul nodului pe care il voi sterge, iar apoi daca avea ambi fii am inlocuit radacina cu dreapta maxim stanga si apoi urmand sa il sterg pe acela.

La union am introdus in tree1 toate elementele din tree2 si l-am returnat pe tree1. La intersectie si complement am creeat un nou arbore cu acele caracteristici ale intersectiei , respectiv complement a 2 arbori.

Task2:

La acest task mentionez ca am folosit inorder asa cum a fost spus si pe forumul temei (cs.curs.pub.ro) ca avem voie sa transformam liste ,iar functia de permutari a fost luata si putin modificata pentru zig-zag din exemplele de exercitii facute la curs (mentionez ca am intrebat personal daca avem voie sa preluam functii de la curs pe domnul profesor, raspunsul fiind afirmativ).

a) submultimile

Pentru a rezolva submultimile, am creat o functie care returneaza o lista cu toate submultimile ale argumentului , iar apoi am selectat cu ajutorul unui filtru acele liste care sunt de lungime k.

b) submultimile zig-zag

Aici am facut o functie de permutari , unde am permutat lista initiala apoi am introdus un filtru ce avea 2 cazuri unul in care primul element este mai mare care al doilea , respectiv celalalt caz in care primul element era mai mic ca urmatorul element.

Bonus:

Prima functie construieste arborele sintactic respectand forma arborelui din poza din enuntul temei , introducand intr-un nod : radacina e mereu simbol iar fii poate fi ori simbol ori numar.

A doua functie evalueaza arborele creat cu prima functie.