Curs 1: Introducere in algoritmică
- Noţiunea de algoritm
- Obiectul disciplinei
- Proprietăţi ale algoritmilor
- Date
- Tipuri de prelucrări
- Exerciţii

1 Noţiunea 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 şi un enunţ care specifică relaţia existentă între datele de intrare şi *soluţia* problemei. În cadrul algoritmului sunt descrise prelucrările necesare pentru a obţine soluţia 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 obţinerea în timp finit a soluţiei acesteia.*

Termenul de algoritm provine de la numele unui matematician persan, al-Khowarizmi (al-Kwarizmi), ce a trăit în secolul al IX-lea şi 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 înţeles în sens larg nefiind neapărat legat de rezolvarea unei probleme cu caracter ştiinţific, ci doar pentru a descrie într-o manieră ordonată activităti care constau în parcurgerea unei succesiuni de paşi (cum este de exemplu utilizarea unui telefon public sau a unui bancomat).

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

Soluţia problemei se obţine prin *execuţia* algoritmului. Algoritmul poate fi executat pe o maşină formală (în faza de proiectare şi analiză) sau pe o maşină 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 maşina pe care va fi executat.

2 Obiectul disciplinei

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

- cunoştinţe specifice domeniului de unde provine problema de rezolvat;
- tehnici generale de rezolvare a problemelor;
- intuiţie şi gândire algoritmică.

Analiza algoritmilor presupune verificarea corectitudinii acestora şi a eficienţei. Un algoritm este considerat corect dacă prin aplicarea lui asupra datelor problemei conduce la soluţia acestuia şi eficient dacă prin execuţia acestuia rezultatul se obţine intr-un interval de timp rezonabil. Analiza presupune aplicarea unor tehnici de demonstrare specifice matematicii.

Indiferent de complexitatea unei aplicaţii informatice, la bazele ei stau algoritmi destinaţi rezolvării problemelor fundamentale ale aplicaţiei. Oricât de sofisticată ar fi tehnologia software utilizată (interfeţe, structurare globală a aplicaţiei) eficienţa aplicaţiei este în mod esenţial determinată de eficienţa algoritmilor implicaţi. Un algoritm prost conceput nu poate fi "reparat" prin artificii de programare. Din acest motiv dobandirea unor cunostinte solide privind elaborarea si analiza algoritmilor este o condiţie necesară în formarea unui bun programator.

3 Proprietăţi ale algoritmilor

Un algoritm trebuie să posede următoarele proprietăţi:

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

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

Rigurozitate. Prelucrările algoritmului trebuie specificate riguros, fără ambiguităţi. În orice etapă a execuţiei algoritmului trebuie să se ştie exact care este următoarea etapă ce va fi executată.

Eficienţă. Algoritmii pot fi efectiv utilizaţi doar dacă folosesc *resurse de calcul* în volum acceptabil. Prin resurse de calcul se înţelege volumul de memorie şi timpul necesar pentru execuţie.

Exemple.

1.*Nu orice problemă poate fi rezolvată algoritmic.* Considerăm un număr natural n şi următoarele două probleme: (i) să se construiască mulţimea divizorilor lui n; (ii) să se construiască mulţimea multiplilor lui n. Pentru rezolvarea primei probleme se poate elabora uşor un algoritm, în schimb pentru a doua problemă nu se poate scrie un algoritm atâta timp cât nu se cunoaşte un criteriu de oprire a prelucrărilor.
2.*Un algoritm trebuie să funcţioneze pentru orice dată de intrare.* Să considerăm problema ordonării crescătoare a şirului de valori: $(2, 1, 4, 3, 5)$. O modalitate de ordonare ar fi următoarea: se compară primul element cu al doilea iar daca nu se află în ordinea bună se interschimbă (în felul acesta se obţine $(1, 2, 4, 3, 5)$); pentru şirul astfel transformat se compară al doilea element cu al treilea şi dacă nu se află în ordinea dorită se interschimbă (la această etapă şirul rămâne neschimbat); se continuă procedeul până penultimul element se compară cu ultimul. În felul acesta se obţine $(1, 2, 3, 4, 5)$.

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

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

Pas 1. Atribuie variabilei x valoarea 1;

Pas 2. Măreşte valoarea lui x cu 2;

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

Este uşor 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 secvenţă de prelucrări:

Pas 1. Atribuie variabilei x valoarea 0;

Pas 2. *Fie* se măreşte x cu 1 *fie* se micşorează x cu 1;

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

Atât timp cât nu se stabileşte un criteriu după care se decide dacă x se măreşte sau se micşorează secvenţa 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 înlocuieşte cu:

Pas 2. Se aruncă o monedă. Dacă se obţine cap se măreşte x cu 1 iar dacă se obţine pajură se micşorează x cu 1;

În acest caz specificarea prelucrărilor nu mai este ambiguă chiar dacă la execuţii diferite se obţin rezultate diferite. Ce se poate spune despre finitudinea acestui algoritm ? Dacă s-ar obţine alternativ cap respectiv pajură, prelucrarea ar putea fi infinită. Există însă şi posibilitatea să se obţină de 11 ori la rând cap (sau pajură), caz în care procesul s-ar opri după 11 repetări ale pasului 2. Atât timp cât şansa 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 şi că numărul de prelucrări $T(n)$ depinde de n. Presupunem că timpul de execuţie a unei prelucrări este 10^{-3}s şi că problema are dimensiunea $n = 100$. Dacă se foloseşte un algoritm caracterizat prin $T(n) = n$ atunci timpul de execuţie va fi $100 \times 10^{-3} = 10^{-1}$ secunde. Dacă, însă se foloseşte un algoritm caracterizat prin $T(n) = 2^n$ atunci timpul de execuţie 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 entităţi purtătoare de informaţie (relevantă pentru problema de rezolvat). Considerăm datele ca fiind containere ce conţin informaţie, *valoarea* curentă a unei date fiind informaţia pe care o conţine la un moment dat.În funcţie 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 informaţiei pe care o poartă datele pot fi:

- *Simple:* conţin 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 relaţie de structură. Dacă toate datele componente au aceeaşi 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: *mulţime finită* (ansamblu de valori distincte pentru care nu are importanţă ordinea în care sunt reţinute), *şir finit* (ansamblu de valori nu neapărat distincte pentru care are importanţă ordinea în care sunt reţinute) şi *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 şirurilor şi mulţimilor) şi cele bidimensionale (pentru reprezentarea matricilor).

5 Tipuri de prelucrări

Asemenea datelor şi prelucrările pot fi clasificate în simple şi 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 şi conţine *operanzii* (specifică datele asupra cărora se efectuează calculele) şi *operatori* (specifică prelucrările ce se vor efectua).
- *Transfer.* Permit preluarea datelor de intrare ale problemei şi furnizarea rezultatului.
- *Control.* În mod normal prelucrările din algoritm se efectuează în ordinea în care sunt specificate. În cazul în care se doreşte modificarea ordinii naturale se transferă controlul execuţiei la o anumită prelucrare.

Structurile de prelucrare sunt:

- *Secvenţială.* Este o succesiune de prelucrări (simple sau structurate). Execuţia structurii secvenţiale constă în execuţia prelucrărilor componente în ordinea în care sunt specificate.
- *De decizie (alternativă).* Permite specificarea situaţiilor în care în funcţie de realizarea sau nerealizarea unei *condiţii* se efectuează o prelucrare sau o altă prelucrare. Condiţia 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 funcţii 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 situaţiilor când o prelucrare trebuie repetată. Se caracterizează prin existenţa unei prelucrări care se repetă şi a unei condiţii de oprire (sau de continuare). În funcţie de momentul în care este analizată condiţia există prelucrări repetitive condiţionate *anterior* (condiţia este analizată înainte de a efectua prelucrarea) şi prelucrări condiţionate *posterior* (condiţia 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 condiţia de oprire o reprezintă faptul că au fost adunaţi toţi cei n termeni.

6 Exerciţii.

1. Se consideră următoarea metodă de înmulţire a două numere întregi x şi y. Se scrie x alături de y (pe aceeaşi linie). Se împarte x la 2 şi câtul impărţirii se scrie sub x (restul se ignoră). Se inmulţeşte y cu 2 iar produsul se scrie sub y. Procedeul continuă construindu-se astfel două coloane de numere. Calculele se opresc în momentul în care pe prima coloană se obţine 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) ?

   Indicaţie. Ca urmare a împărţirilor 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 obţine prin înmulţirea succesivă a celui de al doilea număr cu puteri ale lui 2 şi prin însumarea acelor produse care corespund unor cifre nenule în reprezentarea binară a primului număr.

2. Propuneţi un algoritm pentru determinarea părţii întregi a unui număr real.

   Indicaţie. Se va ţine cont de semnul numărului. În cazul în care este pozitiv se scade succesiv valoarea 1 până căa nd se ajunge la o valoare mai mică strict decât 1. Numărul de scăderi efectuate indică valoarea părţii întregi. Pentru numere negative se adună succesiv 1 până se obţine o valoare mai mare sau egală cu 0.

3. Considerăm că o vânzătoare dispune doar de monede de 2 unităţi şi 5 unităţi. Propuneţi un algoritm care să stabileasca modul de plată a unui rest (restul este de cel puţin 4).

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

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

   Indicaţie. Rearanjarea se face efectuând doar mişcări de răsturnare a unui "set" de clătite dintre cele aflate in partea de sus.

5. Considerăm următoarele două probleme:
   (a) O gospodină a făcut o serie de cumpărături şi are la dispoziţie două sacoşe. Problema constă în a distribui cumpărăturile în cele două sacoşe astfel încât diferenţa între greutăţile celor două sacoşe să fie cât mai mică.
   (b) Se consideră două dispozitive de stocare (de exemplu discuri magnetice) şi o mulţime de fişiere a căror dimensiuni însumate nu depăşeşte capacitatea globală a celor două discuri. Se încearcă repartizarea fişierelor în cele două discuri astfel încât diferenţa dintre spaţiile rămase libere să fie cât mai mică.
   Este vreo legătură între cele două probleme ? Găsiţi metode de rezolvare.

   Indicaţie. Ambele probleme pot fi formalizate astfel: se consideră o mulţime de numere pozitive, $A = \{a_1, a_2, \ldots, a_n\}$ şi se cere să se determine două submulţimi disjuncte B şi C

astfel încât $B \cup C = A$ şi $|\sum_{a \in B} a - \sum_{a \in C} a|$ este minimă. Cea mai simplă metodă este cea a "forţei brute" prin care se generează toate perechile de submulţimi (B, C) şi se alege cea care minimizează diferenţa specificată. Numărul de partiţii distincte este $2^{n-1} - 1$ dacă n este impar şi $2^{n-1} - 1 + C_n^{n/2}/2$ dacă n este par. Pentru n mare, numărul de partiţii 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 ...