Tema Nr. 3: Analiza și compararea metodelor avansate de sortare – HeapSort și QuickSort / QuickSelect

Timp alocat: 2 ore

Implementare

Se cere implementarea **corectă** și **eficientă** a sortării rapidă (Quicksort) și Quick-Select (Randomized-Select). Se cere și analizarea comparative a complexității *sortării folosind heap-uri (Heapsort, implmentat în tema 2)* și *sortarea rapidă (Quicksort)*.

Informații utile și pseudo-cod găsiți în notițele de curs sau în carte[1]:

- *Heapsort*: capitolul 6 (Heapsort)
- *Quicksort*: capitolul 7 (Quicksort)
- *QuickSelect/Randomized Select:* capitolul 9

Cerințe minimale pentru notare

- Pregătiți un exemplu pentru exemplificarea corectitudinii metodelor de sortare implementate.
- Interpretați graficul și notați observațiile personale în antetul fișierului *.cpp*, într-un comentariu bloc informativ.
- 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).
- ***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.***

Cerințe

1. **QuickSort: implementare (3p)**

Corectitudinea algoritmului va trebui exemplificată pe date de intrare de dimensiuni mici.

2. **QuickSort: evaluare în caz mediu statistic (2p)**

! Înainte de a începe să lucrați pe partea de evaluare, asigurați-vă că aveți un **algoritm corect**!

Se cere compararea celor două metode de sortare în cazul **mediu statistic**. Pentru cazul **mediu statistic** va trebui să repetați măsurătorile de *m* ori (m=5) și să raportați valoarea lor medie; de asemenea, pentru cazul **mediu statistic**, asigurați-vă că folosiți **aceleași** date de intrare pentru cele două metode.

Pașii de analiză:
- variați dimensiunea șirului de intrare (*n*) între [100…10000], cu un increment de maxim 500 (sugerăm 100).
- pentru fiecare dimensiune (*n*), generați date de intrare adecvate metodei de construcție; rulați metoda numărând operațiile elementare (atribuiri și comparații - pot fi numărate împreună pentru această temă).

! Doar atribuirile și comparație care se fac pe datele de intrare și pe datele auxiliare corespunzătoare se iau în considerare.

Generați un grafic ce compară cele două metode de construcție în cazul **mediu statistic** pe baza numărului de operații obținut la pasul anterior.

Dacă o curba nu poate fi vizualizată corect din cauza că celelalte curbe au o rată mai mare de creștere, atunci plasați noua curbă pe un alt grafic. Denumiți adecvat graficele și curbele.

3. **QuickSort: evaluare în caz favorabil și defavorabil (1p)**

4. **QuickSort și HeapSort: analiză comparativă a cazului mediu statistic, interpretare (2p)**

5. **Analiza comparativă a *unuia* din algoritmii de sortare din L1 (la alegere) în versiune iterativă vs recursivă. Analiza se va efectua atât din *perspectiva numărului de operații, cât și a timpului de rulare* (2p)**

Pentru analiza comparativă a versiunii iterative vs recursive, alegeți oricare din cei 3 algoritmi din laboratorul 1 (bubble sort, insertion sau selection). Folosiți varianta iterativă pe care ați implementat-o și predat-o în cadrul laboratorului (corectată, dacă este nevoie, în funcție de feedback-ul pe care l-ati primit) și implementați același algoritm de sortare în mod recursiv.

Trebuie să măsurați atât efortul total, cât și timpul de rulare al celor două versiuni (iterativ și recursiv)=> două grafice, fiecare comparand cele două versiuni de algoritm.

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 (*nr_tests* din exemplul de mai sus) trebuie ales în funcție de procesor și modul de compilare. Sugerăm valori mai mari, precum 100 sau 1000.

6. **Bonus: QuickSelect - Randomized-Select (0.5p)**

Corectitudinea algoritmului va trebui exemplificată pe date de intrare de dimensiuni mici.

Pentru QuickSelect (Randomized-Select) nu trebuie facuta analiza complexității, doar corectitudinea trebuie exemplificată.