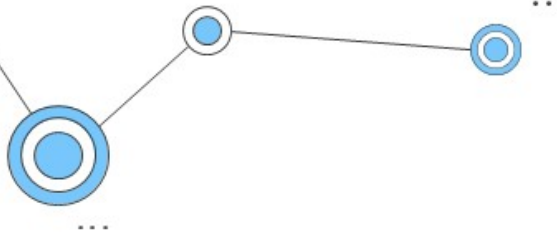


Algoritmi Fundamentali

Lector dr.
Dorin IORDACHE

...

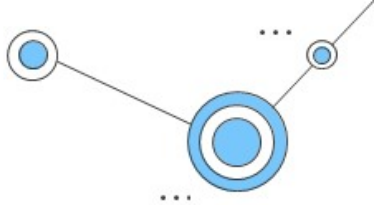


“Cei mai mulți programatori buni nu programează pentru că vor bani sau recunoștință, ci pentru că e distractiv să programezi.”

Linus TORVALD



Tematica disciplinei



1. Notiunea de algoritm si caracteristici
2. Complexitatea calculului
3. Algoritmi elementari
4. Metode de cautare si sortare
5. Metode avansate de sortare
6. Algoritmi recursivi
7. Metode de elaborare a algoritmilor
8. Algoritmi de cautare a subsirurilor de caractere
9. Algoritmi de compresie
Algoritmi de aproximare



Calendar



Evaluare



Examen – scris	- 60%
Participarea activa la laboratoare	- 30%
Rezolvare teme	- 10%



Bibliografie recomandată

1. Cormen, T; Leiserson, G; Rive, R.: Introducere in algoritmi, Comp. Libris Agora, Cluj, 2000.
2. Knuth D. E., Tratat de programarea calculatoarelor, vol. I, II, III, Ed. Teora, Bucuresti, 2002.
3. Livovschi, L, Georgescu, H.: Analiza si sinteza algoritmilor, Ed. St. si Enc., Bucuresti, 1986.
4. Andonie R., Gârbacea I., Algoritmi fundamentali, o perspectivă C++, Editura Libris, 1995



Cursul nr. 1

Algoritm

Agenda

01

Rezolvarea problemelor

...

02

Definire algoritm

...

03

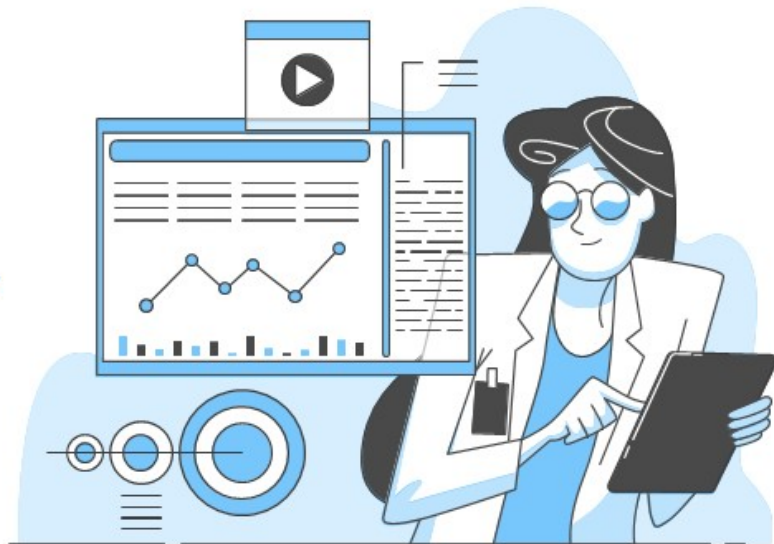
Proprietatile algoritmilor

...

04

Descrierea algoritmilor

...





01

Rezolvarea problemelor





ALGORITM

Cuvântul este derivat din fonetica pronunției prenumelui lui Abu Ja'far Mohammed ibn Musa **al-Khowarizmi**, un matematician arab care a inventat un set de reguli pentru realizarea celor patru operații aritmetice de bază (adunare, scădere, înmulțire și împărțire).

...

Originea cuvântului algoritm

al-Khowarizmi - matematician persan (790-840)

Muhammad ibn
Musa al-Khwarizmi

algorism → algorithm



- A fost printre primii ce a folosit cifra 0
- A scris prima carte de algebra (numele acestei discipline provine de la același matematician)



Muhammad ibn
Musa al-Khwarizmi

Al-Khwarizmi's method of solving linear and quadratic equations worked by first reducing the equation to one of six standard forms (where b and c are positive integers):

- Squares equal roots ($ax^2 = bx$).
- Squares equal number ($ax^2 = c$).
- Roots equal number ($bx = c$).
- Squares and roots equal number ($ax^2 + bx = c$). $x^2 + 10x = 39$.
- Squares and number equal roots ($ax^2 + c = bx$). $x^2 + 21 = 10x$.
- Roots and number equal squares ($bx + c = ax^2$). $3x + 4 = x^2$.

rezolvarea ecuației:

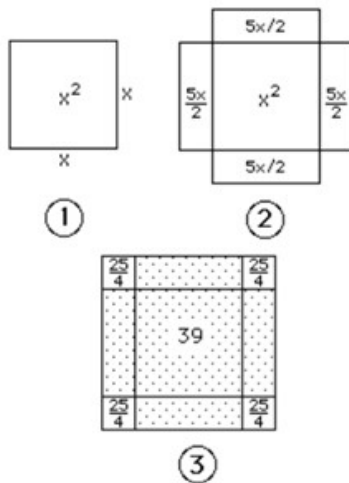
$$x^2 + 10x = 39 \quad x=?$$

Algoritmul

1. x^2 = aria pătratului de latură x (figura 1)
2. La x^2 trebuie să adăugăm $10x$ și acest lucru se face adăugând patru dreptunghiuri fiecare cu lățime $10/4$ adică $5/2$ și lungime x (figura 2)
3. Completăm aria adăugând 4 pătrate mici de arie $25/4$ ($\frac{5}{2} * \frac{5}{2} = \frac{25}{4}$).
4. Atunci aria: $4 * \frac{25}{4} + 39 = 25 + 39 = 64$, din $x^2 = 64$, rezultă latura = 8
dar latura pătratului este: $2 * \frac{5}{2} + x = 8$, rezultă
 $5 * x = 8, x = 8 - 5 = 3$
 $x = 3$

$x^2 + 10x = 39$ $x = 3$ ($9 + 30 = 39$ adevărat)

al-Khwarizmi completes the square



$x^2 + 12x = 28$????

Algoritmul

1. Știm că: $x^2 + 10x = 39$
2. $x^2 + 10x + 21 = 10x + 10x$
3. $39 + 21 = 20x$
4. $60 = 20x$
5. $x = 3$



ALGORITHM



reprezintă o metodă sau o procedură de calcul, alcătuită din pași elementari necesari, pentru rezolvarea unei probleme.

...



Ce presupune un algoritm?

Am inteles enuntul unei probleme
am identificat contextul, starea initiala si starea finala (i.e. cerintele, prelucrarile necesare si rezultatul asteptat)

Am imaginat o solutie a problemei
am identificat elementele auxiliare necesare solutionarii problemei
am identificat pasii care conduc la obtinerea solutiei (algoritmul)

Am pus in aplicare solutia problemei
am urmat pasii mentionati mai sus

Cât ne “costă” ?

La aceasta intrebare vom raspunde peste câteva cursuri. Deocamdata nu ne gandim la asta! **Important este sa obtinem un rezultat corect!**

Ce presupune un algoritm?

o soluție a unei probleme presupune următoarele:

- structurarea datelor problemei;
- elaborarea unui algoritm de rezolvare;
- o analiză care să asigure corectitudinea și eficiența algoritmului folosit.

Rezolvarea problemelor



Problema

ansamblu de intrebari referitoare la anumite entitati care reprezinta universul problemei



Enuntul

descrierea proprietatilor entitatilor si a relatiei dintre datele de intrare si solutia problemei



Metoda

procedura de construire a solutiei pornind de la datele de intrare

Date de
intrare

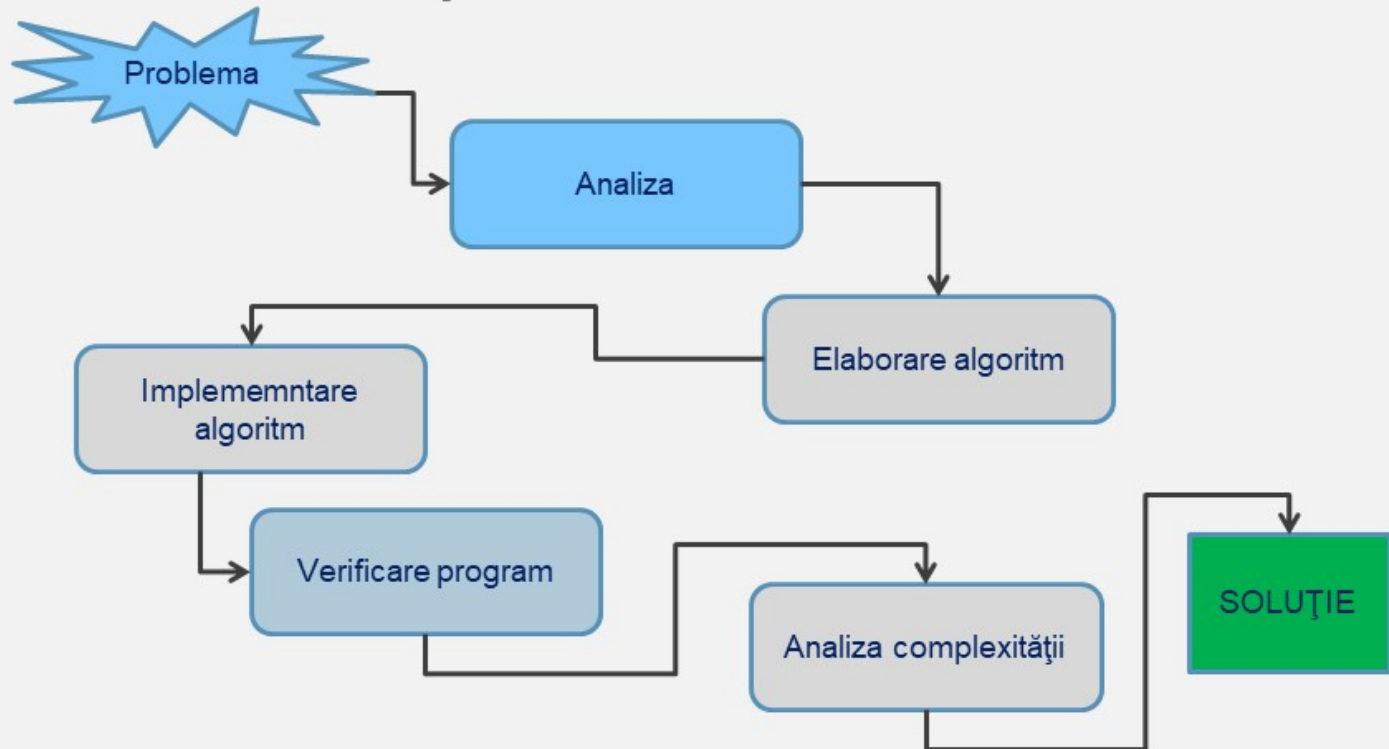


Metoda de
rezolvare



Rezultat

Etapele rezolvarii unei probleme



Exemplu

Fie a și b două numere naturale nenule. Să se determine numărul c care are următoarele proprietăți:

- c divide pe a și pe b (c este divizor comun al lui a și al lui b)
- c este mai mare decât orice alt divizor al lui a și b

Universul problemei:

multimea numerelor naturale (a și b reprezintă datele de intrare, c reprezintă rezultatul)

Enunțul problemei: (relația dintre datele de intrare și rezultat):

c este cel mai mare divizor comun al lui a și b

Observatie:

Aceasta problema face parte din clasa celor care calculeaza valoarea unei functii (care asociaza unei perechi de numere naturale valoarea celui mai mare divizor comun)

Un alt tip de probleme sunt cele care cer sa se verifice daca datele de intrare satisfac o anumita proprietate. Acestea sunt denumite probleme de decizie.

Exemplu: sa se verifice daca un numar natural este prim sau nu

In ambele cazuri solutia poate fi obtinuta folosind un calculator doar daca exista o metoda care sa furnizeze rezultatul dupa un numar finit de prelucrari ... o astfel de metoda este un algoritm



02

Definire algoritmo



Exista diferite definitii ...

Algorithm =

o descriere pas cu pas a metodei de
rezolvare a unei probleme

Algorithm =

o succesiune finita de operatii care aplicate
datelor de intrare ale unei probleme conduc la
solutie

Algorithm =

reteta de rezolvare a unei probleme

Exemple

Algoritmi in viata de zi cu zi:

Utilizarea unui telefon, bancomat, automat pentru cafea etc

Algoritmi specifici matematicii:

Algoritmul lui Euclid

(este considerat primul algoritm)

Determinarea celui mai mare divizor comun a doua numere

Algoritmul lui Eratostene

Generarea numerelor prime

Algoritmul lui Horner

Calculul valorii unui polinom

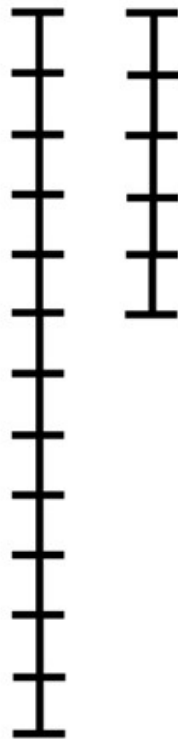


Euclid

pentru numerele 252 și 105

Barele reprezintă unitățile de 21 (CMMDC).

La fiecare pas, numărul mai mic este scăzut din cel mai mare, până când unul dintre numere ajunge să fie zero. Celălalt este CMMDC.



De la problemă la algoritm

Problema:

- **Datele problemei**
 - a, b - nr.naturale
- **Metoda de rezolvare**
 - Imparte a la b si retine restul
 - Imparte b la rest si retine noul rest
 - continua impartirile pana se ajunge la un rest nul
 - Ultimul rest nenul reprezinta rezultatul

Algoritm:

- **Variable** = obiecte abstracte ce corespund datelor problemei
 - deimpartit, impartitor, rest
- **Secventa de prelucrari**
 1. Atribuie deimpartitului valoarea lui a si impartitorului valoarea lui b
 2. Calculeaza restul impartirii deimpartitului la impartitor
 3. Atribuie deimpartitului valoarea impartitorului si impartitorului valoarea restului anterior
 4. Daca restul e nenul reia de la etapa 2

De la algoritm la program

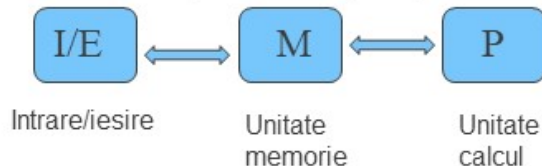
Algoritm:

- **Variabile** = obiecte abstracte ce corespund datelor problemei
 - deimpartit, impartitor, rest
- **Secventa de prelucrari**
 1. Atribuire deimpartitului valoarea lui a si impartitorului valoarea lui b
 2. Calculeaza restul impartirii deimpartitului la impartitor
 3. Atribuire deimpartitului valoarea impartitorului si impartitorului valoarea restului anterior
 4. Daca restul e nenul reia de la etapa 2

Program:

- **Variabile** = obiecte abstracte ce corespund datelor problemei
 - Fiecare variabila are asociata o zona in memoria calculatorului
- **Secventa de instructiuni**
 - Fiecare instructiune corespunde unei prelucrari elementare care poate fi executata de catre calculator

Model (extrem de) simplificat



De la algoritm la program

Algoritm:

- **Variabile** = obiecte abstracte ce corespund datelor problemei
 - deimpartit, impartitor, rest
- **Secventa de prelucrari**
 1. Atribuire deimpartitului valoarea lui a si impartitorului valoarea lui b
 2. Calculeaza restul impartirii deimpartitului la impartitor
 3. Atribuire deimpartitului valoarea impartitorului si impartitorului valoarea restului anterior
 4. Daca restul e nenul reia de la etapa 2

```
int euclid(int a, int b)
{
    int c;
    while (b) {
        c = a % b;
        a = b;
        b = c;
    }
    return a;
}
```

Eratostene

Generarea
primelor 64 de numere prime

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

Eratostene

Generare
numere prime

```
#include<stdio.h>
#define N 100
int main()
{
    int num[N], i, j, n=64;
    for(i = 0; i < n; i++)
        num[i] = i + 1;
    for(i = 1; (num[i] * num[i]) <= n; i++)
        if(num[i] != 0)
            for(j = num[i] * num[i]; j <= n; j += num[i])
                num[j - 1] = 0;
    printf("Algoritmul - Ciurul lui Eratostene -\n");
    printf("Numerele prime de la 2 la %d sunt:\n", n);
    for(i = 1; i < n; i++)
    {
        if(num[i] != 0)
            printf("%d ", num[i]);
    }
    printf("\n");

    return 0;
}
```

Algoritmul - Ciurul lui Eratostene -
Numerele prime de la 2 la 64 sunt:

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61

Horner

Calculul valorii polinomiale

$$a_0 + a_1x + a_2x^2 + a_3x^3 + \dots + a_nx^n \\ = a_0 + x \left(a_1 + x \left(a_2 + x \left(a_3 + \dots + x(a_{n-1} + x a_n) \dots \right) \right) \right).$$

```
#include <iostream>
using namespace std;

int horner(int poly[], int n, int x)
{
    int result = poly[0];
    for (int i=1; i < n; i++)
        result = result*x + poly[i];
    return result;
}

int main()
{
    // 2x^3 - 6x^2 + 2x - 1 for x = 3
    // 2*27 - 6*9 + 2*3 - 1 = 54 - 54 + 6 - 1 = 5
    int poly[] = {2, -6, 2, -1};
    int x = 3;
    int n = sizeof(poly)/sizeof(poly[0]);
    cout << "Valoarea polinomului este: " << horner(poly, n, x);
    return 0;
}
```

Valoarea polinomului este: 5



03

Proprietati algoritm



Proprietățile algoritmilor

- **Corectitudinea**
- **Caracterul univoc sau determinist**
- **Generalitatea**
- **Claritatea**
- **Verificabilitatea**
- **Optimalitatea**
- **Finitudinea**
- **Eficiența**
- **Existența unei intrări** (datele de prelucrat)
- **Existența unei ieșiri** (rezultatele)

Proprietățile algoritmilor

o soluție a unei probleme presupune următoarele:

- structurarea datelor problemei;
- elaborarea unui algoritm de rezolvare;
- o analiză care să asigure corectitudinea și eficiența algoritmului folosit.



Proprietati

- Generalitate
- Finitudine
- Neambiguitate
- Eficienta



Generalitate

Un algoritm trebuie sa functioneze corect pentru toate instantele de date de intrare nu doar pentru instante particulare

Exemplu:

Sa consideram problema sortarii crescatoare a unui sir de valori numerice.

De exemplu:

(2,1,4,3,5)

date intrare

(1,2,3,4,5)

rezultat



Generalitate

Metoda:

Descriere:

Pas 1: 2 ↔ 1 4 3 5

-Compara primele doua elemente; daca nu sunt in ordinea dorita se interschimba

Pas 2: 1 2 4 3 5

-Compara al doilea cu al treilea si aplica aceeasi strategie

Pas 3: 1 2 4 ↔ 3 5

Pas 4: 1 2 3 4 5

.....
- Continua pana ultimele doua elemente au fost comparate

Secvența a fost ordonată

Generalitate

Este algoritmul suficient de general ? Asigura ordonarea crescatoare a oricarui sir de valori ?

- Raspuns: NU
- Contra-exemplu:**

3 2 1 4 5
2 3 1 4 5
2 1 3 4 5
2 1 3 4 5

In acest caz metoda nu functioneaza deci nu poate fi considerata un algoritm general de ordonare ... e necesara reluarea procesului de parcurgere a secventei

Proprietati

- Generalitate
- Finitudine
- Neambiguitate
- Eficienta



Finitudine

Un algoritm trebuie sa se termine dupa un numar finit de prelucrari

Exemplu:

Pas1: Asignează 1 lui x ;

Pas2: Adaugă 2 la x ;

Pas3: Daca $x=10$ atunci STOP;
altfel se reia de la Pasul 2

Cum se execută acest algoritm ?

Finitudine

Execuția algoritmului anterior

➡ **Pas1:** Asigneaza 1 lui x; $x=1$

➡ **Pas2:** Adauga 2 la x; $x=3$ $x=5$ $x=7$ $x=9$ $x=11$

➡ **Pas3:** Daca $x=10$ atunci STOP;

➡ altfel afiseaza x; se reia de la Pasul 2

Ce putem spune despre acest algoritm ?

Finitudine

Execuția algoritmului anterior

➡ **Pas1:** Asigneaza 1 lui x; $x=1$

➡ **Pas2:** Adauga 2 la x; $x=3$ $x=5$ $x=7$ $x=9$ $x=11$

➡ **Pas3:** Daca $x \geq 10$ atunci STOP;

➡ altfel afiseaza x; se reia de la Pasul 2

Algoritm corectat

Proprietati

- Generalitate
- Finitudine
- **Neambiguitate**
- Eficienta



Neambiguitate

Operatiile dintr-un algoritm trebuie definite in mod riguros:

- La executia fiecarui pas trebuie specificat clar ce trebuie executat si care va fi urmatorul pas

Exemplu:

Pas 1: Atribuire 0 lui x

Pas 2: **Fie** incrementeaza x cu 1 **fie** decrementeaza x cu 1

Pas 3: Daca $x \in [-2, 2]$ atunci se reia de la Pasul 2; altfel Stop.

Atat timp cat nu este specificat un criteriu clar in baza caruia sa se decida daca x este incrementat sau decrementat secventa de mai sus nu poate fi considerata algoritm

Neambiguitate

Modificam algoritmul anterior dupa cum urmeaza:

Pas 1: Atribuire 0 lui x

Pas 2: **Arunca o moneda**

Pas 3: Daca se obtine cap
atunci incrementeaza x cu 1
altfel decrementeaza x cu 1

Pas 4: Daca $x \in [-2, 2]$ atunci se reia de la Pasul 2, altfel Stop.

Algoritmul poate fi executat dar ... la rulari diferite poate conduce la rezultate diferite

Acesta este un exemplu de **algoritm aleator**

Proprietati

- Generalitate
- Finitudine
- Neambiguitate
- **Eficienta**



Eficiența

Un algoritm trebuie sa foloseasca un volum rezonabil de resurse de calcul:
memorie si **timp de calcul**

Finitudinea nu e suficienta daca timpul necesar obtinerii unui rezultat este prea mare

Exemplu:

Sa consideram un dictionar continand 50000 de cuvinte.

Sa se gaseasca un algoritm care pentru un cuvant dat ca intrare determina toate anagramele cuvantului care sunt prezente in dictionar.

Exemplu de anagrama: cort->troc

Eficiența

Prima abordare:

Pas 1: genereaza toate anagramele cuvantului

Pas 2: pentru fiecare anagrama a cuvantului se verifica daca este prezenta in dictionar (folosind de algoritm eficient, de exemplu cautare binara)

A doua abordare:

Pas 1: se sorteaza literele cuvantului initial

Pas 2: Pentru fiecare cuvânt din dictionar având m litere:

- Se sorteaza literele cuvantului
- Se compara versiunile sortate ale cuvantului initial si ale fiecarui cuvânt din dictionar

Care varianta este mai buna (in raport cu numarul de operatii efectuate) ?

Eficiența

Prima abordare:

Pas 1: genereaza toate anagramele cuvantului

Pas 2: pentru fiecare anagrama a cuvantului se verifica daca este prezenta in dictionar (folosind de algoritm eficient, de exemplu cautare binara)

Sa consideram ca:

Dictionarul contine n cuvinte

Cuvantul analizat contine m litere

O estimare a numarului de comparatii la nivel de litere:

- Numarul de anagrame: $m!$
- Numarul de comparatii pentru fiecare anagrama: $\log_2 n$
- Numarul de litere comparate pentru fiecare anagrama: m

$$m! * m * \log_2 n$$

Eficiența

A doua abordare:

Pas 1: se sorteaza literele
cuvantului initial

Pas 2: Pentru fiecare cuvant din
dictionar avand m litere:

- Se sorteaza literele
cuvantului
- Se compara versiunile
sortate ale cuvantului initial si ale
fiecarui cuvant din dictionar

Estimarea numarului de comparatii:

- Sortarea cuvantului initial necesita circa m^2 comparatii

- Cautarea secventiala si sortarea fiecarui
cuvant necesita circa nm^2 comparatii

- Compararea versiunilor sortate necesita
cel mult nm comparatii

$$n m^2 + nm + m^2$$

Eficiența

Care abordare este mai buna ?

Prima abordare

$$m! m \log_2 n$$

Exemplu: $m=11$ (cuvantul algoritmica)
 $n=50000$ (numarul de cuvinte din dictionar)

$$2 * 10^9$$

o comparatie = $1\text{ms} = 10^{-3}\text{ s}$

555.5 ore

A doua abordare

$$n m^2 + n m + m^2$$

$$6 * 10^6$$

1.66 ore



04

Descrierea algoritmilor



Descrierea algoritmilor

Metodele de rezolvare a problemelor sunt de regula descrise într-un **limbaj matematic**

Limbajul matematic nu este întotdeauna adecvat intrucat:

- Operatii considerate elementare din punct de vedere matematic nu corespund unor prelucrari elementare cand sunt executate pe un calculator.

Exemplu: calculul unei sume, evaluarea unui polinom etc

Descriere matematica

$$\sum_{i=1}^n i = 1 + 2 + \dots + n$$

Descriere algoritmica

?

Descrierea algoritmilor

Exista cel putin doua modalitati:

Scheme logice:

- Descrieri grafice ale fluxului de prelucrari din algoritm
- Sunt destul de rar utilizate la ora actuala
- Totusi pot fi utile in descrierea structurii generale a unei aplicatii

Pseudocod:

- Limbaj artificial bazat pe
 - vocabular (set de cuvinte cheie)
 - sintaxa (set de reguli de construire a frazelor limbajului)
- Nu e la fel de restrictiv ca un limbaj de programare



Mulumesc

pentru atenție!

dorin.iordache@365.univ-ovidius.ro