1 Tema Nr. 6: Arbori Multicăi

Transformări între diferite reprezentări
Timp alocat: 2 ore

1.1 Implementare

1. Se cere implementarea **corectă** și **eficientă** a traversării *iterative* (se poate găsi în Curs 6) și *recursive* a unui arbore binar. Puteți găsi orice informații necesare și pseudocod în notele de curs și seminar.
2. În plus, se cere implementarea **corectă** și **eficientă** a unor algoritmi de complexitate *liniară* pentru transformarea arborilor multicăi între următoarele reprezentări:
    1. **R1**: *reprezentarea părinte*: pentru fiecare index, valoare din vector reprezintă indexul părintele, ex: $\Pi = \{2, 7, 5, 2, 7, 7, -1, 5, 2\}$
    2. **R2**: *reprezentare arbore multicăi*: fiecare nod conține cheia si un vector de noduri copil
    3. **R3**: *reprezentare binara:* fiecare nod conține cheia si doi pointeri: unul către primul copil si al doilea către fratele din dreapta (ex: următorul frate).

    Așadar, trebuie să definiți transformarea **T1** din *reprezentarea părinte* (**R1**) în *reprezentarea arbore multicăi* (**R2**), iar apoi transformarea **T2** din *reprezentarea arbore multicăi* (**R2**) în *reprezentarea binară* (**R3**). Pentru toate reprezentările (**R1**, **R2**, **R3**) trebuie să implementați afișarea prietenoasă (pretty print, **PP**) (vezi pagina 3).

    Definiți structurile de date. Puteți folosi structuri intermediare (ex: memorie adițională).

1.2 Cerințe minimale pentru notare

Lipsa oricărei cerințe minimale (chiar și parțială) poate rezulta într-o notă mai mică prin penalizări sau refuzul de a prelua tema, rezultând în nota 0.

- *Demo*: Pregătiți un exemplu pentru exemplificarea corectitudinii fiecărui algoritm implementat. Corectitudinea fiecărui algoritm se demonstrează printr-un exemplu simplu (maxim 10 valori).
- Notați observațiile personale în antetul fișierului *main.cpp*, într-un comentariu bloc informativ.
- Nu preluăm teme care nu sunt indentate și care nu sunt organizate în funcții (de exemplu, nu prelăum teme unde tot codul este pus în main).
- *Punctajele din barem sunt corespondente unei rezolvări corecte și complete a cerinței, calitatea interpretărilor din comentariul bloc și* ***răspunsul corect dat de dumeavostră la întrebările puse de către profesor.***

1.3 Cerințe

1.3.1 Implementare a parcurgerii *iterative* (cu *memorie adițională constantă*) și *recursive* a unui arbore binar în *O(n)* (3p)

Demo: Corectitudinea algoritmilor va trebui exemplificată pe date de intrare de dimensiuni mici.

Este necesară o singură ordine de parcurgere a arborelui — Preorder, Inorder sau Postorder — și puteți alege după preferințe.

Puteți genera un vectot sortat care poate fi transformat într-un arbore binar printr-o abordare recursivă.

1.3.2 Transformări între reprezentările arborilor

1. **Implementare corectă pentru Pretty-print pentru R1 (2p)**

   Demo: Corectitudinea algoritmilor trebuie demonstrată folosind exemplul $\Pi = \{2, 7, 5, 2, 7, 7, -1, 5, 2\}$ cu un pretty-print, așa cum este prezentat în figura de mai jos.

2. **Implementare corectă pentru *T1* (de la R1 la R2) și pretty-print pentru *R2* (1p) + *T1* în timp liniar (1p)**

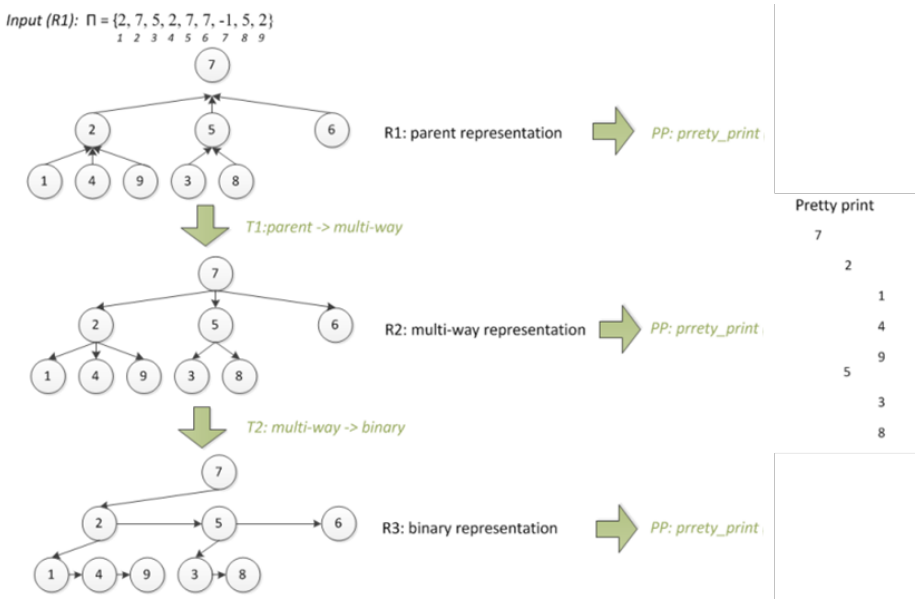
   Demo: Corectitudinea algoritmilor trebuie demonstrată folosind exemplul $\Pi = \{2, 7, 5, 2, 7, 7, -1, 5, 2\}$ cu un pretty-print, așa cum este prezentat în figura de mai jos.

3. **Implementare corectă pentru *T2* (de la R2 la R3) și pretty-print pentru *R3* (2p) + *T2* în timp liniar (1p)**

   Demo: Corectitudinea algoritmilor trebuie demonstrată folosind exemplul $\Pi = \{2, 7, 5, 2, 7, 7, -1, 5, 2\}$ cu un pretty-print, așa cum este prezentat în figura de mai jos.

Folosiți afișarea prietenoasă pentru cele trei reprezentări. *Fiecare reprezentare (R1,R2,R3) necesită o afișare prietenoasă cu o implementare diferită dar același rezultat.*

Analizați eficienta în timp și spațiu a celor două transformări. Ați atins *O(n)*? Ați folosit memorie adițională?


Input (R1): Π = {2, 7, 5, 2, 7, 7, -1, 5, 2}
1 2 3 4 5 6 7 8 9
R1: parent representation
PP: prrety_print
T1:parent -> multi-way
R2: multi-way representation
PP: prrety_print
T2: multi-way -> binary
R3: binary representation
PP: prrety_print
Pretty print
7
2
1
4
9
5
3
8