1 Tema Nr. 8: Mulțimi disjuncte

Timp alocat: 2 ore

1.1 Implementare

Se cere implementarea **corectă** și **eficientă** a operațiilor de bază pe **mulțimi disjuncte** (*capitolul 21.1* din [1]) și a algoritmului lui **Kruskal** (găsirea arborelui de acoperire minimă, *capitolul 23.2* din [1]) folosind mulțimi disjuncte.

Se cere să folosiți o pădure de arbori pentru reprezentarea mulțimilor disjuncte. Fiecare arbore trebuie extins cu un câmp *rank* (înălțimea arborelui).

Operațiile de bază pe **mulțimi disjuncte** sunt**:**

- MAKE_SET (x)
  - creează o mulțime nouă ce conține elementul x
- UNION (x, y)
  - realizează reuniunea dintre mulțimea care îl conține pe x și mulțimea care îl conține pe y
  - euristica *union by rank* ține cont de înălțime celor doi arbori pentru a realiza reuniunea dintre mulțimi
  - pseudocodul poate fi găsit la *capitolul 21.3*[1]
- FIND_SET (x)
  - caută mulțime în care se află x
  - euristica *path compression* leagă toate elementele de pe ramura cu x la rădăcina arborelui

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 dumeavostră la întrebările puse de către profesor.***

1.3 Cerințe

1.3.1 Implementare corectă a MAKE_SET, UNION și FIND_SET (5p)

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

- creați (MAKE) 10 mulțimi + afișare conținuturilor seturilor
- executați secvența UNION și FIND_SET pentru 5 elemente + afișare conținuturilor seturilor

1.3.2 Implementarea corectă și eficientă a algoritmului lui Kruskal (2p)

Demo: Corectitudinea algoritmului va trebui demonstrată pe date de intrare de dimensiuni mici.

- creați un graf cu 5 noduri și 9 muchii + **afișare muchii**
- aplicarea algoritmului lui Kruskal + **afișarea muchiilor alese**

1.3.3 Evaluarea operațiilor pe mulțimi disjuncte (MAKE, UNION, FIND) folosind algoritmului lui Kruskal (3p)

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

O dată ce sunteți siguri că algoritmul funcționează corect:

- variați n de la 100 la 10000 cu un pas de 100
- pentru fiecare n
  - construiți un graf **conex**, **neorientat** și **aleatoriu** cu ponderi pe muchii (**n** noduri, **n*4** muchii)
  - determinați arborele de acoperire minima folosind algoritmul lui Kruskal
    - evaluați efortul computațional al fiecărei operații de bază (MAKE, UNION, FIND – *reprezentați rezultatele sub forma unui grafic cu trei serii*) pe mulțimi disjuncte ca suma comparațiilor și atribuțiilor efectuate; astfel, ar trebui să existe **3 serii în grafic**, câte una pentru fiecare operație.

References

[1] Thomas H. Cormen et al. *Introduction to Algorithms*. 2nd. The MIT Press, 2001. ISBN: 0262032937.