

1 Tema Nr. 4: Interclasarea eficientă a k liste ordonate

Timp alocat: 2 ore

1.1 Implementare

Se cere implementarea **corectă** și **eficientă** a unei metode de complexitate $O(n \log k)$ pentru **interclasarea a k liste sortate**. Unde n este numărul total de elemente (Sugestie: folosiți un heap, vezi notițele de la *Seminarul al 2-lea*).

Cerințe de implementare:

- Folosiți liste înlățuite pentru a reprezenta cele k secvențe sortate și secvența de ieșire

Intrare: k șiruri de numere $< a_1^i, a_2^i, \dots, a_{m_i}^i >$, $\sum_{i=1}^k m_i = n$

Ieșire: o permutare a reuniunii șirurilor de la intrare $a'_1 \leq a'_2 \leq \dots \leq a'_n$

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 dumea voastră la întrebările puse de către profesor.*

1.3 Cerințe

1.3.1 Demo pentru generarea a k liste aleatoare sortate de dimensiuni diferite (având în total n elemente, unde n și k sunt date) și interclasarea a 2 liste (5p)

Demo: Corectitudinea algoritmului (*generare și interclasare*) va trebui demonstrată pe date de intrare de dimensiuni mici. Generați liste de dimensiuni $k=4$,

$n=20$ și interclasăți două dintre acestea.

1.3.2 Adaptare operațiilor de *min-heap* pe structura nouă și interclasarea a k liste (3p)

Demo: Corectitudinea algoritmului de (*interclasare*) va trebui demonstrată pe date de intrare de dimensiuni mici (ex: $k=4$, $n=20$). În cazul în care ați ajuns la acest punct, demostrația de interclasare de la primul demo nu este necesară, dar generarea este.

1.3.3 Evaluarea algoritmului în cazul mediu statistic (2p)

Se cere analiza algoritmului în cazul **mediu statistic**. Pentru cazul **mediu statistic** va trebui să repetați măsurările de câteva ori. Din moment ce k și n pot varia, se va face o analiză în felul următor:

- Se alege, pe rând, 3 valori constante pentru k (**k1=5, k2=10, k3=100**); generează k siruri **aleatoare** sortate pentru fiecare valoare a lui k astfel încât numărul elementelor din toate sirurile să varieze între **100 și 10000** cu un increment maxim de 400 (sugerăm 100); rulați algoritmul pentru toate valorile lui n (pentru fiecare valoare a lui k); generați un grafic ce reprezintă **suma atribuirilor și a comparațiilor** făcute de acest algoritm pentru fiecare valoare a lui k (în total sunt 3 curbe).
- Se alege **$n=10.000$** ; valoarea lui k va varia între **10 și 500** cu un increment de 10; generați k siruri **aleatoare** sortate pentru fiecare valoare a lui k astfel încât numărul elementelor din toate sirurile să fie 10000; testați algoritmul de interclasare pentru fiecare valoare a lui k și generați un grafic care reprezintă **suma atribuirilor și a comparațiilor**.