TEHNICI DE ELABORARE A ALGORITMILOR

METODA TRIERII

Metoda Trierii – schema generală

- Fie `P` o problemă, soluția căreia se află printre elementele mulțimii `S` cu un număr finit de elemente.

$$S = \{s_1, s_2, s_3, ..., s_i, ..., s_n\}$$

- În cele mai simple cazuri, elementele s_i, $s_i \in S$, pot fi reprezentate prin valori care aparțin unor *tipuri ordinale* de date: `int`, `bool`, `char`, `enum`.
- În problemele mai complexe, elementele pot fi reprezentate prin date de tip *tablou unidimensional/bidimensional*, `string` sau `struct`.
- În majoritatea problemelor *soluțiile posibile* $S = \{s_1, s_2, ..., s_n\}$ *nu sunt indicate explicit în enunțul problemei* și *elaborarea algoritmilor* pentru calcularea lor *cade în sarcina programatorului.*
- *Soluția problemei* se determină prin *analiza fiecărui element* s_i din mulțimea S.
- Algoritmii bazați pe *metoda trierii* au o *structură simplă*, însă *complexitatea temporală a lor este mare*.

Metoda Trierii – schema generală

Schema generală a unui *algoritm bazat pe metoda trierii* poate fi redată cu ajutorul unei instrucțiuni repetitive:

```
1 for i in range (1,n):
2       if (SolutiePosibila(Si))
3           PrelucrareSolutie(Si)
```

- `SoluțiePosibilă` este o *funcție booleană* care returnează valoarea `1/true`, dacă elementul s_i *satisface condițiile problemei* și `false` *în caz contrar*;
- `PrelucrareSoluție` este o *funcție procedurală* care efectuează *prelucrarea elementului selectat*, *conform condițiilor* impuse de enunțul problemei. De obicei în această procedură, soluția s_i este *afișată la ecran* sau este *scrisă într-un fișier*.

Metoda Trierii – Problema Pușculiței

Într-o pușculiță se afla N *monede de diferite valori* cu *greutatea totală* G grame. Greutatea fiecărei monede de o anumită valoare este dată în tabelul ce urmează:

Valoarea monedei	**Greutatea monedei**, *grame*
1	1
5	2
10	3
25	4
50	5

Elaborați un program care determină *suma minimă* S care ar putea sa fie în pușculiță.

Metoda Trierii – Problema Pușculiței

Soluție

Rezolvarea problemei prin metoda trierii constă în examinarea tuturor combinațiilor de monede, aceasta se face prin 5 cicluri incluse în interiorul programului principal:

```
1 for a in range (0,n):
2     for b in range (0,n):
3         for c in range (0,n):
4             for d in range (0,n):
5                 for e in range (0,n):
6                     if (Sol_Pos(a,b,c,d,e))
7                         Prel_Sol(a,b,c,d,e)
```

Variabila `a` reprezintă numărul de monede de valoarea `1`, `b` – numărul de monede de valoarea `2` etc. Instrucțiunile de ciclu încep de la `0` deoarece *poate să nu existe monede*. Funcția `Sol_Pos(a,b,c,d,e)` determina dacă `a,b,c,d,e` sunt o *soluție posibilă*: *suma reprezentată* `a,b,c,d,e` este egala cu `S` și numărul total de monede au greutatea `G`. Procedura `Prel_Sol(a,b,c,d,e)` – determină suma `Sum` și o compară cu suma minimă `SumMin`. Dacă `Sum` *este mai mică* ca `SumMin`, lui `SumMin` i se atribuie valoarea `Sum`, dacă `SumMin` are valoarea `-1` (*nu a fost încă determinată o valoare minimă*), i se atribuie valoarea din `Sum`.

Metoda Trierii – Problema Pușculiței

```
1  valm=[1,5,10,25,50]
2  nr=[1,2,3,4,5]
3
4  def Sol_Pos(a,b,c,d,e):
5      if (nr[0]*a+nr[1]*b+nr[2]*c+nr[3]*d+nr[4]*e==g) and (a+b+c+d+e==n):
6          return True
7      else:
8          return False
9
10 def Prel_Sol(a,b,c,d,e):
11     SumMin=-1
12     Sum=valm[0]*a+valm[1]*b+valm[2]*c+valm[3]*d+valm[4]*e
13     if Sum>SumMin:
14         SumMin=Sum
15     if SumMin==-1:
16        SumMin=Sum
17     return print(SumMin)
18
19 n=int(input('Dati numarul de monede din pusculita n='))
20 g=int(input('Dati greutatea monedelor din pusculita g='))
21
22 for a in range (0,n):
23     for b in range (0,n):
24         for c in range (0,n):
25             for d in range (0,n):
26                 for e in range (0,n):
27                     if (Sol_Pos(a,b,c,d,e)):
28                         Prel_Sol(a,b,c,d,e)
```

Metoda Trierii – Problema Pușculiței

Valoarea monedei	Greutatea monedei, *grame*
1	1
5	2
10	3
25	4
50	5

Shell

```
Dati numarul de monede din pusculita n=3
Dati greutatea monedelor din pusculita g=5
11
12
```

Shell

```
Dati numarul de monede din pusculita n=5
Dati greutatea monedelor din pusculita g=10
26
27
37
38
58
```

Shell

```
Dati numarul de monede din pusculita n=2
Dati greutatea monedelor din pusculita g=6
30
51
```

Metoda Trierii – Problema Pușculiței

```
valm=[1,5,10,25,50]
nr=[1,2,3,4,5]

def Sol_Pos(a,b,c,d,e):
        if (nr[0]*a+nr[1]*b+nr[2]*c+nr[3]*d+nr[4]*e==g) and (a+b+c+d+e==n):
                return True
        else:
                return Falsedef

def Prel_Sol(a,b,c,d,e):
        SumMin=-1
        Sum=valm[0]*a+valm[1]*b+valm[2]*c+valm[3]*d+valm[4]*e
        if Sum>SumMin:
                SumMin=Sum
         if SumMin==-1:
                SumMin=Sum
        return print(SumMin)

n=int(input('Dati numarul de monede din pusculita n='))
g=int(input('Dati greutatea monedelor din pusculita g='))
for a in range (0,n):
        for b in range (0,n):
                for c in range (0,n):
                        for d in range (0,n):
                                for e in range (0,n):
                                        if (Sol_Pos(a,b,c,d,e)):
                                                Prel_Sol(a,b,c,d,e)
```

Metoda Trierii – Avantaje și Dezavantaje

Avantaje și neajunsuri

- Complexitatea temporală a algoritmilor bazați pe metoda trierii este determinată de *numărul de elemente* k din *mulțimea soluțiilor posibile* S.

✓ *Avantajul principal* al algoritmilor bazați pe metoda trierii constă în faptul că programele respective sunt relativ simple, iar depanarea lor nu necesită teste sofisticate.

✗ În *majoritatea problemelor de o reală importanță practică*, *metoda trierii* conduce la algoritmii exponențiali. Întrucât algoritmii exponențiali sunt inacceptabili în cazul datelor de intrare foarte mari, metoda trierii este aplicată numai în scopuri didactice sau pentru elaborarea unor programe timpul de execuție al cărora nu este critic.

Metoda Trierii – Rezolvă independent

1. Se consideră numerele naturale din mulţimea {0, 1, 2, …, n}. Elaboraţi un program care determină pentru cîte numere K din această mulţime suma cifrelor fiecărui număr este egală cu m. În particular, pentru n=100 si m=2, în mulţimea {0, 1, 2, …, 100} există 3 numere care satisfac condiţiile problemei: 2, 11 si 20.Prin urmare, K=3.

2. Două numere se numesc prietenoase, dacă sunt alcătuite din unele și același cifre, nu neapărat în aceeași ordine. De exemplu, numerele 1132 și 32321 sunt prietenoase, iar 12 și 123 - nu (în primul număr lipsește cifra 3). Elaborați un program care determină dacă oricare două numere sunt prietenoase.