
Curs 1: Introducere in algoritmică

- Noțiunea de algoritm
 - Obiectul disciplinei
 - Proprietăți ale algoritmilor
 - Date
 - Tipuri de prelucrări
 - Exerciții
-

1 Noțiunea de algoritm

În termeni generali un *algoritm* este o metodă de rezolvare pas cu pas a *problemelor*. O problemă se consideră a fi constituită din *date* de intrare și un enunț care specifică relația existentă între datele de intrare și *soluția* problemei. În cadrul algoritmului sunt descrise prelucrările necesare pentru a obține soluția problemei pornind de la datele de intrare. În continuare considerăm că

Un algoritm este o succesiune bine precizată de prelucrări care aplicate asupra datelor de intrare ale unei probleme permit obținerea în timp finit a soluției acesteia.

Termenul de algoritm provine de la numele unui matematician persan, al-Khowarizmi (al-Kwarizmi), ce a trăit în secolul al IX-lea și care a scris o lucrare despre efectuarea calculelor numerice într-o manieră algebrică. Primul algoritm se consideră a fi algoritmul lui Euclid (utilizat pentru determinarea celui mai mare divizor comun a două numere naturale). Termenul de algoritm poate fi înțeles în sens larg nefiind neapărat legat de rezolvarea unei probleme cu caracter științific, ci doar pentru a descrie într-o manieră ordonată activități care constau în parcurgerea unei succesiuni de pași (cum este de exemplu utilizarea unui telefon public sau a unui bancomat).

În matematică există o serie de algoritmi: cel al rezolvării ecuației de gradul doi, algoritmul lui Eratostene (pentru generarea numerelor prime mai mici decât o anumită valoare), schema lui Horner (pentru determinarea câtului și restului împărțirii unui polinom la un binom) etc.

Soluția problemei se obține prin *execuția* algoritmului. Algoritmul poate fi executat pe o mașină formală (în faza de proiectare și analiză) sau pe o mașină fizică (calculator) după ce a fost codificat într-un limbaj de programare. Spre deosebire de un program, care depinde de un limbaj de programare, un algoritm este o entitate matematică care este independentă de mașina pe care va fi executat.

2 Obiectul disciplinei

Obiectul disciplinei de "Algoritmă" îl reprezintă studiul algoritmilor din perspectiva elaborării și analizei lor. Elaborarea unui algoritm necesită:

- cunoștințe specifice domeniului de unde provine problema de rezolvat;
- tehnici generale de rezolvare a problemelor;
- intuiție și gândire algoritmică.

Analiza algoritmilor presupune verificarea corectitudinii acestora și a eficienței. Un algoritm este considerat corect dacă prin aplicarea lui asupra datelor problemei conduce la soluția acestuia și eficient dacă prin execuția acestuia rezultatul se obține într-un interval de timp rezonabil. Analiza presupune aplicarea unor tehnici de demonstrare specifice matematicii.

Indiferent de complexitatea unei aplicații informatice, la bazele ei stau algoritmi destinați rezolvării problemelor fundamentale ale aplicației. Oricât de sofisticată ar fi tehnologia software utilizată (interfețe, structurare globală a aplicației) eficiența aplicației este în mod esențial determinată de eficiența algoritmilor implicați. Un algoritm prost conceput nu poate fi "reparat" prin artificii de programare. Din acest motiv dobândirea unor cunostinte solide privind elaborarea și analiza algoritmilor este o condiție necesară în formarea unui bun programator.

3 Proprietăți ale algoritmilor

Un algoritm trebuie să posedă următoarele proprietăți:

Generalitate. Un algoritm destinat rezolvării unei probleme trebuie să permită obținerea rezultatului pentru orice date de intrare și nu numai pentru date particulare de intrare.

Finitudine. Un algoritm trebuie să admită o descriere finită și fiecare dintre prelucrările pe care le conține trebuie să poată fi executată în timp finit. Prin intermediul algoritmilor nu pot fi prelucrate structuri infinite.

Rigurozitate. Prelucrările algoritmului trebuie specificate riguros, fără ambiguități. În orice etapă a execuției algoritmului trebuie să se știe exact care este următoarea etapă ce va fi executată.

Eficiență. Algoritmii pot fi efectiv utilizați doar dacă folosesc *resurse de calcul* în volum acceptabil. Prin resurse de calcul se înțelege volumul de memorie și timpul necesar pentru execuție.

Exemple.

1. *Nu orice problemă poate fi rezolvată algoritmic.* Considerăm un număr natural n și următoarele două probleme: (i) să se construiască mulțimea divizorilor lui n ; (ii) să se construiască mulțimea multiplilor lui n . Pentru rezolvarea primei probleme se poate elabora ușor un algoritm, în schimb pentru a doua problemă nu se poate scrie un algoritm atâta timp cât nu se cunoaște un criteriu de oprire a prelucrărilor.

2. *Un algoritm trebuie să funcționeze pentru orice dată de intrare.* Să considerăm problema ordonării crescătoare a șirului de valori: $(2, 1, 4, 3, 5)$. O modalitate de ordonare ar fi următoarea: se compară primul element cu al doilea iar dacă nu se află în ordinea bună se interschimbă (în felul acesta se obține $(1, 2, 4, 3, 5)$); pentru șirul astfel transformat se compară al doilea element cu al treilea și dacă nu se află în ordinea dorită se interschimbă (la această etapă șirul rămâne neschimbat); se continuă procedeul până penultimul element se compară cu ultimul. În felul acesta se obține $(1, 2, 3, 4, 5)$.

Deși metoda descrisă mai sus a permis ordonarea crescătoare a șirului $(2, 1, 4, 3, 5)$ ea nu poate fi considerată un algoritm de sortare întrucât dacă este aplicată șirului $(3, 2, 1, 4, 5)$ conduce la $(2, 1, 3, 4, 5)$.

2. *Un algoritm trebuie să se oprească.* Se consideră următoarea secvență de prelucrări:

Pas 1. Atribuie variabilei x valoarea 1;

Pas 2. Mărește valoarea lui x cu 2;

Pas 3. Dacă x este egal cu 100 atunci se oprește prelucrarea altfel se reia de la Pas 2.

Este ușor de observat că x nu va lua niciodată valoarea 100, deci succesiunea de prelucrări nu se termină niciodată. Din acest motiv nu poate fi considerată un algoritm corect.

3. *Prelucrările dintr-un algoritm trebuie să fie neambigue.* Considerăm următoarea secvență de prelucrări:

Pas 1. Atribuie variabilei x valoarea 0;

Pas 2. *Fie* se mărește x cu 1 *fie* se micșorează x cu 1;

Pas 3. Dacă $x \in [-10, 10]$ se reia de la Pasul 2, altfel se oprește algoritmul.

Atât timp cât nu se stabilește un criteriu după care se decide dacă x se mărește sau se micșorează secvența de mai sus nu poate fi considerată un algoritm. Ambiguitatea poate fi evitată prin utilizarea unui limbaj riguros de descriere a algoritmilor. Să considerăm că Pas 2 se înlocuiește cu:

Pas 2. Se aruncă o monedă. Dacă se obține cap se mărește x cu 1 iar dacă se obține pajură se micșorează x cu 1;

În acest caz specificarea prelucrărilor nu mai este ambiguă chiar dacă la execuții diferite se obțin rezultate diferite. Ce se poate spune despre finitudinea acestui algoritm ? Dacă s-ar obține alternativ cap respectiv pajură, prelucrarea ar putea fi infinită. Există însă și posibilitatea să se obțină de 11 ori la rând cap (sau pajură), caz în care procesul s-ar opri după 11 repetări ale pasului 2. Atât timp cât șansa ca algoritmul să se termine este nenulă algoritmul poate fi considerat corect (în cazul prezentat mai sus este vorba despre un algoritm aleator).

4. *Un algoritm trebuie să se oprească după un interval rezonabil de timp.* Să considerăm că rezolvarea unei probleme implică prelucrarea a n date și că numărul de prelucrări $T(n)$ depinde de n . Presupunem că timpul de execuție a unei prelucrări este 10^{-3} s și că problema are dimensiunea $n = 100$. Dacă se folosește un algoritm caracterizat prin $T(n) = n$ atunci timpul de execuție va fi $100 \times 10^{-3} = 10^{-1}$ secunde. Dacă, însă se folosește un algoritm caracterizat prin $T(n) = 2^n$ atunci timpul de execuție va fi de circa 10^{27} secunde adică aproximativ 10^{19} ani.

4 Date

Prelucrările efectuate în cadrul unui algoritm se efectuează asupra unor *date*. Acestea sunt entități purtătoare de informație (relevantă pentru problema de rezolvat). Considerăm datele ca fiind containere ce conțin informație, *valoarea* curentă a unei date fiind informația pe care o conține la un moment dat. În funcție de rolul jucat în cadrul algoritmului datele pot fi *constante* (valoarea lor rămâne nemodificată pe parcursul algoritmului) sau *variabile* (valoarea lor poate fi modificată).

Din punctul de vedere al informației pe care o poartă datele pot fi:

- *Simple*: conțin o singură valoare (poate fi un număr, o valoare de adevăr sau un caracter).
- *Structurate*: sunt constituite din mai multe date simple între care există o relație de structură. Dacă toate datele componente au aceeași natură atunci structura este *omogenă*, altfel este o structură *heterogenă*.

Datele structurate omogene ce vor fi utilizate în continuare sunt cele destinate reprezentării unor structuri algebrice simple: *mulțime finită* (ansamblu de valori distincte pentru care nu are importanță ordinea în care sunt reținute), *șir finit* (ansamblu de valori nu neapărat distincte pentru care are importanță ordinea în care sunt reținute) și *matrice* (tabel bidimensional de valori).

Pentru reprezentarea acestor date vom folosi structura de *tablou* caracterizată prin faptul că fiecare valoare componentă poate fi specificată prin precizarea unuia sau mai multor indici. Cel mai frecvent sunt folosite tablourile unidimensionale (pentru reprezentarea șirurilor și mulțimilor) și cele bidimensionale (pentru reprezentarea matricilor).

5 Tipuri de prelucrări

Asemenea datelor și prelucrările pot fi clasificate în simple și structurate. Prelucrările simple sunt:

- *Atribuire*. Permite afectarea unei valori unei variabile. Valoarea atribuită poate fi rezultatul evaluării unei expresii. O *expresie* descrie un calcul efectuat asupra unor date și conține *operanții* (specifică datele asupra cărora se efectuează calculele) și *operatori* (specifică prelucrările ce se vor efectua).
- *Transfer*. Permite preluarea datelor de intrare ale problemei și furnizarea rezultatului.
- *Control*. În mod normal prelucrările din algoritm se efectuează în ordinea în care sunt specificate. În cazul în care se dorește modificarea ordinii naturale se transferă controlul execuției la o anumită prelucrare.

Structurile de prelucrare sunt:

- *Secvențială*. Este o succesiune de prelucrări (simple sau structurate). Execuția structurii secvențiale constă în execuția prelucrărilor componente în ordinea în care sunt specificate.
- *De decizie (alternativă)*. Permite specificarea situațiilor în care în funcție de realizarea sau nerealizarea unei *condiții* se efectuează o prelucrare sau o altă prelucrare. Condiția este de regulă o expresie a cărui rezultat este o valoare logică (adevărat sau fals). O astfel de prelucrare apare de exemplu în evaluarea unei funcții definite prin:

$$f(x) = \begin{cases} -1 & \text{dacă } x < 0 \\ 0 & \text{dacă } x = 0 \\ 1 & \text{dacă } x > 0 \end{cases}$$

- *De ciclare (repetitivă)*. Permite modelarea situațiilor când o prelucrare trebuie repetată. Se caracterizează prin existența unei prelucrări care se repetă și a unei condiții de oprire (sau de continuare). În funcție de momentul în care este analizată condiția există prelucrări repetitive condiționate *anterior* (condiția este analizată înainte de a efectua prelucrarea) și prelucrări condiționate *posterior* (condiția este analizată după efectuarea prelucrării). O astfel de prelucrare apare de exemplu în calculul unei sume finite $\sum_{i=1}^n 1/i^2$. În acest caz prelucrarea care se repetă este adunarea iar condiția de oprire o reprezintă faptul că au fost adunați toți cei n termeni.

6 Exerciții.

1. Se consideră următoarea metodă de înmulțire a două numere întregi x și y . Se scrie x alături de y (pe aceeași linie). Se împarte x la 2 și câtul împărțirii se scrie sub x (restul se ignoră). Se înmulțește y cu 2 iar produsul se scrie sub y . Procedeeul continuă construindu-se astfel două coloane de numere. Calculele se opresc în momentul în care pe prima coloană se obține valoarea 1. Se adună toate valorile de pe coloana a doua care corespund unor valori impare aflate pe prima coloană.

Este metoda descrisă un algoritm ? Este el corect (determină produsul celor două numere) ?

Indicație. Ca urmare a împărțirilor succesive la 2 valorile de pe prima coloană descresc până se ajunge la un cât egal cu 1. Prin urmare prelucrarea este finită. Corectitudinea prelucrării derivă din faptul că metoda realizează de fapt conversia în baza 2 a primului număr iar produsul se obține prin înmulțirea succesivă a celui de al doilea număr cu puteri ale lui 2 și prin însumarea acelor produse care corespund unor cifre nenule în reprezentarea binară a primului număr.

2. Propuneți un algoritm pentru determinarea părții întregi a unui număr real.

Indicație. Se va ține cont de semnul numărului. În cazul în care este pozitiv se scade succesiv valoarea 1 până când se ajunge la o valoare mai mică strict decât 1. Numărul de scăderi efectuate indică valoarea părții întregi. Pentru numere negative se adună succesiv 1 până se obține o valoare mai mare sau egală cu 0.

3. Considerăm că o vânzătoare dispune doar de monede de 2 unități și 5 unități. Propuneți un algoritm care să stabilească modul de plată a unui rest (restul este de cel puțin 4).

Indicație. Dacă restul este un număr par atunci vânzătoarea poate da doar monede de 2. Dacă numărul este impar vânzătoarea poate da o monedă de 5 iar ceea ce rămâne (un număr par) în monede de 2.

4. La un restaurant bucătarul a pregătit clătite pe care le-a așezat pe un platou sub forma unei stive. Din păcate nu toate clătitele au același diametru astfel că stiva nu arată prea frumos. Chelnerul ia platoul și având la dispoziție o spatulă reușește să aranjeze cu o singură mână clătitele astfel încât să fie în ordinea descrescătoare a diametrelor. Cum a procedat? Descrieți problema într-o manieră abstractă.

Indicație. Rearanjarea se face efectuând doar mișcări de răsturnare a unui "set" de clătite dintre cele aflate în partea de sus.

5. Considerăm următoarele două probleme:

(a) O gospodină a făcut o serie de cumpărături și are la dispoziție două sacoșe. Problema constă în a distribui cumpărăturile în cele două sacoșe astfel încât diferența între greutatea celor două sacoșe să fie cât mai mică.

(b) Se consideră două dispozitive de stocare (de exemplu discuri magnetice) și o mulțime de fișiere a căror dimensiuni însumate nu depășește capacitatea globală a celor două discuri. Se încearcă repartizarea fișierelor în cele două discuri astfel încât diferența dintre spațiile rămase libere să fie cât mai mică.

Este vreo legătură între cele două probleme ? Găsiți metode de rezolvare.

Indicație. Ambele probleme pot fi formalizate astfel: se consideră o mulțime de numere pozitive, $A = \{a_1, a_2, \dots, a_n\}$ și se cere să se determine două submulțimi disjuncte B și C

astfel încât $B \cup C = A$ și $|\sum_{a \in B} a - \sum_{a \in C} a|$ este minimă. Cea mai simplă metodă este cea a "forței brute" prin care se generează toate perechile de submulțimi (B, C) și se alege cea care minimizează diferența specificată. Numărul de partiții distincte este $2^{n-1} - 1$ dacă n este impar și $2^{n-1} - 1 + C_n^{n/2}/2$ dacă n este par. Pentru n mare, numărul de partiții ce trebuie testate devine mare (pentru $n = 10$ este 637 iar pentru $n = 100$ este de ordinul 10^{29}). Ar trebui găsită o metodă mai eficientă ...