Curs 1: INTRODUCERE

Tehnici avansate de programare

Lect.dr. Iulia Banu
Departamentul de Informatică,
Universitatea din București

semestrul 1, 2019

Cuprins

Algoritmi
corecți
eficienți

- identificarea structurilor de date și a tehnicilor potrivite;
- analiza corectitudinii și a eficienței soluțiilor propuse;
- numeroase aplicații, exemple:
  - Probleme de planificare optimă, folosire optimă a resurselor;
  - Programare jocuri, Branch and bound;
  - Geometrie computațională;
  - Cele mai apropiate două puncte dintr-o mulțime de puncte dată;
  - Procesare text;
  - Căutare web, similitudini;
- probleme/întrebări interviuri;

Watch your language!

Why does Python live on land? Because it is above C level!

Biscayne National Park Service https://wsvn.com/news

Watch your language!

- Interpreted language: one line at a time
- .py ==> bytocode ==> virtual machine
- Implementarea default CPython
- Alte implementari PyPy(RPython), IronPython (C#), JPython (Java)
- Utilizat de Google, Facebook, Instagram etc.
- Simplifica scrierea codului
- Numeroase module:
  - Web framework: Django
  - Machine learning: Numpy, SciPy, Theano, TensorFlow
  - Web scraping: Scrapy, BeautifulSoup
  - procesare imagini: PIL/Pillow, scikit-image etc.

Conținut curs

- Tehnici de programare:
  - Greedy
  - Divide et Impera
  - Programare Dinamica
  - Backtracking
  - Branch and Bound;
- Alte tipuri de algoritmi: algoritmi euristici, algoritmi probabiliști (Monte Carlo, Las Vegas);
- Algoritmi genetici;
- Principiul lui Dirichlet;
- Analiza complexitații unor algoritmi, NP-completitudine.

Tipuri de algoritmi

- *Algoritmi deterministi*. La executări diferite, pentru același input produc același rezultat.

- *Algoritmi probabiliști*. La executări diferite, pentru același input, pot produce rezultate diferite.
  Pașii unui algoritm probabilist depind de input și de o serie de alegeri aleatoare.

- *Euristici*. Găsirea unei aproximări a soluției exacte cu un algoritm mai rapid decât algoritmii care ar calcula soluția exactă.
  tradeoff corectitudine *vs* optimalitate, timp de execuție redus
  Cât de *bună* este aproximarea?

Probleme NP-complete, NP-hard

Dacă știm despre o problemă că este NP-completă sau NP-hard știm că descoperirea unui algoritm polinomial pentru rezolvarea ei ar însemna rezolvarea tuturor probelmelor NP-complete.

Dacă se găsește algoritm polinomial pentru o problemă NP-completă, atunci

$$\mathrm{NP} = \mathrm{P}$$

Verificare în timp polinomial = rezolvare în timp polinomial?

Millennium Prize Problems

Probleme NP-complete, NP-hard

- O problemă NP poate fi rezolvată în timp exponențial prin generarea tuturor soluțiilor candidat și verificarea în timp polinomial a fiecărei soluții candidat.
- O problemă de decizie B este **NP-completă** dacă B $\in$ NP și $\forall$A $\in$ NP, A $\leq$ p B.
- **NP-hard** O problemă B este în clasa problemelor NP-hard dacă $\forall$A $\in$ NP, A $\leq$ p B.
- Dacă se găsește algoritm polinomial pentru o problemă NP-completă, atunci P $=$ NP

Probleme NP-complete, NP-hard

- O problemă B este **NP-completă** dacă $B \in NP$ și $\forall A \in NP$, $A \leq p$ B.

Cum demonstrăm ca o problemă C este NP-completă?

- Arătăm că există verificator polinomial pentru C.
- Demonstră că o problemă NP-completă B poate fi redusă în timp polinomial la C.
- Demonstrăm că C are soluții dacă și numai dacă B are soluții.

Testare

specificații

verificăm dacă pentru un set de input-uri se obțin output-urile conforme cu specificațiile.

aserțiuni

se stabilesc condiții pe care le vom verifica în anumite momente ale execuției. Se pot testa:

- constrângeri asupra inpurt-urilor și output-urilor etc.
- invarianți

Demonstrarea corectitudinii

terminarea programului

Trebuie demonstrat că algoritmul se termină în timp finit.

Fie E o expresie calculată în funcție de variabilele programului. Se poate alege un șir $\{x_n\}$ strict descrescător de numere naturale pozitive, unde $\{x_n\}$ este valoarea expresiei E la pasul n al algoritmului.

corectitudine parțială

Presupunând că algoritmul se termină, rezultatul obținut la final este cel corect.

Invarianți = relații ce trebuie îndeplinite în orice stare a programului.

Corectitudinea algoritmilor

Exemplu

Metoda de înmulțire a țăranului rus:

```
x ← a; y ← b; p ← 0;
while x>0
    { xy+p=ab } (*)
    if x impar then p ← p+y;
    x ← x div 2;
    y ← y+y;
write(p);
```

Eficiența algoritmilor

Spațiu

Timpul de execuție

T(n) = timpul de executare pentru orice set de date de intrare de dimensiune n.

Definiție (Notație asimptotică)

Fie $f : \mathbb{N} \to \mathbb{R}_+$

$\mathcal{O}(g) := \{f | \exists c > 0, \exists n_0 \in \mathbb{N}\ a.i\ 0 \leq f(n) \leq cg(n)\ \forall n > n_0\}$

$\Omega(g) := \{f | \exists c > 0, \exists n_0 \in \mathbb{N}\ a.i\ 0 \leq cg(n) \leq f(n)\ \forall n > n_0\}$

$\Theta(g) := \{f | \exists c_1, c_2 > 0, \exists n_0 \in \mathbb{N}\ a.i$

$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\ \forall n > n_0\}$$

Eficiența algoritmilor

Exemplu: Să se determine minimul și maximul elementelor unui vector.

Orice algoritm corect va efectua cel puțin $\lceil 3n/2 \rceil - 2$ comparații.

if n impar then m $\leftarrow a_1$; M $\leftarrow a_1$; k $\leftarrow$ 2;
else if $a_1 < a_2$ then m $\leftarrow a_1$; M $\leftarrow a_2$;
    else m $\leftarrow a_2$; M $\leftarrow a_1$;
  k $\leftarrow$ 3;
while k$\leq$n-1
  if $a_k < a_{k+1}$ then
    if $a_k <$ m then m $\leftarrow a_k$;
    if $a_{k+1} >$ M then M $\leftarrow a_{k+1}$;
  else
    if $a_{k+1} <$ m then m $\leftarrow a_{k+1}$;
    if $a_k >$ M then M $\leftarrow a_k$;
  k $\leftarrow$ k+2;

Ore alocate, program

Întrebări, consultații sala **318**
iulia.banu@fmi.unibuc.ro

- curs 2h
- laborator 2h
- seminar 1h
- Python hour 1h
- probleme/proiecte suplimentare, consultații vineri 17-20

Modalitatea de evaluare

Săptămâna 14

- oficiu **10%**
- test laborator **30%**
- examen scris **30%**
- teme/proiecte laborator **30%**, maxim **5%** pentru fiecare temă
- seminar **1p bonus** se acordă doar dacă punctajul total, fără bonus este > 4.5

! condiție de promovare: total $>$ **45%** și
nota la testul de laborator minim **15%**

Bibliografie

- Horia Georgescu. Tehnici de programare. Editura Universității din București 2005
- Leon Livovschi, Horia Georgescu. Sinteza și analiza algoritmilor. 1986
- T.H. Cormen, C.E. Leiserson, R.R. Rivest – Introducere in algoritmi, Mit Press, trad. Computer Libris Agora
- Jon Kleinberg, Éva Tardos, Algorithm Design, Addison-Wesley 2005 http://www.cs.princeton.edu/~wayne/kleinberg-tardos/
- S. Dasgupta, C.H. Papadimitriou, U.V. Vazirani, Algorithms, McGraw-Hill, 2008
- https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011/
- http://moodle.fmi.unibuc.ro/course/