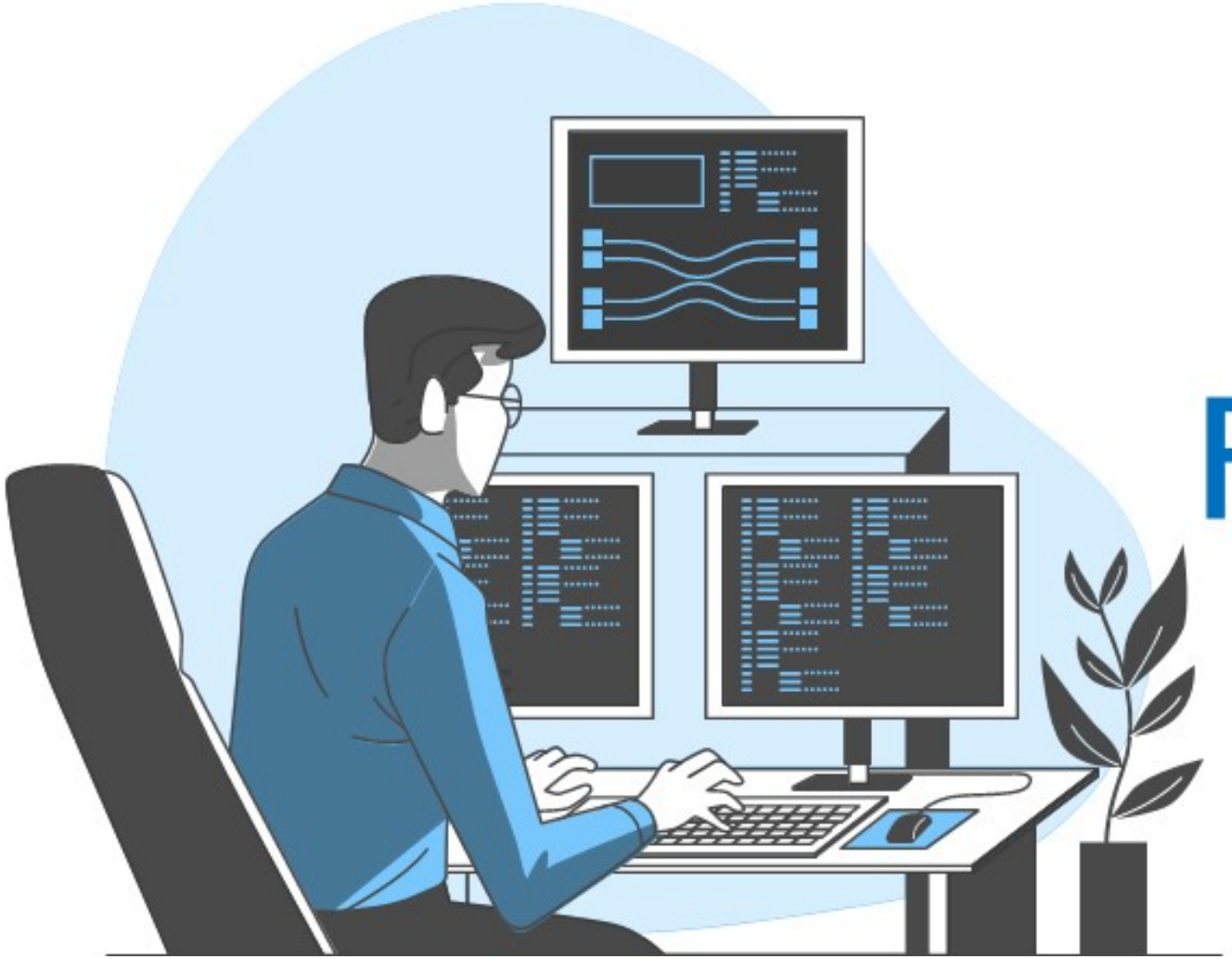
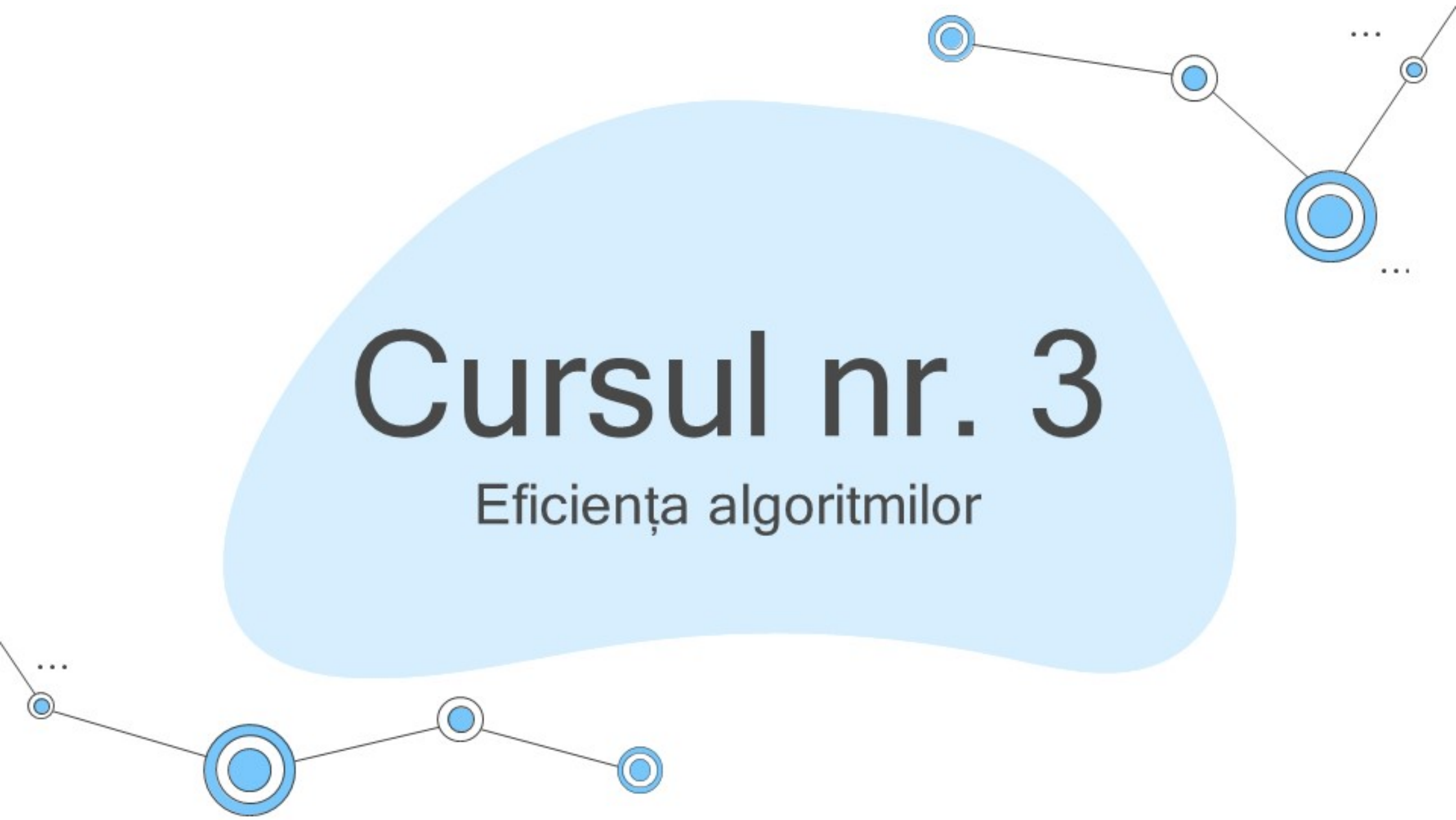


Algoritmi Fundamentali

Lector dr.
Dorin IORDACHE



Cursul nr. 3

Eficiența algoritmilor

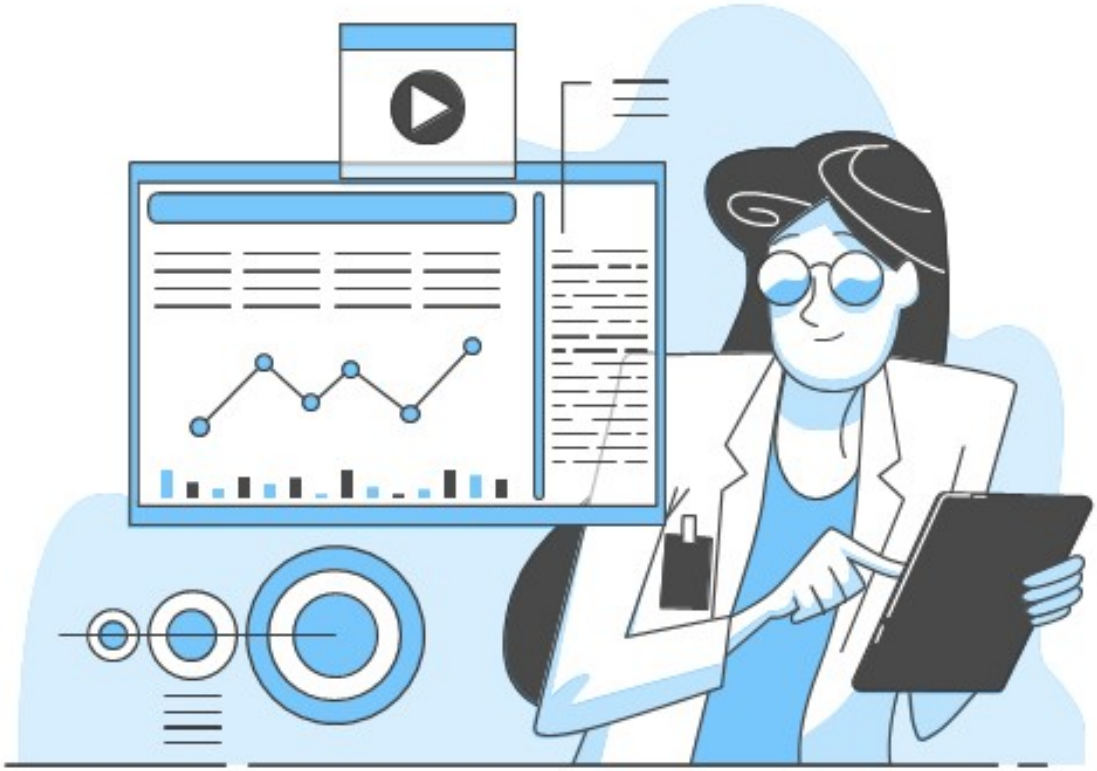
Agenda

01 Generalitati

02 Masurarea eficienței algoritmilor

03 Exemple

04 Cazuri favorabile și nefavorabile



01

Generalitati

ALGORITM

Analiza eficientei algoritmilor inseamna:

estimarea volumului de resurse de calcul necesare executiei algoritmilor

Observatie: uneori se foloseste termenul de analiza a complexitatii

Utilitate: analiza eficientei este utila pentru a compara algoritmii intre ei si pentru a obtine informatii privind resursele de calcul necesare pentru executia algoritmilor

Analiza consta in

Resurse de calcul:

Spatiu de memorie = spatiu necesar stocarii datelor prelucrate de catre algoritm

Timp de executie = timp necesar executiei prelucrarilor din cadrul algoritmului

Algoritm eficient: algoritm care necesita un volum rezonabil de resurse de calcul

Daca un algoritm utilizeaza mai putine resurse de calcul decat un alt algoritm atunci este considerat mai eficient

Ipoteze

Componentele utilizării **spațiului**/memoriei:

1. spațiu de instrucțiuni
   Afectat de: compilator, opțiunile compilatorului, computerul țintă (cpu)
2. spațiu de date
   Afectat de: dimensiunea datelor/memoria alocată dinamic, variabilele programului statice,
3. spațiu de stivă în timpul rulării
   Afectat de: compilator, apeluri de funcții de rulare și recursivitate, variabile locale, parametri

Ipoteze

Durata reală de funcționare depinde de mulți factori:

1. Viteza computerului:
   CPU (nu doar viteza de ceas), I/O etc.
2. Compilatorul, opțiunile compilatorului.
3. Cantitatea de date - ex. caută o listă lungă sau scurtă.
4. Datele reale - ex. în căutarea secvenţială dacă numele este primul sau ultimul.

Analiza consta in

Exista doua tipuri de eficienta

- Eficienta in raport cu spatiul de memorie = se refera la spatiul de memorie necesar algoritmului
- Eficienta in raport cu timpul de executie = se refera la timpul necesar executiei prelucrarilor din algoritm

Ambele tipuri de analiza a eficientei algoritmilor se bazeaza pe urmatoarea ipoteza:

volumul resurselor de calcul necesare depinde de volumul datelor de intrare = dimensiunea problemei

Scopul analizei eficientei este sa raspunda la intrebarea:

Cum depinde volumul resurselor necesare de dimensiunea problemei ?

Dimensiunea problemei ...

... de regula foarte simplu pornind de la enuntul problemei si de la proprietatile datelor de intrare

Dimensiunea problemei = volumul de memorie necesar pentru a stoca toate datele de intrare ale problemei

Dimensiunea problemei este exprimata in una dintre urmatoarele variante:

- numarul de componente (valori reale, valori intregi, caractere etc) ale datelor de intrare
- numarul de biti necesari stocarii datelor de intrare

Exemple

1. Determinarea minimului unui tablou x[1..n]

   Dimensiunea problemei: n

2. Calculul valorii unui polinom de ordin n

   Dimensiunea problemei: n

   Obs: numarul coeficientilor este de fapt n+1

3. Calculul sumei a doua matrici cu m linii si n coloane

   Dimensiunea problemei: (m,n) sau mn

4. Verificarea primalitatii unui numar n

   Dimensiunea problemei: n sau $\log_2 n$

   Obs: numarul de biti necesari stocarii valorii n este de fapt $[\log_2 n]+1$ ([..] reprezinta partea intreaga inferioara)

02

Masurarea eficientei algoritmilor

Masurarea eficientei presupune

Stabilirea:

- unui model de calcul
- unei unități de măsură a timpului de execuție

Convenții

Model de calcul: masina cu acces aleator (Random Access Machine = RAM)

Caracteristici (ipoteze simplificatoare):

Toate prelucrarile sunt executate secvential
(nu exista paralelism in executia algoritmului)

Timpul de executie al unei prelucrari elementare nu depinde de valorile operanzilor
(timpul de executie pentru a calcula 1+2 nu difera de timpul de executie pentru 12433+4567)

Timpul necesar accesarii datelor nu depinde de locatia datelor in memorie (nu este diferenta intre timpul necesar prelucrarii primului element al unui tablou si cel al prelucrarii ultimului element)

Convenții

Unitate de masura = timpul necesar executiei unei prelucrari elementare (prelucrare de baza)

Operatii elementare (de baza):

- Atribuire
- Operatii aritmetice
- Comparatii
- Operatii logice

Timp de executie al algoritmului = numarul de operatii elementare executate

Estimarea timpului de executie exprima dependenta dintre numarul de operatii elementare executate si dimensiunea problemei

Masurarea eficienței

Tabel de costuri:

Operatie	Cost	Nr. repetari
1	c_1	n_1
2	c_2	n_2
3	c_3	n_3
....	....	.....
o	c_o	n_o

Timp de executie:

$$T(n) = \sum_{op=1}^{o} Cost_{op} * Nr.Repetariop$$

03

Exemple

Exemplul 1

Preconditii: n>=1

Postconditii: S=1+2+…+n

Dim. problema: n

Algoritm: Sum(n)

```
1:  S←0
2:  i ← 0
3:  WHILE i<n DO
4:          i ← i+1
5:          S ← S+i
6:  ENDWHILE
7: RETURN S
```

Tabel de costuri:

Operatie	Cost	Nr. repetari
1	c1	1
2	c2	1
3	c3	n+1
4	c4	n
5	c5	n

Timp de executie:

$T(n)=(c3+c4+c5)n+(c1+c2+c3)= a*n +b$

Obs: unele dintre costuri sunt identice (c1=c2), iar altele pot fi considerate diferite (c5>c1)

Exemplul 1

Observatii:
Varianta cea mai simpla este sa se considere toate prelucrarile simple ca avand acelasi cost
Daca in exemplul anterior se considera ca toate operatiile elementare au cost unitar atunci se obtine urmatoarea estimare pentru timpul de executie: $T(n)=3(n+1)$

Constantele ce intervin in expresia timpului de executie nu sunt foarte importante. Elementul important este faptul ca timpul de executie depinde liniar de dimensiunea problemei.
Algoritmul anterior este echivalent cu:

```
S ← 0
FOR i ← 1,n DO
        S ← S+i
ENDFOR
```

Se observa ca gestiunea contorului ciclului FOR implica executia a 2(n+1) operatii; celelalte (n+1) operatii corespund calculului sumei (initializarea lui S si actualizarea de la fiecare repetare a corpului ciclului)

Exemplul 2

Preconditii: x[1..n], n>=1 Postconditii: m=min(x[1..n])

Dimensiunea problemei: n

Algoritm:

```
Minim(x[1..n])
1:  m ← x[1]
2:   FOR i ← 2,n DO
3:       IF x[i]<m THEN
4:              m ← x[i]
5:       ENDIF
6:   ENDFOR
7:RETURN  m
```

Tabel de costuri:

Op.	Cost	Rep.	Total
1	1	1	1
2	2n	1	2n
3	1	n-1	n-1
4	1	t(n)	t(n)

T(n)=3n+t(n)

Obs: Timpul de executie depinde nu doar de dimensiunea pb. ci si de proprietatile datelor de intrare

Exemplul 2

Daca timpul de executie depinde de proprietatile datelor de intrare atunci trebuie analizate cel putin cazurile extreme:

Cel mai favorabil caz : x[1]<=x[i], i=1..n => t(n)=0 => T(n)=3n

Cel mai defavorabil caz x[1]>x[2]>…>x[n])=> t(n)=n-1 => T(n)=4n-1

Rezulta ca: 3n<=T(n)<=4n-1

Atat limita inferioara cat si cea superioara depind liniar de dimesiunea problemei

Varianta de analiza ce ia in calcul doar operatia dominanta, respectiv comparatia, ce conduce la

T(n) =n-1

Algoritm:

```
Minim(x[1..n])
1:  m ← x[1]
2:   FOR i ← 2,n DO
3:       IF x[i]<m THEN
4:                m ← x[i]
5:       ENDIF
6:   ENDFOR
7: RETURN  m
```

Exemplul 3

Preconditii: x[1..n], n>=1, v o valoare
Postconditii: variabila logica "gasit" contine valoarea de adevar a afirmatiei "*valoarea v este in tabloul x[1..n]*"
Dimensiunea problemei: n

Algoritm (cautare secventiala):

```
caut(x[1..n],v)
1: gasit ← False
2: i ← 1
3: WHILE (gasit=False) AND (i<=n) DO
4:     IF x[i]=v THEN          //t1(n)
5:         gasit ← True        //t2(n)
       ELSE
6:         i ← i+1             //t3(n)
7:     ENDIF
8: ENDWHILE
9: RETURN gasit
```

Tabel de costuri:

Op.	Cost	Rep.	Total
1	1	1	1
2	1	1	1
3	1	t(n)+1	t(n)+1
4	1	t(n)	t(n)
5	1	1	1
6	1	t(n)-1	t(n)-1

$T(n) = 1 + 1 + (t(n) + 1) + t(n) + 1 + (t(n) - 1) = 3 * t(n)+3.$

Exemplul 3

Timpul de executie depinde de proprietatile tabloului x[1..n].
Caz 1: valoarea v apartine tabloului (fie k cel mai mic indice cu proprietatea ca x[k]= v)
Caz 2: valoarea v nu se afla in tablou

$$t1(n)=\begin{cases} k & \text{daca v este in x[1..n]} \\ n & \text{daca v nu este in x[1..n]} \end{cases}$$

$$t2(n)=\begin{cases} 1 & \text{daca v este in x[1..n]} \\ 0 & \text{daca v nu este in x[1..n]} \end{cases}$$

$$t3(n)=\begin{cases} k-1 & \text{daca v este in x[1..n]} \\ n & \text{daca v nu este in x[1..n]} \end{cases}$$

Algoritm (cautare secventiala):

```
caut(x[1..n],v)
1:  gasit ← False
2:  i ← 1
3:  WHILE (gasit=False) AND (i<=n) DO
4:      IF x[i]=v THEN          //t1(n)
5:          gasit ← True        //t2(n)
        ELSE
6:          i ← i+1             //t3(n)
7:      ENDIF
8:  ENDWHILE
9: RETURN gasit
```

Exemplul 3

Cel mai favorabil caz: x[1]=v

t1(n)=1, t2(n)=1, t3(n)=0

T(n)= 6

Cel mai defavorabil caz: v nu se afla in x[1..n]

t1(n)=n, t2(n)=0, t3(n)=n

T(n)=3n+3

Marginile (inferioara si superioara) ale timpului de executie sunt:

6<= T(n) <= 3(n+1)

Obs: limita inferioara este constanta iar cea superioara depinde liniar de dimensiunea pb.

$$t1(n)=\begin{cases} k & \text{daca v este in x[1..n]} \\ n & \text{daca v nu este in x[1..n]} \end{cases}$$

$$t2(n)=\begin{cases} 1 & \text{daca v este in x[1..n]} \\ 0 & \text{daca v nu este in x[1..n]} \end{cases}$$

$$t3(n)=\begin{cases} k-1 & \text{daca v este in x[1..n]} \\ n & \text{daca v nu este in x[1..n]} \end{cases}$$

Exemplul 4

Preconditii: x[1..n], n>=1, v o valoare
Postconditii: variabila logica "gasit" contine valoarea de adevar a afirmatiei "*valoarea v este in tabloul x[1..n]*"
Dimensiunea problemei: n

```
Caut2(x[1..n],v)
1: i ← 1
2: while x[i]<>v and i<n do
3:     i ← i+1
   endwhile
4: if x[i]=v then
5:     gasit ← true
   else
6:     gasit ← false
endif
7: return gasit
```

Tabel de costuri:

Op.	Cost	Rep.	Total
1	1	1	1
2	1	t(n)+1	t(n)+1
3	1	t(n)	t(n)
4	1	1	1
5	1	1	1
6	1	1	1

$T(n) = 1 + (t(n) + 1) + t(n) + 1 + 1 + 1 = 2 * t(n) + 5.$

Discutie exemplu 3 si 4

Algoritm (cautare secventiala):

```
caut(x[1..n],v)
1:  gasit ← False
2:  i ← 1
3:  WHILE (gasit=False) AND (i<=n) DO
4:      IF x[i]=v THEN          //t1(n)
5:          gasit ← True        //t2(n)
        ELSE
6:          i ← i+1             //t3(n)
7:      ENDIF
8:  ENDWHILE
9: RETURN gasit
```

$T(n) = 3 * t(n)+3.$

```
Caut2(x[1..n],v)
1:  i ← 1
2:  while x[i]<>v and i<n do
3:      i ← i+1
    endwhile
4: if x[i]=v then
5:      gasit ← true
   else
6:      gasit ← false
endif
7: return gasit
```

$T(n) = 2 * t(n)+5.$

Exemplul 4

Pentru anumite probleme cazul cel mai favorabil si cel mai defavorabil sunt cazuri rare (exceptii)

Astfel … timpul de executie in cazul cel mai favorabil respectiv in cazul cel mai defavorabil nu furnizeaza suficienta informatie

In aceste cazuri se efectueaza o alta analiza… analiza cazului mediu

Scopul acestei analize este sa furnizeze informatii privind comportarea algoritmului in cazul unor date de intrare arbitrare (care nu corespund nici celui mai favorabil nici celui mai defavorabil caz)

04

Cazuri favorabile si nefavorabile

Caz favorabil și nefavorabil

Analiza în cazul cel mai favorabil:

- furnizezaza o margine inferioara pentru timpul de executie
- Permite identificarea algoritmilor ineficienti (daca un algoritm are un cost ridicat chiar si in cel mai favorabil caz atunci el nu reprezinta o solutie acceptabila)

Analiza in cazul cel mai nefavorabil:

- furnizezaza cel amai mare tip de executie executie in raport cu toate datele de intrare de dimensiune n (reprezinta o margine superioara a timpului de executie)
- Marginea superioara a timpului de executie este mai importanta decat marginea inferioară

Caz mediu

Aceasta analiza se bazeaza pe cunoasterea distributiei de probabilitate a datelor de intrare

Aceasta inseamna cunoasterea (estimarea) probabilitatii de aparitie a fiecareia dintre instantele posibile ale datelor de intrare (cat de frecvent apare fiecare dintre posibilele valori ale datelor de intrare)

Timpul mediu de executie este valoarea medie (in sens statistic) a timpilor de executie corespunzatori diferitelor instante ale datelor de intrare

Caz mediu

Ipoteze. Presupunem ca sunt satisfacute urmatoarele ipoteze:

- datele de intrare pot fi grupate in clase astfel incat timpul de executie corespunzator datelor din aceeasi clasa este acelasi
- sunt m=M(n) clase cu date de intrare
- Probabilitatea de aparitie a unei date din clasa k este P_k
- Timpul de executie al algoritmului pentru date de intrare apartinand clasei k este $T_k(n)$

In aceste ipoteze timpul mediu de executie este:

$T_a(n)= P_1T_1(n)+P_2T_2(n)+\ldots+P_mT_m(n)$

Observatie: daca toate clasele de date au aceeasi probabilitate atunci timpul mediu de executie este:

$$T_a(n)=(T_1(n)+T_2(n)+\ldots+T_m(n))/m$$

Caz mediu - exemplu

Exemplu: cautare secventiala (operatie dominanta: comparatia)

Ipoteze privind distributia de probabilitate a datelor de intrare:

- Probabilitatea ca valoarea v sa se afle in tablou: p
  - valoarea v apare cu aceeasi probabilitate in fiecare dintre pozitiile din tablou
  - probabilitatea ca valoarea v sa se afle pe pozitia k este 1/n
- Probabilitatea ca v sa nu se afle in tablou: 1-p

$T_a(n)=p(1+2+\ldots+n)/n+(1-p)n=p(n+1)/2+(1-p)n=(1-p/2)n+p/2$

Daca $p=0.5$ se obtine $T_a(n)=3/4\ n+1/4$

Concluzie: timpul mediu de executie al algoritmului de cautare secventiala depinde liniar de dimensiunea datelor de intrare

Caz mediu - analiza

Exemplu: cautare secventiala (varianta cu santinela sau fanion)

Idee de baza:

- Tabloul este extins cu un element suplimentar (n+1) a carui valoare este v
- Tabloul este parcurs pana la intalnirea lui v (valoarea va fi gasita in cel mai rau caz pe pozitia n+1 – in acest caz rezultatul algoritmului este : tabloul x[1..n] nu contine valoarea)

Caz mediu - analiza

Algoritm:
Cautare_fanion(x[1..n],v)
i:=1
WHILE x[i]<>v DO
    i:=i+1
ENDWHILE
RETURN i
Operatie dominanta: comparatia

Probabilitatea ca valoarea v sa fie pe pozitia k din {1,2,…,n+1} este 1/(n+1)

Timpul mediu de executie:

$T_a(n)=(1+2+\ldots+(n+1))/(n+1)=(n+2)/2$

Observatie:

- Schimband ipoteza privind distributia datelor de intrare valoarea timpului mediu se modifica (in cazul cautarii secventiale dependenta de dimensiunea problemei ramane liniara

Timpul mediu de executie NU este in mod necesar media aritmetica dintre timpii de executie corespunzatori cazurilor extreme (cel mai favorabil si cel mai defavorabil)

Rezumat

Pas 1: Identificarea dimensiunii problemei

Pas 2: Identificarea operatiei dominante

Pas 3: Determinarea numarului de executii ale operatiei dominante

Pas 4. Daca numarul de executii ale operatiei dominante depinde de proprietatile datelor de intrare atunci se analizeaza:

- Cel mai favorabil caz
- Cel mai nefavorabil caz
- Cazul mediu

Rezumat

Tabel de costuri:

Operatie	Cost	Nr. repetari
1	c_1	n_1
2	c_2	n_2
3	c_3	n_3
....	....	.....
o	c_o	n_o

Timp de executie:

$$T(n) = \sum_{op=1}^{o} Cost_{op} * Nr.Repetariop$$

Mulțumesc

pentru atenție!

dorin.iordache@365.univ-ovidius.ro