Tema Nr. 8: Parcurgere iterativă vs recursivă arbore binar. Hibridizare quicksort. Analiza comparativă a timpului de execuție.

Timp alocat: 2 ore

Implementare

Se cere implementarea **corectă** și **eficientă** a *parcurgerii iterative si recursive* al unui arbore binar, respectiv *hibridizare pentru quicksort*

Informații utile și pseudocod găsiți în notițele de la curs si seminar:

- *Parcurgere recursiva si iterativa a unui arbore binar in O(n)*
- *Hibridizare quicksort utilizând insertion sort iterativ - in quicksort, pentru dimensiuni de șir < prag, se utilizeaza insertion sort (folosiți implementare quicksort din tema 3 și insertion sort din tema 1).*

Notare și cerințe

- Implementare parcurgere iterativă și recursivă a unui arbore binar în O(n) și cu memorie aditionala constanta + demo – 5p
- Analiza comparativă parcurgere iterativă vs recursivă din perspectiva numărului de operații – 2p
- Hibridizare quicksort. Demo și analiză comparativă a numărului de operații și a timpului efectiv de execuție – 2p
- Determinarea unui optim al pragului folosit în hibridizare + motivatie (grafice/masuratori) – 1p

Evaluare

! Înainte de a începe să lucrați pe partea de evaluare, asigurați-vă că aveți un **algoritm corect**! Corectitudinea algoritmilor va trebui exemplificată pe date de intrare de dimensiuni mici.

1. Pentru analiza comparativa a versiunii iterative vs recursive, trebuie sa numărați doar operațiile de afișare a cheii variind numărul de noduri din arbore de la 100 la 10000 cu un prag de 100.

   Pentru partea de construcție al arborelui, puteti sa porniti de la un tablou a cărui dimensiune variază și să alegeți în mod aleatoriu rădăcina.

2. Pentru hibridizare quicksort trebuie să utilizați versiunea iterativă de insertion sort din prima temă în cazul în care dimensiunea vectorului este mică (sugeram utiliarea insection sort daca vectorul are sub 30 de elemente). Comparați timpul de rulare și numărul de operații (asignari + comparatii) pentru quicksort implementat în tema 3 cu cel hibridizat.

   Determinarea optimului din perspectiva pragului utilizat se realizează prin varierea dimensiunii vectorului pe care se aplică insertion sort și compararea rezultatelor obținute din perspectiva performanței. Pentru comparatie folosiți ca și dimensiune a vectorului ce urmează sa fie sortat cu quicksort hibrid 10.000.

   Pentru a măsura timpul de execuție puteți folosi Profiler similar cu exemplul de mai jos.

```
profiler.startTimer("your_function", current_size);
for(int test=0; test<nr_tests; ++test) {
   your_function(array, current_size);
}
profiler.stopTimer("your_function", current_size);
```

   Numărul de teste care trebuie sa fie repetate (*nr_tests* din exemplul de mai sus) trebuie ales în funcție de procesor și modul de compilare. Sugeram valori mai mari, precum 100 sau 1000.

   În momentul în care măsurați timpul de execuție, asigurați-vă că opriți orice alte procese care nu sunt necesare.

3. Pregătiți un exemplu pentru exemplificarea corectitudinii algoritmilor implementați.

4. Nu preluăm teme care nu sunt indentate și care nu sunt organizate în funcții (de exemplu nu preluam teme unde tot codul este pus in main).

5. ***Punctajele din barem se dau pentru rezolvarea corectă și completă a cerinței, calitatea interpretărilor din comentariul bloc și răspunsul corect dat de voi la întrebările puse de către profesor.***