Facultatea de Matematică și Informatică

UNIVERSITATEA BABEȘ-BOLYAI
BABEȘ-BOLYAI TUDOMÁNYEGYETEM
BABEȘ-BOLYAI UNIVERSITÄT
BABEȘ-BOLYAI UNIVERSITY
TRADITIO ET EXCELLENTIA

Backtracking – Probleme Dificile

Paul-Ioan Someșan

Backtracking și Programare Dinamică

- Backtracking-ul este o metoda de programare care rezolva problemele prin generarea tuturor solutiilor.
- Atunci cand ne dorim doar determinarea numarului de solutii, aceasta metoda este prea lenta – aproape mereu exista variante mai eficiente.
- În această prezentare:
  - pornim de la probleme clasice rezolvate prin Backtracking
  - analizăm structura soluțiilor și modul în care acestea sunt generate
  - construim trecerea naturală către Programarea Dinamică, necesară în rezolvarea eficientă a grilelor

Ce este Backtracking-ul (și ce nu este)

- **Backtracking-ul nu înseamnă doar generarea tuturor soluțiilor**
- Backtracking-ul este o tehnică de **explorare recursivă a tuturor posibilităților**
- El poate fi folosit pentru:
  - generarea tuturor soluțiilor
  - **numărarea tuturor soluțiilor**
  - determinarea unei soluții optime (maxim / minim)
- **Ideea centrală:**
  - construim soluția **pas cu pas**
  - la fiecare pas alegem dintre mai multe posibilități valide
  - folosim **funcții recursive cu ramificări multiple**
  - explorăm toate modalitățile posibile de construcție

De la Backtracking la Programare Dinamică

De ce Backtracking-ul nu este suficient

- Backtracking-ul este **corect**, dar:
  - are complexitate **exponențială**
  - generează aceleași subprobleme de foarte multe ori
- În multe probleme **nu ne interesează soluțiile în sine**, ci:
  - **numărul total de soluții**
  - valoarea maximă / minimă
  - lungimea unei structuri optime

Aici intervine Programarea Dinamică

Programarea Dinamică:

- evită generarea tuturor posibilităților
- memorează rezultate intermediare
- permite determinarea rezultatului **mult mai eficient**

Este mult mai ușor de aplicat:

- în **probleme de concurs**
- la **grile**
- chiar și **pe hârtie**, prin relații de recurență clare

Ideea-cheie a prezentării

- **Backtracking → înțelegerea problemei**
- **Programare Dinamică → rezolvarea eficientă**
- Backtracking-ul ne ajută să:
  - înțelegem spațiul soluțiilor
  - identificăm subproblemele
- Programarea Dinamică ne ajută să:
  - calculăm rapid rezultatul
  - evităm calculele redundante

Concluzie:

- În foarte multe probleme, Programarea Dinamică este **forma optimizată a unui Backtracking bine înțeles**.

Backtracking vs Programare Dinamică (exemplu: Fibonacci)

- **Problema:** Se cere determinarea celui de-al **n-lea termen din șirul lui Fibonacci**

$$F(0) = 0, F(1) = 1, F(n) = F(n-1) + F(n-2)$$

```
int fibo(int n){
    if(n <= 1)
        return n;
    return fibo(n-1) + fibo(n-2);
}
```

```
int fibo(int n){
    int f[101];
    f[0] = 0; f[1] = 1;
    for(int i = 2; i <= n; ++i)
        f[i] = f[i-1] + f[i-2];
    return f[n];
}
```

Exercitii

Sursa: Simulare Examen Admitere UBB 2025 – Zece la Examene

13. Se considera algoritmul `f(n)` unde n este număr natural ($1 \leq n \leq 10^3$).

```
Algorithm f(n):
        If n = 0 then
                Return 1
        Else
                sum ← 0
                For i ← 1, n execute
                        sum ← sum + f(n - i)
                EndFor
                Return sum
        EndIf
EndAlgorithm
```

Precizați care dintre următoarele afirmații sunt adevărate:

A. Numărul total de apeluri care se efectuează pentru a calcula valoarea f(7) este 128.
B. Numărul total de apeluri care se efectuează pentru a calcula valoarea f(11) este 1024.
C. Numărul total de apeluri care se efectuează pentru a calcula valoarea f(9) este 512.
D. Numărul total de apeluri care se efectuează pentru a calcula valoarea f(11) este 2048.

Sursa: Examen Admitere UBB Vara 2024

24. Se consideră algoritmul `calculeaza(v, b, n, i)`, unde b, n, i sunt numere naturale nenule ($1 \leq b, n, i \leq 10^3$), iar v este un vector cu n elemente numere naturale ($v[1], v[2], ..., v[n]$, $0 \leq v[i] \leq 10^3$, pentru $i = 1, 2, ..., n$):

```
Algorithm calculeaza(v, b, n, i):
    If b = 0 then
        Return True
    EndIf
    If i = n then
        Return False
    EndIf
    Return calculeaza(v, b - v[i], n, i + 1) OR calculeaza(v, b, n, i + 1)
EndAlgorithm
```

Pentru care din următoarele date de intrare algoritmul returnează *True*?

A. $v = [3, 1, 7, 4, 2]$, $b = 10$, $n = 5$, $i = 1$
B. $v = [2, 6, 4, 8, 12]$, $b = 12$, $n = 5$, $i = 1$
C. $v = [3, 1, 7, 4, 2]$, $b = 10$, $n = 5$, $i = 2$
D. $v = [2, 6, 4, 8, 12]$, $b = 12$, $n = 5$, $i = 3$

Sursa: Concurs Mate-Info UBB 2024

21. Se consideră algoritmul `f(x, n, m)`, unde n și m sunt numere naturale ($1 \leq n, m \leq 10^4$), iar x este un vector de n numere naturale ($x[1], x[2], ..., x[n]$, $1 \leq x[i] \leq 10^4$, pentru $i = 1, 2, ..., n$):

Ce valoare va returna algoritmul, dacă apelul are forma `f(x, 9, 41)`, unde $x = [41, 15, 5, 8, 10, 1, 16, 18, 19]$?

A. 1 B. 3 C. 5 D. 7

```
Algorithm f(x, n, m):
    If m = 0 then
        Return 1
    EndIf
    If n = 0 then
        Return 0
    EndIf
    If x[n] > m then
        Return f(x, n - 1, m)
    Else
        Return f(x, n - 1, m) + f(x, n - 1, m - x[n])
    EndIf
EndAlgorithm
```

Sursa: Concurs Mate-Info UBB 2025

21. Se consideră algoritmul `p(x, n, a, b, c, d)`, unde x este un vector cu n ($0 \leq n \leq 100$) elemente întregi ($x[1], x[2], ..., x[n]$, unde $-100 \leq x[i] \leq 100$, pentru $i = 1, 2, ..., n$), iar a, b, c și d sunt numere întregi ($0 \leq a, b, c, d \leq 100$).

```
Algorithm p(x, n, a, b, c, d):
    If n = 0 then
        Return a = b AND c = d
    EndIf
    p1 ← p(x, n - 1, a + x[n], b, c * x[n], d)
    p2 ← p(x, n - 1, a,  b + x[n], c, d * x[n])
    Return p1 OR p2
EndAlgorithm
```

Știind că x = `[2, 9, 5, 6, 8, 4, 1, 2, 5, 3, 4, 1, 9, 6, 8, 3]`, care dintre următoarele afirmații sunt adevărate?

A. În urma apelului `p(x, 16, 0, 0, 1, 1)`, algoritmul returnează *True*.
B. În urma apelului `p(x, 16, 0, 0, 1, 1)`, algoritmul returnează *False*.
C. Corespunzător apelului `p(x, 16, 0, 0, 1, 1)`, algoritmul intră în ciclu infinit.
D. În urma apelului `p(x, 16, 0, 0, 1, 1)`, algoritmul returnează același rezultat pentru oricare permutare a elementelor vectorului x.

Sursa: Examen Admitere UBB Vara 2025

23. Se consideră algoritmul `interesant(a, b, c, n)`, unde c și n sunt numere naturale ($1 \leq n \leq 100$, $1 \leq c \leq 10^4$), iar a și b sunt doi vectori de lungime n, cu elemente numere întregi ($a[1], a[2], ..., a[n]$ și $b[1], b[2], ..., b[n]$, $1 \leq a[i], b[i] \leq 100$, pentru $i = 1, 2, ..., n$). Algoritmul `max(x, y)` returnează valoarea maximă dintre numerele x și y.

```
Algorithm interesant(a, b, c, n):
    If n = 0 OR c = 0 then
        Return 0
    EndIf
    If a[n] > c then
        Return interesant(a, b, c, n - 1)
    EndIf
    x ← b[n] + interesant(a, b, c - a[n], n - 1)
    y ← interesant(a, b, c, n - 1)
    Return max(x, y)
EndAlgorithm
```

Ce valoare se returnează în urma apelului `interesant([2, 3, 1], [6, 10, 3], 5, 3)`?

A. 13
B. 10
C. 16
D. 19

Sursa: Concurs Mate-Info UBB 2025

22. Se consideră algoritmii `rec(n, x, i, j)` și `ceFace(n, x)`, unde n este număr natural ($1 \leq n \leq 10^3$), iar x este un vector cu n elemente numere întregi ($x[1], x[2], ..., x[n]$, $-100 \leq x[k] \leq 100$, pentru $k = 1, 2, ..., n$), iar i și j sunt numere întregi în intervalul $[0, n]$. Algoritmul `maxim(a, b)` returnează valoarea mai mare dintre a și b.

```
Algorithm rec(n, x, i, j):
    If i = n then
        Return 0
    EndIf
    a ← rec(n, x, i + 1, j)
    b ← 0
    If j = 0 then
        b ← 1 + rec(n, x, i + 1, i)
    Else
        If x[i] > x[j] then
            b ← 1 + rec(n, x, i + 1, i)
        EndIf
    EndIf
    Return maxim(a, b)
EndAlgorithm
```

```
Algorithm ceFace(n, x):
    Return rec(n, x, 1, 0)
EndAlgorithm
```

Care dintre următoarele afirmații sunt adevărate?

A. Pentru un vector x ordonat strict crescător algoritmul `ceFace(n, x)` va returna valoarea n.
B. Complexitatea de timp a algoritmului în cazul cel mai defavorabil este $O(n^2)$.
C. În urma apelului `ceFace(8, [10, 15, 9, 30, 21, 50, 42, 60])` algoritmul returnează valoarea 5.
D. În urma apelului `ceFace(2, [3, 2])` algoritmul returnează valoarea 1.

Sursa: Examen Admitere UBB Vara 2024

18. Se consideră algoritmul `ceFace(n)`, unde n este număr natural nenul ($1 \leq n < 10^3$).

```
Algorithm ceFace(n):
    Return ceFaceRecursiv(n, 1, 1)
EndAlgorithm

Algorithm ceFaceRecursiv(n, a, b):
    If n = 0 then
        Return 1
    Else
        If n < 0 OR b > n then
            Return 0
        Else
            Return ceFaceRecursiv(n, a + b, a) + ceFaceRecursiv(n - a, a + b, a)
        EndIf
    EndIf
EndAlgorithm
```

Care dintre următoarele afirmații sunt adevărate?

A. În intervalul [11, 16] există o singură valoare x, pentru care algoritmul `ceFace(x)` returnează 1.
B. Pentru orice număr n, algoritmul `ceFace(n)` va returna valoarea 0 sau 1.
C. Algoritmul `ceFace(n)` returnează numărul de moduri de a scrie numărul n ca sumă de numere consecutive.
D. Algoritmul `ceFace(n)` returnează numărul de mulțimi diferite ale căror elemente sunt numere Fibonacci diferite de 0 și care au suma egală cu n.

Sursa: Examen Admitere UBB Vara 2024

27. Se consideră algoritmii `f(n, p)` și `g(n)`, unde ***n*** și ***p*** sunt inițial numere naturale ($1 \leq n, p \leq 10^6$ la momentul apelului inițial).

```
Algorithm g(n):
    If n < 2 then
        return False
    EndIf
    i ← 2
    While i * i ≤ n execute
        If n MOD i = 0 then
            return False
        EndIf
        i ← i + 1
    EndWhile
    return True
EndAlgorithm
```

```
Algorithm f(n, p):
    If n = 0 then
        return 1
    EndIf
    If n > 0 AND n ≥ p then
        c ← 0
        If g(p) = True then
            c ← c + f(n - p, p + 1)
        EndIf
        return c + f(n, p + 1)
    EndIf
    return 0
EndAlgorithm
```

Precizați care dintre următoarele afirmații sunt adevărate:

A. Algoritmul `g(n)` returnează *True* dacă numărul ***n*** este prim și *False* în caz contrar.
B. Pentru apelul `f(n, 2)` se returnează numărul de moduri diferite în care numărul ***n*** poate fi scris ca sumă de cel puțin un termen de numere prime distincte în ordine strict crescătoare.
C. Pentru apelul `f(n, 2)` se returnează suma divizorilor primi ai numărului ***n***.
D. Apelurile `f(n, 1)` și `f(n, 2)` vor returna același rezultat, oricare ar fi ***n***.

Sursa: Concurs Mate-Info UBB 2022

26. Se consideră algoritmul `f2(a,b)` cu parametrii ***a*** și ***b*** numere naturale, și algoritmul `f(arr, i, n, p)` având ca parametri șirul ***arr*** cu ***n*** numere întregi (***arr***[1], ***arr***[2], …, ***arr***[n]), și numerele întregi ***i*** și ***p***.

```
Algorithm f2(a, b):
    If a > b then
        return a
    else
        return b
    EndIf
EndAlgorithm

Algorithm f(arr, i, n, p):
    If i = n then
        return 0
    EndIf
    n1 ← f(arr, i + 1, n, p)
    n2 ← 0
    If p + 1 ≠ i then
        n2 ← f(arr, i + 1, n, i) + arr[i]
    EndIf
    return f2(n1, n2)
EndAlgorithm
```

Precizați care este rezultatul apelului `f(arr, 1, 9, -10)`, dacă șirul ***arr*** conține valorile (10, 1, 5, 4, 7, 12, 1, 12, 6).

A. 24
B. 37
C. 39
D. 56

Sursa: Simulare Examen Admitere UBB 2025 – Zece la Examene

22. Se considera algoritmul `nrMod(n, k)` care primește ca si parametrii de intrare 2 numere naturale nenule n si k ($1 \leq n, k \leq 20$).

```
Algorithm nrMod(n, k):
        If k = n or k = 1 then
                Return 1
        EndIf
        rez ← 0
        For i ← 1, n - k + 1 execute
                rez ← rez + nrMod(n - i, k - 1)
        EndFor
        Return rez
EndAlgorithm
```

Precizați care dintre următoarele afirmații sunt adevărate:

A. `nrMod(16, 6) = 4380`
B. `nrMod(14, 5) = 715`
C. `nrMod(12, 3) = 56`
D. `nrMod(15, 5) = 1001`

Facultatea de Matematică și Informatică

UNIVERSITATEA BABEȘ-BOLYAI
BABEȘ-BOLYAI TUDOMÁNYEGYETEM
BABEȘ-BOLYAI UNIVERSITÄT
BABEȘ-BOLYAI UNIVERSITY
TRADITIO ET EXCELLENTIA

Va multumesc!

Zecelagrile.ro – peste 4000 de grile de informatica