1 Tema nr. 1: Analiza și Compararea Metodelor Directe de Sortare

Timp alocat: 2 ore

1.1 Implementare

Se cere implementarea **corectă** și **eficientă** a 3 metode directe de sortare (*Sortarea Bulelor*, *Sortarea prin Inserție* – folosind inserție liniară sau binară, și *Sortarea prin Selecție*)

- Intrare: un șir de numere $< a_1, a_2, \ldots, a_n >$
- Ieșire: o permutare ordonată a șirului de la intrare $< a'_1 \leq a'_2 \leq \ldots \leq a'_n >$

Toate informațiile necesare și pseudo-codul se găsesc în notițele de la **Seminarul nr. 1** (Sortarea prin Inserție este prezentată și în **carte(?) – secțiunea 2.1**). Să verificați că ați implementat varianta eficientă pentru fiecare din algoritmii de sortare (dacă mai multe versiuni au fost prezentate)

1.2 Cerințe minimale pentru notare

- Interpretați graficul și notați observațiile personale în antetul fișierului *main.cpp*, într-un comentariu bloc informativ.
- Pregatiti un exemplu pentru exemplificarea corectitudinii fiecărui algoritm implementat
- 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.***

1.3 Cerințe

1.3.1 Implementarea metodelor de sortare (5.5p)

- Bubble sort (1.5p)
- Insertion sort (2p)
- Selection sort (2p)

Corectitudinea algoritmilor va trebui demonstrată pe un vector de dimensiuni mici (care poate să fie codat în funcția *main*).

1.3.2 Analiză algoritmilor pentru caz mediu statistic (1.5p – 0.5 per algoritm)

! Înainte de a începe să lucrați pe partea de evaluare a complexitatii algoritmilor, asigurati-va că aveti un algoritm corect!

Se cere compararea celor 3 algoritmi în cazurile: **favorabil (best)**, **mediu statistic (average)** și **defavorabil (worst)**. Pentru cazul **mediu** va trebui să repetați măsurătorile de **m** ori (m=5 este suficient) și să raportați media rezultatelor; de asemenea, pentru cazul **mediu,** asigurați-vă că folosiți **aceleași** date de intrare pentru cele 3 metode de sortare (astfel încât compararea lor să fie corectă); identificați și generați date de intrare pentru cazurile: **favorabil** și **defavorabil**, pentru toate cele 3 metode de sortare.

Pașii de analiză ai metodelor de sortare pentru fiecare din cele 3 cazuri (**favorabil, defavorabil, mediu)**:

- variați dimensiunea șirului de la intrare (n) între $[100 \ldots 10.000]$, cu un increment de maxim 500 (sugerăm 100).

- pentru fiecare dimensiune, generati datele de intrare adecvate pentru metoda de sortare; rulați metoda de sortare numărând operațiile (numărul de atribuiri, numărul de comparații și suma lor).

> **!** Doar atribuirile (=) și comparațiile (<, ==, >, !=) care se fac pe datele de intrare și pe datele auxiliare corespunzătoare se iau în considerare.

1.3.3 Analiză in caz favorabil și defavorabil (3p - câte 0.5p pentru fiecare caz a fiecărui algoritm)

Pentru fiecare caz de analiză (**favorabil, defavorabil** si **mediu),** generati grafice care compara cele 3 metode de sortare; folosiți grafice diferite pentru numărul de atribuiri, comparații și suma lor. Dacă o curbă nu poate fi vizualizată corect din cauza că celelalte curbe au o rată mai mare de creștere (ex: o funcție liniară pare constantă atunci când este plasată în același grafic cu o funcție pătratică), atunci plasați noua curbă și pe un alt grafic. Denumiti adecvat graficele și curbele.

Corectitudinea algoritmilor va trebui demonstrată pe un vector de dimensiuni mici (care poate să fie codat în funcția "main").

1.3.4 Bonus: Insertion sort prin inserție binară (0.5p)

Corectitudinea algoritmilor va trebui demonstrată pe un vector de dimensiuni mici (care poate să fie codat în funcția "main").

Va trebui să comparați insertion sort prin inserție binară cu toți ceilalți algoritmi (bubble, insertion, selection) pentru toate cazurile (favorabil, mediu statistic, defavorabil).