

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* 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 părinte, ex: $\Pi = \{2, 7, 5, 2, 7, 7, -1, 5, 2\}$
 2. **R2**: *reprezentare arbore multicăi*: fiecare nod conține cheia și un vector de noduri copil
 3. **R3**: *reprezentare binară*: fiecare nod conține cheia și doi pointeri: unul către primul copil și 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 2).

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).
- Graficele create trebuie să fie ușor de evaluat, adică grupate și adunate prin funcțiile Profiler după cerințele temei. Tema nu va fi evaluată dacă conține o multitudine de grafice negrupate. De exemplu, analiza comparativă implică gruparea într-un singur grafic a algoritmilor comparați.
- Interpretați graficul/graficele și 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 dumeavoastră la întrebările puse de către profesor.***

1.3 Cerințe

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

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

1.3.2 Implementarea corectă la pretty-print la *R1* (2p)

Demo: Corectitudinea algoritmilor va trebui demonstrată pe exemplul $\Pi = \{2, 7, 5, 2, 7, 7, -1, 5, 2\}$.

1.3.3 Implementarea corectă la *T1* (din *R1* în *R2*) și pretty-print la *R2* (1p) + *T1* în timp liniar (1p)

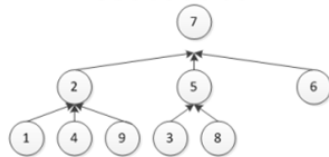
Demo: Corectitudinea algoritmilor va trebui demonstrată pe exemplul $\Pi = \{2, 7, 5, 2, 7, 7, -1, 5, 2\}$.

1.3.4 Implementarea corectă la *T2* (din *R2* în *R3*) și pretty-print la *R3* (2p) + *T2* în timp liniar (1p)

Demo: Corectitudinea algoritmilor va trebui demonstrată pe exemplul $\Pi = \{2, 7, 5, 2, 7, 7, -1, 5, 2\}$.

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

Input (R1): $\Pi = \{2, 7, 5, 2, 7, 7, -1, 5, 2\}$
 1 2 3 4 5 6 7 8 9

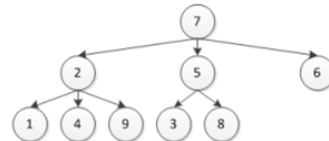


R1: parent representation



PP: pretty_print

T1: parent -> multi-way

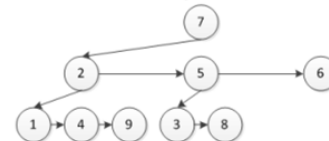


R2: multi-way representation



PP: pretty_print

T2: multi-way -> binary



R3: binary representation



PP: pretty_print

Pretty print

7

2

1

4

9

3

8

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