Conceptul de ALGORITM - abordare MODERNĂ

Marin Vlada

Dezvoltarea informaticii actuale se datorează cercetărilor, rezultatelor și experiențelor din domeniile sistemelor de calcul, algoritmicii și programării, dar mai ales a interdependenței acestor domenii prin așa-numita triadă "sistem de calcul - algoritmică - programare". La baza acestei interdependențe se află conceptul de algoritm, concept care a construit pentru om o nouă filosofie: "gândirea algoritmică". Această gândire algoritmică a făcut posibilă apariția și dezvoltarea Tehnologiei Informației (IT) care reprezintă de fapt implementarea filosofiei procesării, gestionării și transmiterii informațiilor.

În cele ce urmează vom realiza o abordare modernă a conceptului de algoritm și vom evidenția necesitatea eliminării abordărilor vechi care, din păcate, mai există în multe manuale adresate elevilor și studenților.

Mulți dintre autorii manualelor de informatică din țara noastră greșesc și dau o importanță minoră conceptului de algoritm, atunci când consideră că "*algoritmul este o noțiune primară*".

Dezvoltarea unei gândiri algoritmice se poate realiza doar prin activități de instruire și formare care să aibă la bază conceptul fundamental de ***algoritm*** într-o abordare modernă și nu una învechită. Abordarea propusă va putea fi adaptată la orice nivel de instruire și la orice arie curriculară.

Introducere

Întâmplător sau nu, deceniul 7 al secolului al XX-lea a fost unul al marilor schimbări în domeniul informaticii și al sistemelor de calcul:

- ***inventarea microprocesorului*** ("*bijuteria de bază*" a sistemelor de calcul actuale) ca urmare a rezultatelor din trei domenii fundamentale: sisteme, circuite integrate și microprogramare; a urmat construirea și răspândirea pe scara largă a sistemelor de calcul de tip *PC* (*Personal Computer*), impulsionarea dezvoltării rețelelor de calculatoare, precum și apariția și dezvoltarea de noi sisteme de operare; performanțe sporite ale dispozitivelor de *intrare / ieșire*;
- ***supremația și răspândirea structurilor de control*** în *algoritmică* și *programare*; apariția *limbajului pseudocod* pentru reprezentarea și elaborarea algoritmilor; conceperea și scrierea primelor limbaje de programare care implementează *structurile de control*, adaptarea continuă a limbajelor de programare prin implementarea *structurilor de control*, a *structurilor de date*, precum și a *facilităților programării orientate obiect* (*OOP - Object Oriented Programming*);
- ***succese spectaculoase în domeniul inteligenței artificiale*** prin construirea primelor sisteme expert; conceperea și scrierea primului limbaj de programare logică (limbajul *Prolog*) care oferă suportul ***programării declarative***; dezvoltarea și utilizarea largă a metodelor și tehnicilor *inteligenței artificiale* în rezolvarea câtorva dintre cele mai dificile probleme;
- ***delimitarea problemelor*** rezolvate cu calculatorul (*probleme decidabile*) în două clase distincte: clasa problemelor rezolvate prin *metode imperative* (*procedurale*) și clasa problemelor rezolvate prin *metode declarative*; delimitarea clasei *problemelor nedecidabile*; apariția și dezvoltarea tehnicilor pentru comunicarea și căutarea la distanță a informațiilor.

Toate aceste aspecte au fost și sunt într-o interdependență continuă ținând seama de particularitatea informaticii care oferă sisteme de calcul performante și produse *software* competitive în rezolvarea problemelor. Utilizarea efi-

cientă a sistemelor de calcul și a produselor *software* necesită o instruire continuă, atât pentru programatori, cât și pentru utilizatori.

Dezvoltarea gândirii algoritmice trebuie luată în considerare ca obiectiv în instruire, atunci când se învață *algoritmica* (metode și tehnici) dar și când se învață *programarea* (utilizarea limbajelor de programare).

Practica instruirii elevilor și studenților a demonstrat că învățarea unui limbaj de programare este în general "mai ușoară" decât învățarea elaborării algoritmilor (algoritmica). Acest lucru se poate justifica prin faptul că elaborarea unui algoritm este echivalentă cu implementarea (reprezentarea) raționamentelor (proceselor demonstrative) deduse din metodele și tehnicile utilizate în rezolvarea unei probleme.

Rezolvarea problemelor necesită nu numai cunoștințe clare și precise, dar și capacitate de sinteză și control și mai ales capacitate de creație. Dacă vrem să facem o analogie, un ***programator*** poate fi "*compozitorul*" care realizează o "*lucrare muzicală*".

În toate etapele instruirii trebuie avută în vedere următoarea interdependență: "***sistem de calcul - algoritmică - programare***".

Succesele unui concurent la olimpiadele de informatică depind de cunoașterea utilizării unui limbaj de programare modern, dar mai ales depind de "bogăția" și stăpânirea cunoștințelor în elaborarea algoritmilor. Și mai există încă ceva: experința acumulată în activitatea de rezolvare a problemelor prin formarea unei gândiri algoritmice solide și consistente. Același lucru se poate afirma și despre succesele unui programator care lucrează într-o firmă de *software*.

Complexitatea problemelor care necesită descrierea mai multor procese de calcul complexe a determinat folosirea noțiunii de *algoritm* în activitatea de rezolvare a problemelor. Multe procese naturale, multe activități umane, pot fi descrise într-o *formă algoritmică* prin definirea unor *informații* și *acțiuni* clare și precise, eliminându-se ambiguitățile în *interpretare* și în *operații*. Algoritmizarea este o cerință fundamentală în rezolvarea oricărei probleme cu ajutorul calculatorului.

Experiența a demonstrat că nu orice problemă poate fi rezolvată prin algoritmizarea rezolvării, adică prin descrierea unui algoritm de rezolvare. Așa s-a delimitat *clasa problemelor decidabile* (o problemă este ***decidabilă*** dacă există *un algoritm* pentru rezolvarea ei) de *clasa problemelor nedecidabile* (o problemă este ***nedecidabilă*** dacă nu există un *algoritm* pentru rezolvarea ei).

Un algoritm implementează diverse metode și tehnici de rezolvare care au fost descoperite sau definitivate într-un anumit moment în evoluția științifică a domeniului respectiv.

Există algoritmi care urmează metode dezvoltate înainte de apariția calculatoarelor, dar cele mai multe probleme cer abordări noi. Chiar dacă ne gândim numai la "problema celor patru culori", care a fost rezolvată (în anul 1977) doar prin utilizarea calculatorului și prin utilizarea unei metode "noi" (metoda ***backtracking***), facem dovada acestei afirmații.

Gândirea algoritmică

În etapa actuală de dezvoltare științifică și tehnică, rezolvarea unei probleme dintr-un domeniu (informatică, matematică, fizică, chimie etc.) reprezintă *o activitate de creație*, *un raționament* prin *construirea*, *generarea*, *descrierea* unui:

- ***proces demonstrativ*** (*demonstrația*) care să arate *existența* uneia sau mai multor soluții și / sau să determine efectiv soluțiile exacte;
- ***proces computațional*** (*algoritmul*) care să codifice un *proces demonstrativ*, o metodă sau o tehnică de rezolvare în scopul determinării (eventual aproximative) a soluțiilor exacte.

În general, în procesul (activitatea) de rezolvare a unei probleme dintr-un anumit domeniu (științific, economic, social etc.), este necesară evidențierea *ipotezei* (condițiile, stările inițiale, parametri inițiali) și a *concluziei* (cerințele, obiectivele, scopurile) din analiza și studiul enunțului problemei.

Procesul de rezolvare (raționamentul) constă în utilizarea selectivă a *legilor*, *teoremelor*, *propozițiilor* etc. din domeniul problemei, pentru ca pornind de la *ipoteză*, prin aplicarea succesivă a *legilor*, *teoremelor* etc. să se obțină *concluzia* problemei.

Legătura dintre *ipoteză*, *concluzie* și *procesul de rezolvare* (raționamentul) determină o structură asemănătoare conceptului de program, și anume (*Ipoteză - Date de intrare; Concluzie - Date de ieșire*):

Figura 1

De exemplu, procesul de rezolvare a unei probleme poate fi descris astfel: există un număr de teoreme $T_1, \ldots, T_n$ determinate de ipotezele $I_1, \ldots, I_n$ și de concluziile $C_1, \ldots, C_n$; acestea sunt selectate și apoi aplicate astfel încât să se poată realiza identificările : $I \supseteq I_1$, $C_1 \supseteq I_2$, ..., $C_n \supseteq C$, adică:

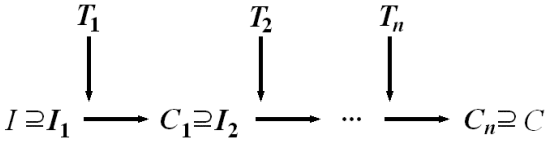

Figura 2

Această prezentare poate fi înțeleasă dacă, de exemplu, facem analogia cu rezolvarea problemelor de geometrie

care necesită delimitarea *ipotezei* și a *concluziei*, ca apoi să se utilizeze numai acele *teoreme, propoziții, proprietăți* care, pornind de la *ipoteză*, prin aplicarea lor succesivă, să se obțină *concluzia* cerută.

Evident, din experința căpătată în rezolvarea problemelor, se poate afirma faptul că selectarea teoremelor și aplicarea lor se poate realiza doar prin "stăpânirea" domeniului respectiv la nivel de specialist sau de expert.

Este cunoscut faptul că ființele umane pot rezolva o problemă în trei moduri (metode evidențiate de inteligența artificială):

- pornind de la *ipoteză* și obținând *concluzia* ("*metoda înlănțuirii înainte*");
- pornind de la *concluzie* și obținând *ipoteza* ("*metoda înlănțuirii înapoi*");
- pornind simultan, de la *ipoteză* și *concluzie* ("*metoda mixtă*").

În toate științele (matematică, fizică, chimie etc.) există multe probleme care, chiar dacă au fost studiate într-o anumită etapă, abordarea lor cu metode și tehnici moderne poate să trezească oricând interesul tinerilor, al cercetătorilor în general. Această afirmație poate fi susținută cu atât mai mult cu cât, astăzi, prin utilizarea unor metode specifice tehnologiei informației și prin utilizarea calculatorului, se pot concepe tehnici care altă dată erau imposibil de imaginat.

Din punct de vedere metodologic, trebuie să ne obișnuim să reformulăm problemele în mod explicit și adecvat rezolvării lor matematice sau - când este posibil - cu ajutorul calculatorului. Pentru acest lucru trebuie să cunoaștem "*limitele*" gândirii demonstrative și în același timp "*limitele*" gândirii algoritmice, apoi să cunoaștem performanțele calculatoarelor.

Să amintim aici istoria problemei celor patru culori care începe în anul 1852, când studentul englez ***Francis Guthrie*** a enunțat următoarea problemă pentru colorarea unei hărți [11, 21]:

"*Sunt suficiente patru culori pentru a colora o hartă care reprezintă diverse țări, cu condiția ca oricare două țări vecine (care au frontiera comună) să fie colorate cu culori diferite*".

Mulți matematicieni cu renume au studiat această problemă (***A. de Morgan***, ***A. Cayley***, ***A. B. Kempe***, ***H. Heesch***, ***K. Appel***, ***W. Hakel***), dar rezolvarea ei completă nu a fost posibilă decât în anul 1977 (***K. Appel***, ***W. Hakel***, "*The four color problem*" în Mathematics Today, L. A. Steen(ed.), Springer Verlag, 1978) și acest lucru numai după ce s-a utilizat calculatorul, deoarece fiind o problemă combinatorială cu spațiul soluțiilor de ordin foarte mare, a fost necesară conceperea unei metode algoritmice pentru căutarea eficientă cu ajutorul calculatorului.

Teoretic, problema a fost rezolvată în cazul a cinci culori. De altfel, rezolvarea acestei probleme a impus în informatică o metodă devenită clasică în domeniul elaborării algoritmilor: metoda ***backtracking***.

În zilele noastre apar tot mai des probleme a căror rezolvare (*proces demonstrativ*) este rezultatul îmbinării metodelor pur matematice cu metodele algoritmice (computaționale).

Uneori, rezultatele computaționale pot oferi surse de inspirație pentru metodele matematice. Informaticienii au și venit în întâmpinarea matematicienilor și specialiștilor prin elaborarea diverselor sisteme de programe care realizează calcule numerice, logice, simbolice, reprezentări spațiale etc. cu ajutorul cărora pot fi dobândite cunoștințe de matematică sau se pot rezolva probleme din diverse domenii.

Totuși, aceste aspecte nu trebuie să conducă la neglijarea procesului demonstrativ. Cele mai utilizate astfel de sisteme de programe sunt: ***Mathematica***, ***Statistica***, ***Matlab***, ***Mathcad***, ***Derive***, ***Maple***, ***Origin***, ***SlideWrite***, ***Workplace***, ***Eureka*** etc.

Este cunoscut faptul că în științele naturii se obțin rezultate deosebite prin îmbinarea metodelor teoretice cu metodele experimentale. Pentru matematică (și alte științe), informatica (tehnologia informației) este cea care oferă rezultate experimentale prin utilizarea calculatorului în rezolvarea problemelor.

Metoda algoritmică va fi un experiment pentru justificarea rezultatelor procesului demonstrativ de la metoda matematică, dar în același timp poate să fie o metodă de rezolvare independentă.

Metoda algoritmică trebuie să substituie rezultatele obținute pe cale teoretică prin metode computaționale eficiente ținând seama de limitele proceselor algoritmice și de performanțele calculatoarelor. Este adevărat că uneori metodele matematice pot conduce la simplificarea metodelor algoritmice și invers.

Elevii care participă la olimpiadele școlare de informatică au constatat că, de multe ori, o analiză profundă a abordării matematice poate să conducă la obținerea unei metode algoritmice eficiente.

Participanții la olimpiadele de matematică pot constata același lucru la o analiză profundă a abordării algoritmice. De aici, importanța atât a proceselor demonstrative (metode matematice), cât și a proceselor computaționale (metode algoritmice).

În funcție de natura metodelor / tehnicilor implementate în procesele computaționale, algoritmii pot fi: numerici, seminumerici, formali, combinatoriali, neuronali, de căutare, de sortare, recursivi, de rescriere, secvențiali, paraleli, determiniști, nedeterminiști, probabilistici, aleatori, euristici, de tip *Monte Carlo*, genetici, de simulare, geometrici computaționali etc.

Definiții matematice

Al-Khwarizmi, Muhammed ibn Musa (cca. 780-cca. 850) a folosit pentru prima dată reguli precise și clare pentru a descrie procese de calcul (operații aritmetice fundamentale)

în lucrarea sa "*Scurtă carte despre calcul algebric*".


Figura 3

Mai târziu, această descriere apare sub denumirea de algoritm în "*Elementele lui Euclid*". Algoritmul lui *Euclid* pentru calculul celui mai mare divizor comun a două numere naturale este, se pare, primul algoritm cunoscut în matematică.

Inițial, noțiunea de algoritm a fost considerată primară. În matematica modernă noțiunea de algoritm a primit mai multe definiții:

- algoritmul normal al lui *A. A. Markov*;
- algoritmul operațional al lui *A. A. Leapunov*;
- mașina *Turing* / mașina cu stivă;
- funcții recursive;
- sisteme *POST*.

Prima descriere a definiției matematice pentru noțiunea de algoritm a fost dată de matematicianul rus *A. A. Markov*. S-a demonstrat că din punct de vedere matematic aceste definiții sunt echivalente.

În prezent, în domeniul probabilităților și statisticii, sunt cunoscute așa-numitele *procese Markov*. De altfel, astăzi se întâlnește foarte frecvent conceptul de proces sub diverse ipostaze: în reprezentarea și execuția algoritmilor, în funcționarea sistemelor de calcul, în execuția sistemelor de operare, în execuția programelor, în funcționarea rețelelor de calculatoare etc.

În [5] se precizează: "*… vom înțelege prin problemă algoritmică o funcție total definită P: I → F, unde I este mulțimea datelor de intrare ale algoritmului, iar F este mulțimea informațiilor finale. Vom presupune că I și F sunt nevide și cel mult numărabile. O instanță a problemei P este problema P(i), pentru o valoare particulară i din I. Intuitiv, considerăm că algoritmul este o pereche de funcții (predicate) a = (f, g), unde f: I → [0,1], g: I × F → [0, 1]. Predicatul f descrie elementele lui I care pot fi folosite ca întrebări, iar predicatul g indică legăturile care trebuie să existe între valorile de intrare și cele de ieșire la terminarea algoritmului pentru care rezultatele furnizate de algoritm să fie corecte.*"

Ținând seama de o astfel de definiție, algoritmul (*f*, *g*) este parțial corect pentru problema *P*, dacă pentru orice instanță *P*(*i*) este adevarată în ipoteza că *g*(*I*, *p*(*I*)) există. De asemenea, algoritmul (*f*, *g*) este total corect pentru problema *P*, dacă pentru orice instanță *P*(*i*), cu *f*(*i*) adevărată, *g*(*i*, *P*(*i*)) există și este adevarată.

În [6] se consideră că "*Un algoritm poate fi definit ca o funcție f: D → F, unde F este mulțimea informațiilor finale, iar D este mulțimea informațiilor inițiale*".

Se evidențiază faptul că un algoritm utilizează o memorie virtuală și că prin executarea instrucțiunilor algoritmului se realizează schimbări ale valorilor unor locații de memorie, astfel că starea celulelor generează o succesiune de stări ale algoritmului (în execuție, algoritmul este un proces dinamic).

O imagine intuitivă asupra noțiunii de algoritm este definiția următoare: se presupun *n* celule de memorie c_1, c_2, …, c_n, în care se pot introduce numere sau expresii, asfel încât fiecare celulă conține, la un moment dat, un număr sau o expresie ori poate să fie vidă. Celulele c_i, $i = 1, 2, ..., n$, împreună cu informația conținută în ele formează starea inițială S_0. În funcție de aceasta, algoritmul *f* efectuează anumite modificări asupra conținutului celulelor c_i. Se utilizează asfel o stare intermediară S_1. Pornind de la S_1, algoritmul *f* asigură în același mod, trecerea la o stare S_2 etc. Un sistem de reguli al algoritmului precizează care stare, obținută la un moment dat, poate fi declarată stare finală S_k: $S_0 \rightarrow S_1 \rightarrow \ldots \rightarrow S_{k-1} \rightarrow S_k$.

Vom vedea în cele ce urmează că pentru execuția unui algoritm este necesară o mașina virtuală pentru interpretarea și execuția instrucțiunilor descrise de algoritm. Această execuție constă în generarea și desfășurarea unui proces dinamic care va utiliza o memorie virtuală și un procesor virtual pentru realizarea operațiilor invocate de execuția instrucțiunilor.

De fapt, procesele care se generează într-un sistem de calcul prin execuția unui program scris într-un limbaj de programare, sunt reprezentări efective și reale ale proceselor care se generează prin execuția unui algoritm pe o mașină virtuală; mașina virtuală trebuie să simuleze o mașină reală, un calculator și trebuie să fie constituită din memorie virtuală, procesor virtual și dispozitive de intrare / ieșire virtuale).

Definiții informatice

"*Prin algoritm se înțelege o prescripție care determină un proces de calcul și care este precisă, perfect inteligibilă, nepermițând nici un fel de interpretări din partea celui care o duce la îndeplinire*"[8].

În [2] noțiunea de algoritm se definește astfel: "*Algoritmul desemnează o mulțime exhaustivă și univoc determinată de operații, împreună cu succesiunea în care trebuie aplicate asupra datelor inițiale ale problemei, pentru a obține soluția. El este considerat mod de prezentare a procesului de rezolvare a problemelor și suport după care se realizează programul*".

De asemenea, în [7] se precizează următoarele: "*Algoritmul este o noțiune primară. De obicei, prin algoritm se înțelege o secvență finită și ordonată de operații, care pornind de la o mulțime finită de date (inițiale), conduce (prin aplicarea în mod mecanic și uneori repetată a operațiilor) la o mulțime finită de rezultate.* […] *Modul de rezolvare a unei probleme este un algoritm, căci pornind de la date (prezentate în enunțul problemei) și aplicând acestora un număr finit de operații ordonate (transformări, raționa-*

mente etc.), ajungând la soluția problemei, adică la rezultate".

În [1] conceptul de algoritm este descris astfel: "*Prin algoritm vom înțelege o secvență finită de comenzi explicite și neambigue care executate pentru o mulțime de date (ce satisfac anumite condiții inițiale I), conduce în timp finit la rezultatul corespunzător.*"

De asemenea, pentru înțelegerea noțiunii de algoritm, în [2] se precizează: "*Soluția unei probleme, din punct de vedere informatic, este dată printr-o mulțime de comenzi (instrucțiuni) explicite și neambigue, exprimate într-un limbaj de programare. Această mulțime de instrucțiuni prezentată conform anumitor reguli sintactice formează un program. Un program poate fi privit și ca un algoritm exprimat într-un limbaj de programare. Totuși, un algoritm descrie soluția problemei independent de limbajul de programare în care este redactat programul*".

În legătură cu rezolvarea problemelor sub formă algoritmică, în [2] se arată faptul că nu este suficient ca o anumită problemă (clasă de probleme) să admită soluții (chiar și o singură soluție). Din punct de vedere al informaticii (al practicii), interesează dacă există soluția care poate fi descrisă algoritmic, deci dacă soluția problemei poate fi construită efectiv. De asemenea, pentru rezolvarea anumitor probleme pot exista mai mulți algoritmi (de exemplu, problemele de sortare [23]).

Stabilirea celui mai bun dintre aceștia se realizează în urma unui proces de analiză prin care se determină performanța fiecăruia [1, 5, 12, 24, 25, 26].

Principalele caracteristici ale unui algoritm trebuie să fie ([1, 12, 25]):

- ***generalitate*** - algoritmul trebuie să fie cât mai general astfel ca să rezolve o clasă cât mai largă de probleme;
- ***claritate / determinare*** - acțiunile algoritmului trebuie să fie clare, simple și riguros specificate;
- ***finitudine*** - acțiunile algoritmului trebuie să se termine după un număr finit de operații, aceasta pentru orice set de date valide;
- ***corectitudine*** - algoritmul trebuie să producă un rezultat corect (date de ieșire) pentru orice set de date de intrare valide;
- ***performanță*** - algoritmul trebuie să fie eficient privind resursele folosite, și anume să utilizeze memorie minimă și să se termine într-un timp minim.

Întreaga activitate de cercetare și elaborare de *software* din domeniul tehnologiei informației este determinată de inventarea, conceperea, elaborarea, testarea și implementarea de algoritmi performanți și utili.

Marea diversitate a algoritmilor și marea aplicabilitate a acestora în toate domeniile, face ca această temă să fie mereu actuală și într-o continuă schimbare și perfecționare.

Bibliografie

1. Albeanu Gr., *Algoritmi și limbaje de programare*, Universitatea "Spiru Haret", Editura România de Mâine, București, 2000
2. Apostol C., Roșca I. Gh., Roșca V., Ghilic-Micu B., *Introducere în programare. Teorie și aplicații*, Editura București, 1993
3. Georgescu H., *Programare concurentă. Teorie și aplicații*, Editura Tehnică, București, 1996
4. Kinston J. H., *Algorithms and Structures Design. Correctness, Analysis*, Addison Wesley, Longman, 1998
5. Mitrana V., *Provocarea algoritmilor. Probleme pentru concursurile de informatică*, Editura Agni, București, 1994
6. Odăgescu I., *Optimizarea algoritmilor*, Editura Militară, Bucuresti, 1991
7. Perjeriu E., Văduva I., *Îndrumar pentru lucrări de laborator la cursul de Bazele Informaticii*, Universitatea din București, 1986
8. Popovici C., Georgescu H., State L., *Bazele Informaticii*, vol. I, Universitatea din București, 1990
9. Skiena S., *The Algorithm Design Manual*, Springer Verlag, New York, Inc., 1998
10. Vlada M., *Informatica*, Universitatea din București, Editura Ars Docendi, București, 1999
11. Vlada M., *Poligoane stelate. Problema lui Hopf și Pannwitz*, Gazeta de matematică, nr. 8/1995, pag. 339-348
12. Zaharia M. D., *Structuri de date și algoritmi. Exemple în limbajele C și C++*, Editura Albastră, Cluj-Napoca, 2002
13. Cristea V., Giumale C., Kalisz E., Păunoiu Al., *Limbajul C standard*, Editura Teora, București, 1992
14. Popovici M. D., Popovici M. I., *C++. Tehnologia orientată spre obiecte. Aplicații*, Editura Teora, București, 2000
15. www.math.gatech.edu/~thomas/FC/fourcolor.html
16. mathworld.wolfram.com/Four-ColorProblem.html
17. spicerack.sr.unh.edu/~student/tutorial/fourColor/FourColor.html
18. mathcentral.uregina.ca/RR/database/RR.09.97/fisher1.html
19. www.uccs.edu/~asoifer/book5.html
20. www.uccs.edu/~asoifer/solve98.html
21. www-groups.dcs.st-and.ac.uk/~history/BigPictures/Guthrie.jpeg
22. mathworld.wolfram.com/Algorithm.html
23. www.cs.rit.edu/~atk/Java/Sorting/sorting.html
24. faculty.juniata.edu/rhodes/cs2/ch129.htm
25. www.scism.sbk.ac.uk/law/Section5/chap1/ s5c1p2.htm
26. www.ics.uci.edu/~eppstein/161/people.html
27. www.cs.pitt.edu/~kirk/algorithmcourses/
28. www.gnacademy.org/text/cc/
29. java.sun.com/docs/books/tutorial/java/concepts/
30. www.accu.org/acornsig/public/articles/oop_c.html
31. loki.cs.brown.edu:8081/webae/full.html

Domnul conf. dr. Marin Vlada este cadru didactic la Universitatea București și poate fi contactat prin e-mail la adresa vlada@chem.unibuc.ro.