
Tema 1: Rezolvarea algoritmică a problemelor. Descrierea algoritmilor în pseudocod. Verificarea corectitudinii algoritmilor.

Verificare (prin test la seminar): în săptămâna 4-8.11.2019

Submiterea temelor: Se uploadează prin Classroom (cf indicațiilor primite de la cadrul didactic coordonator de seminar) o arhiva de tip *.zip având numele construit pe baza regulii: NumeStudent_seria_subgrupa_Tema1.zip (de exemplu AdamEva_IR_sgr3_Tema1.zip).

Arhiva trebuie să conțină următoarele fișiere:

- Un fișier în format PDF sau DOC care să conțină soluțiile propuse (răspunsuri la întrebări, algoritmi descriși în pseudocod, explicații etc.)
- Câte un fișier cu codul sursă Python corespunzător implementărilor solicitate în enunț denumit NumeStudent_seria_subgrupa_Tema1_Problema (de exemplu AdamEva_IR_sgr3_Tema1_Pb1b.py)

-
1. *Operații exacte cu numere raționale.* Un număr rațional poate fi întotdeauna specificat ca o fracție ireductibilă (raportul a două numere întregi prime între ele). Se pune problema definirii unor funcții corespunzătoare operațiilor aritmetice (adunare, scădere, înmulțire, împărțire) care să permită obținerea unor rezultate exacte (atât operanzii cât și rezultatele vor fi specificate ca perechi de valori care definesc o fracție ireductibilă). De exemplu pentru operația $1/2+3/4$, funcția de adunare va prelua numărătorul și numitorul fiecărei fracții (adică patru numere întregi: 1,2,3,4) și va returna numărătorul și numitorul rezultatului (5 și 4).
- (a) Descrieți un algoritm care primește ca parametri numărătorul și numitorul unei fracții și returnează valorile corespunzătoare fracției ireductibile echivalente (pentru parametrii 6 și 9 se vor returna valorile 2 și 3).
- (b) Descrieți algoritmi corespunzători operațiilor de adunare, scădere, înmulțire și împărțire a fracțiilor. Fiecare algoritm va primi ca parametri de intrare numărătorul și numitorul fiecărei fracții și va returna numărătorul și numitorul rezultatului.
- (c) Descrieți un algoritm de calcul a valorii unui polinom cu coeficienți întregi pentru un argument număr rațional. Se consideră că polinomul este reprezentat prin tabloul coeficienților (un polinom $c_n X^n + c_{n-1} X^{n-1} + \dots c_1 X + c_0$ este reprezentat printr-un tablou $c[0..n]$). De exemplu pentru polinomul $2X^3 - X^2 + X + 2$ valoarea calculată în $1/2$ este $5/2$. Pentru toate operațiile aritmetice se vor folosi funcțiile descrise la punctul (b). **Indicație.** Se poate folosi ca punct de pornire algoritmul de la Problema 12/Seminar 4.
- (d) Descrieți un algoritm care determină mulțimea tuturor rădăcinilor raționale ale unui polinom cu coeficienți numere întregi. **Indicație:** rădăcinile raționale ale unui polinom cu coeficienți întregi au proprietatea că sunt de forma d_0/d_n unde d_0 este un divizor (pozitiv sau negativ) al termenului liber (c_0) iar d_n este un divizor al coeficientului termenului de grad maxim (c_n). Mulțimea divizorilor luați în considerare include pe 1 și numărul însuși. Nu toate numerele raționale de forma descrisă sunt rădăcini (trebuie selectate doar cele pentru care valoarea polinomului este 0).

Pentru toți algoritmi descriși mai sus se vor implementa funcții Python corespunzătoare.

2. *Reprezentarea numerelor întregi în complement față de 2* pe k biți se caracterizează prin: (i) primul bit este folosit pentru codificarea semnului (0 pentru valori pozitive respectiv 1 pentru valori negative); (ii) ceilalți $(k - 1)$ biți sunt utilizați pentru reprezentarea în baza 2 a valorii (cifrele binare corespunzătoare în cazul numerelor pozitive, respectiv valori complementate după regula specifică în cazul numerelor negative). *Exemplu:* Pentru $k = 8$ reprezentarea lui 12 este 00001100 iar reprezentarea lui -12 este 11110100. În cazul valorilor negative, șirul cifrelor binare este parcurs începând cu cifra cea mai puțin semnificativă până la întâlnirea primei valori egale cu 1 - toate cifrele binare parcurse (toate zerourile de la sfârșit și primul 1 întâlnit) sunt lăsate nemodificate, iar toate cele care vor fi parcurse ulterior vor fi complementate).

- (a) Care este cel mai mare și cel mai mic număr întreg care pot fi reprezentate (în complement față de 2) pe 16 poziții binare (biți)? Argumentați răspunsul.
- (b) Propuneți un algoritm care construiește reprezentarea în complement față de 2 pe 16 poziții binare a unui număr întreg primit ca parametru.
Exemplu: pentru valoarea 12 se obține $[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0]$ iar pentru -12 se obține $[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0]$.
- (c) Propuneți un algoritm care determină valoarea unui număr întreg (pozitiv sau negativ) pornind de la reprezentarea în complement față de 2 pe 16 biți.
- (d) Propuneți un algoritm care calculează suma a două numere întregi date prin reprezentările lor în complement față de 2 pe $k = 8$ poziții binare. Identificați cazurile în care se produce *depășire* (rezultatul nu poate fi reprezentat corect pe $k = 8$ poziții binare). De exemplu, prin adunarea cifrelor binare aflate pe aceeași poziție (începând de la cea mai puțin semnificativă poziție) și transferul reportului către poziția imediat superioară (ca semnificație) pentru $[0, 1, 1, 1, 1, 0, 0, 0]$ și $[0, 0, 1, 1, 1, 0, 0, 1]$ s-ar obține $[1, 0, 1, 1, 0, 0, 0, 1]$, ceea ce nu e corect însemnând că s-a produs o depășire.

Fiecare dintre algoritmii de mai sus va fi descris în pseudocod și implementat în Python.

3. *Aproximarea funcției logaritmice prin serii.* Funcția \ln poate fi aproximată folosind următoarele serii:

$$f(x) = \begin{cases} \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n} (x-1)^n & 0 < x \leq 1 \\ \sum_{n=1}^{\infty} \frac{1}{n} \left(\frac{x-1}{x}\right)^n & x > 1 \end{cases}$$

- (a) Identificați regula de calcul a termenului următor (T_{n+1}) din fiecare serie folosind valoarea termenului curent ($T_n = \frac{(-1)^{n+1}}{n} (x-1)^n$ respectiv $T_n = ((x-1)/x)^n/n$).
- (b) Pentru fiecare dintre cele două serii descrieți algoritmul (și funcțiile Python corespunzătoare) care aproximează suma seriei prin suma finită $T_1 + T_2 + \dots + T_k$. Numărul de termeni din sumă se stabilește în funcție de valoarea ultimului termen adăugat (T_k este primul termen cu proprietatea că $|T_k| < \epsilon$, ϵ fiind o constantă cu valoare mică). Date de test: $\epsilon = 10^{-5}$, $x = 0.5$, $x = 1$, $x = 5$, $x = 10$. Determinați în fiecare caz numărul de termeni incluși în sumă.
- (c) Pentru unul dintre algoritmii propuși la punctul (b) (la alegere) identificați un invariant și demonstrați corectitudinea algoritmului.
- (d) Descrieți un algoritm pentru estimarea erorii de aproximare $\sum_{i=1}^m (f(i \cdot h) - \ln(i \cdot h))^2$.
Date de test: $m = 100$, $h = 5/m$.

4. Se consideră algoritmii `alg1` și `alg2`. Pentru fiecare dintre cei doi algoritmi:

- (a) Implementați fiecare algoritm în Python.
- (b) Stabiliți ce returnează fiecare dintre algoritmi atunci când este apelat pentru valori naturale nenule ale parametrilor. Identificați o proprietate invariantă și demonstrați corectitudinea fiecărui algoritm. *Indicație:* pentru `alg2` se poate folosi proprietatea că pentru orice număr natural nenul n există un număr natural $k > 0$ astfel încât $2^{k-1} \leq n < 2^k$.

```
1: alg1(int a, b)
2: if a < b then
3:   a ↔ b
4: end if
5: c ← 0
6: d ← a
7: while d > b do
8:   c ← c + 1
9:   d ← d - b
10: end while
11: return c, d
```

```
1: alg2(int n)
2: i ← 1
3: x ← 0
4: while i ≤ n do
5:   i ← 2 * i
6:   x ← x + 1
7: end while
8: return x
```

5. Se consideră următoarea tehnică de compresie a unui șir de litere: ”dacă o literă este precedată și urmată de o altă literă atunci ea este transferată ca atare în textul comprimat; dacă o literă apare pe mai multe poziții consecutive atunci în textul comprimat subsecvența care conține aceeași literă va fi înlocuită cu o pereche formată din litera respectivă și numărul de repetări”. De exemplu, textul ”urrraa” se va înlocui cu ”ur4a2”. Descrieți ideea de rezolvare și implementați în Python câte o funcție pentru:

- (a) compresia unui șir de litere;
- (b) decompresia pornind de la un șir ”comprimat”.