Seminar 3
Complexitatea problemelor.

Ștefan Ciobâcă, Dorel Lucanu
Universitatea "Alexandru Ioan Cuza", Iași

Săptămâna 2 Martie - 6 Martie 2020

1. Să se construiască arbori de decizie pentru sortare pentru fiecare dintre următorii algoritmi: heapsort, mergesort, bubblesort, pentru $n = 3$.
2. Să se dea un exemplu de arbore de decizie (cu noduri $i?j$) care **nu rezolvă** problema sortării.
3. Să se demonstreze lemele pentru proprietățile arborilor de decizie pentru căutare.
4. Formalizați următoarele reduceri:
   (a) cmmmc la cmmdc,
   (b) egalitatea de mulțimi la sortare,
   (c) incluziunea de mulțimi la sortare,
   (d) disjuncția de mulțimi la sortare.
5. Să se găsească limite inferioare pentru problemele:
   (a) egalitatea de mulțimi,
   (b) incluziunea de mulțimi,
   (c) disjunctia de mulțimi,
   (d) interclasarea a doua secvențe ordonate.
6. Cum definim complexitatea $O(f(n))$ a unei probleme? Dar $\Omega(f(n))$?
7. Ce înseamnă algoritm optim pentru o problemă?
8. Arătați că dacă un algoritm A rezolvă o problemă P în timp $O(f)$ și $f = O(g)$, atunci problema P are complexitatea $O(g)$.
9. Justificaţi, plecând de la definiţie, că $0.5n^2 - 3n = \theta(n^2)$.
10. Arătați că dacă f este polinom de grad k, atunci $f(n) \in \theta(n^k)$.
11. Arătați că $\theta(f) = O(f) \cap \Omega(f)$ pentru orice funcție f.
12. Fie $T(n) = \begin{cases} \theta(1) & \text{dacă } n = 1 \\ 2T(n/2) + \theta(n) & \text{dacă } n > 1. \end{cases}$
    Arătaţi că $T(n) = \theta(nlog(n))$.
13. Arătaţi prin inducţie că următoarea funcţie întoarce rezultatul $3^n - 2^n$:

```
f(n)
{
  if (n == 0 || n == 1) {
    return n;
  }
  return 5 * f(n - 1) - 6 * f(n - 2);
}
```

14. Fie F șirul lui Fibonacci, definit astfel:

    - $F(0) = 0$
    - $F(1) = 1$
    - $F(n) = F(n-1) + F(n-2)$, pentru orice număr natural $n \geq 2$.

    Funcția f de mai jos primește la intrare un argument n și calculează $F(n)$:

```
f(n)
{
  if (n <= 1) {
    return n;
  } else {
    return f(n - 1) + f(n - 2);
  }
}
```

    Arătați că

$$F(n) = \frac{1}{\sqrt{5}}(\phi^n - (-\phi)^{-n}),$$

    unde $\phi = \frac{(1+\sqrt{5})}{2}$. Arătați că complexitatea algoritmului implementat în funcția f este $\Theta(\phi^n)$. Concluzionați că complexitatea problemei rezolvate este $O(2^n)$.

15. Arătați că dacă $T_1(n) = O(f_1(n))$ și $T_2(n) = O(f_2(n))$, atunci $T_1(n) + T_2(n) = O(f_1(n) + f_2(n))$.

16. Scrieți în Alk algoritmul căutării binare, care rezolvă următoarea problemă:

    `Input:` a - număr întreg și x - un vector de numere întregi, ordonat crescător

    `Output` o poziție i pe care se găsește a în x sau -1, dacă nu se găsește

    și testați algoritmul pe câteva exemple.

17. Scrieți în Alk algoritmul căutării Fibonacci (vezi curs) pentru aceeași problemă și testați-l pe câteva exemple.

18. Analizați complexitatea algoritmilor de la cele două probleme de mai sus. Sunt algoritmii la fel de eficienți din punct de vedere asimptotic în cazul cel mai nefavorabil? Ce concluzie se poate trage despre complexitatea problemei căutării într-un vector ordonat?

19. Implementaţi în Alk algoritmul mergesort și scrieți un număr de teste relevante care să arate că algoritmul funcționeaza conform descrierii problemei (cel puțin pentru cazurile de test).