Metoda Backtracking

Cadru posibil

- Soluția se poate reprezenta sub formă de **vector** $x=(x_1, x_2,\ldots,x_n) \in X$:

  $X=X_1 \times \ldots \times X_n$ = spaţiul soluţiilor candidat

- $p : X \to \{0,1\}$ este o **proprietate** definită pe X pe care trebuie să o verifice soluția – numită condiție internă (finală) pentru x

- **Căutăm $x \in X$ cu proprietatea $p(x)$**

Metoda Backtracking

- Generarea tuturor elementelor produsului cartezian X (și testarea proprietății p) nu este acceptabilă.

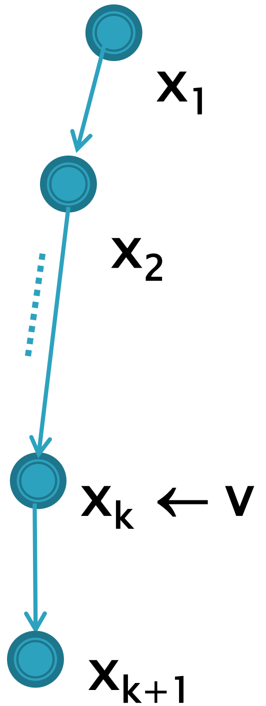
- Metoda backtracking încearcă micşorarea timpului de calcul – prin **evitarea generării unor elemente din X**

Metoda Backtracking

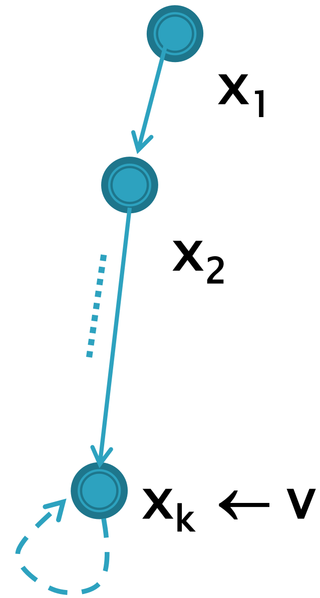
- Vectorul soluție $x = (x_1, x_2, \ldots, x_n) \in X$ este **construit progresiv**, începând cu prima componentă.
- Pentru elementul curent x_k se atribuie pe rând toate valorile din X_k
- Se avansează cu o valoare pentru x_k (se trece la x_{k+1}) dacă este satisfăcută o **condiţie de continuare**
  - Dacă nu este satisfăcută această condiție =>

    nu există valori pentru x_{k+1}, .., x_n astfel încât soluția obținută x să verifice condițiile finale p
  - rezultă de obicei din condițiile finale
- Când se termină valorile pentru x_k se revine la x_{k-1}

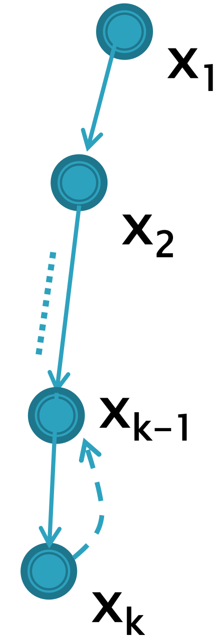
Metoda Backtracking



sunt verificate condiţiile de continuare

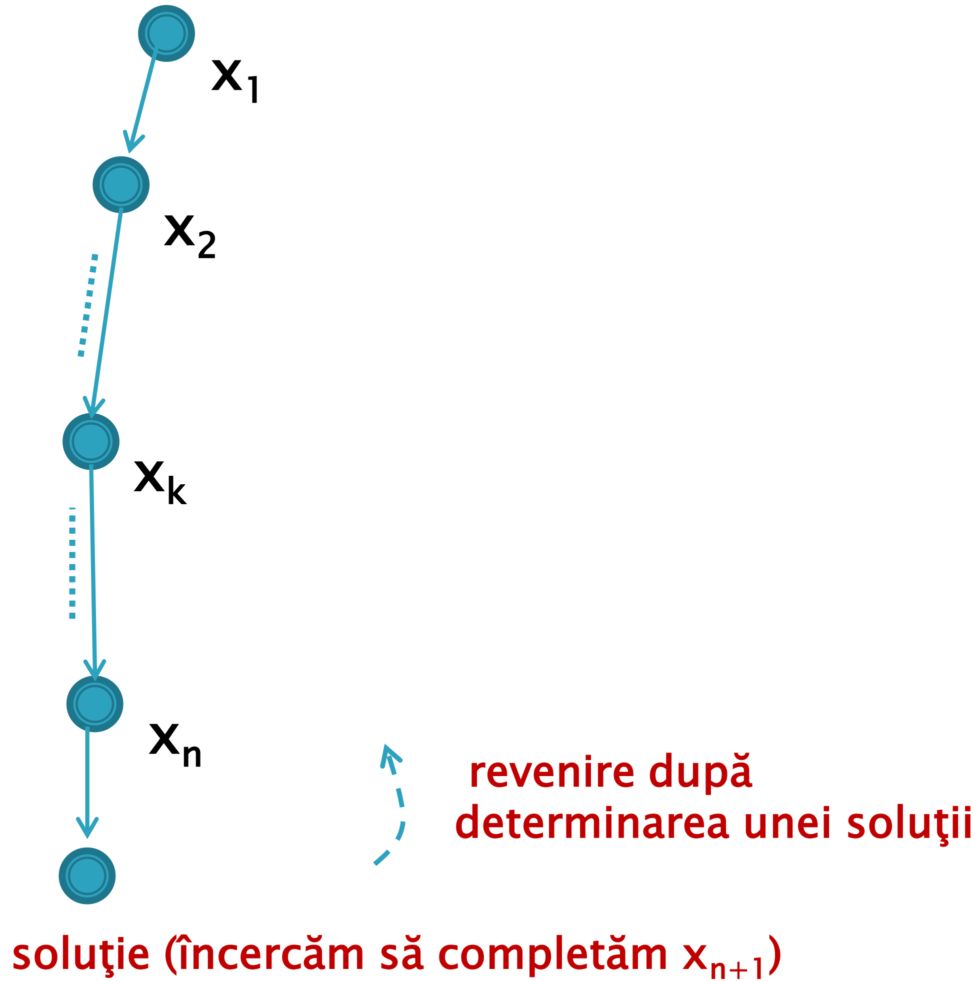


nu sunt verificate condiţiile de continuare



nu mai există valori pentru x_k neconsiderate

Metoda Backtracking



Exemplu

- **Exemplu** – Generarea permutărilor

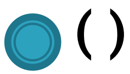
Permutări, n=3

```
1
1 1
1 2
1 2 1
1 2 2
1 2 3
1 2 3 soluție
1 3
1 3 1
1 3 2
1 3 2 soluție
1 3 3
```

```
2
2 1
2 1 1
2 1 2
2 1 3
2 1 3 soluție
etc
```

Exemplu

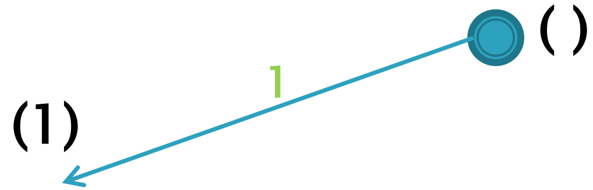
- Permutări `{1, 2, 3}`



Exemplu

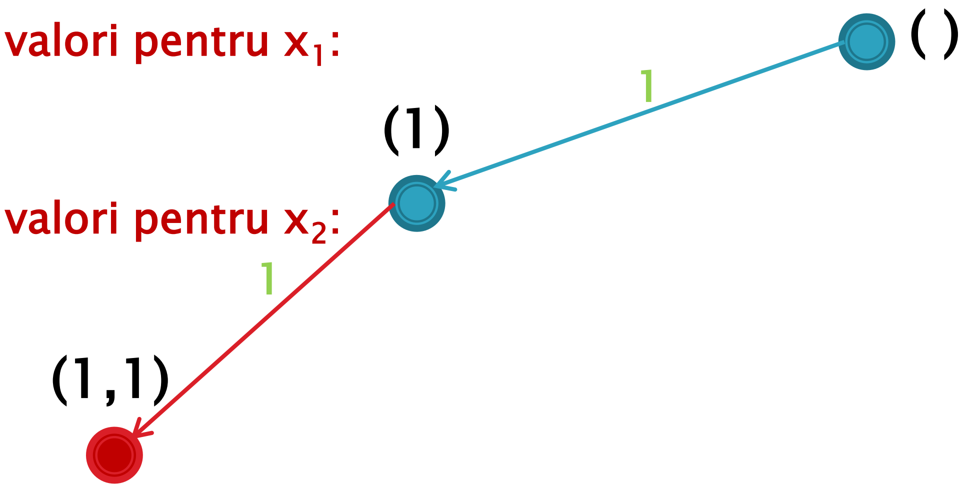
- Permutări {1, 2, 3}

valori pentru x_1:



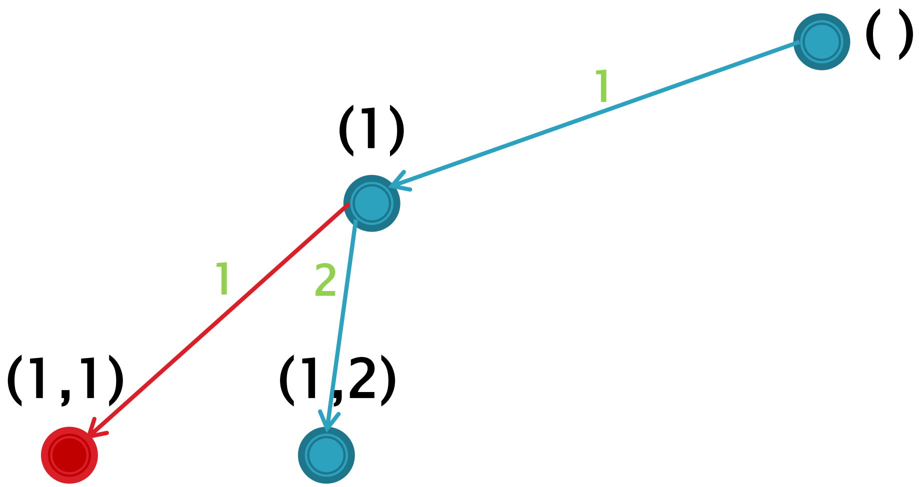
Exemplu

- Permutări `{1, 2, 3}`



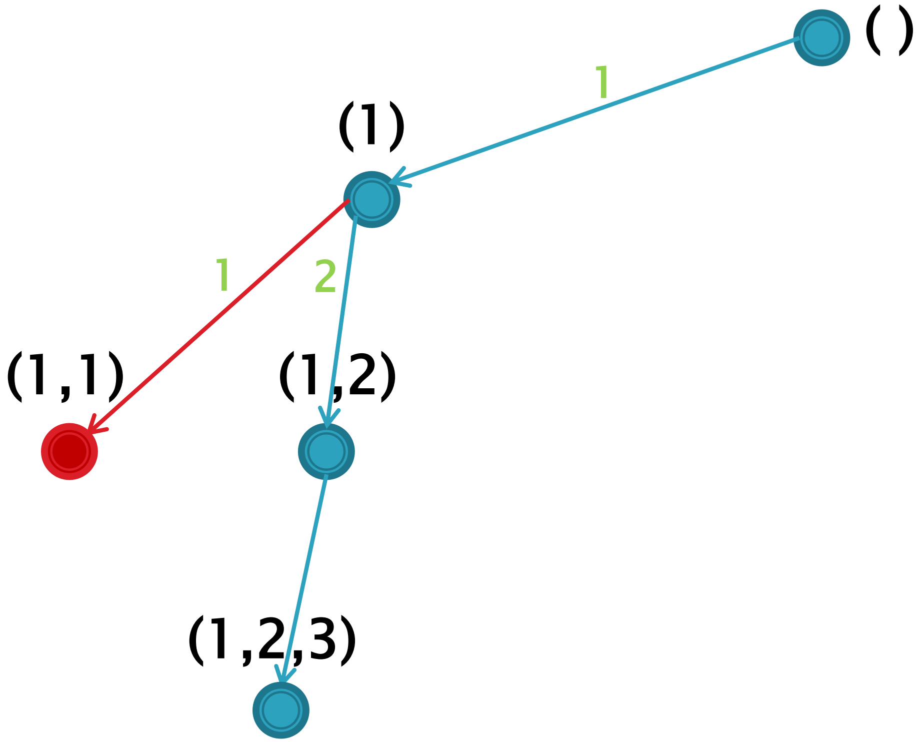
Exemplu

- Permutări `{1, 2, 3}`



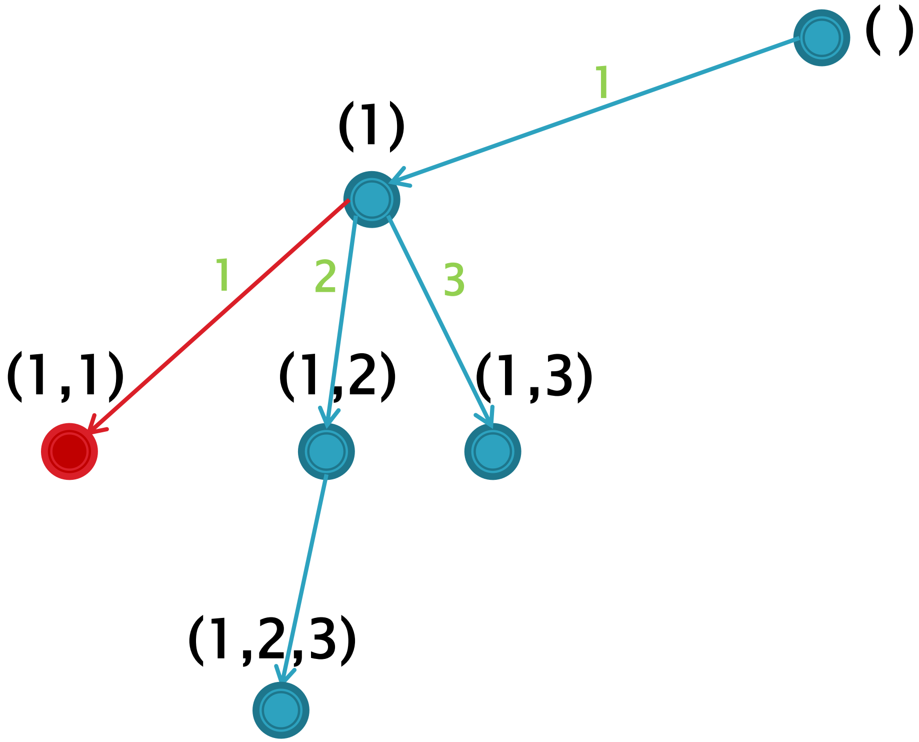
Exemplu

- Permutări `{1, 2, 3}`



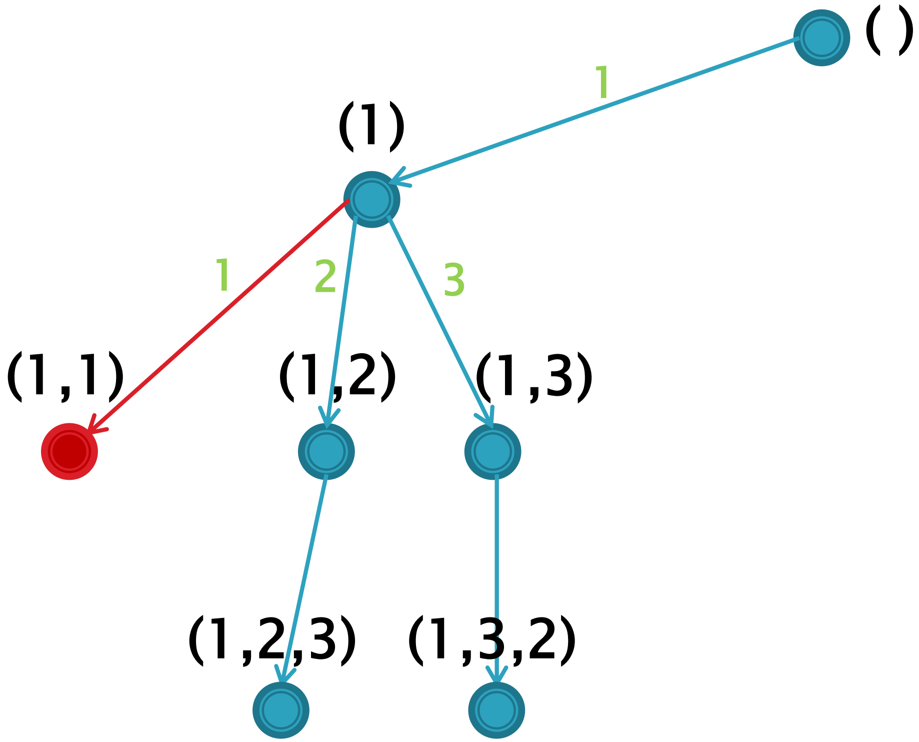
Exemplu

- Permutări `{1, 2, 3}`



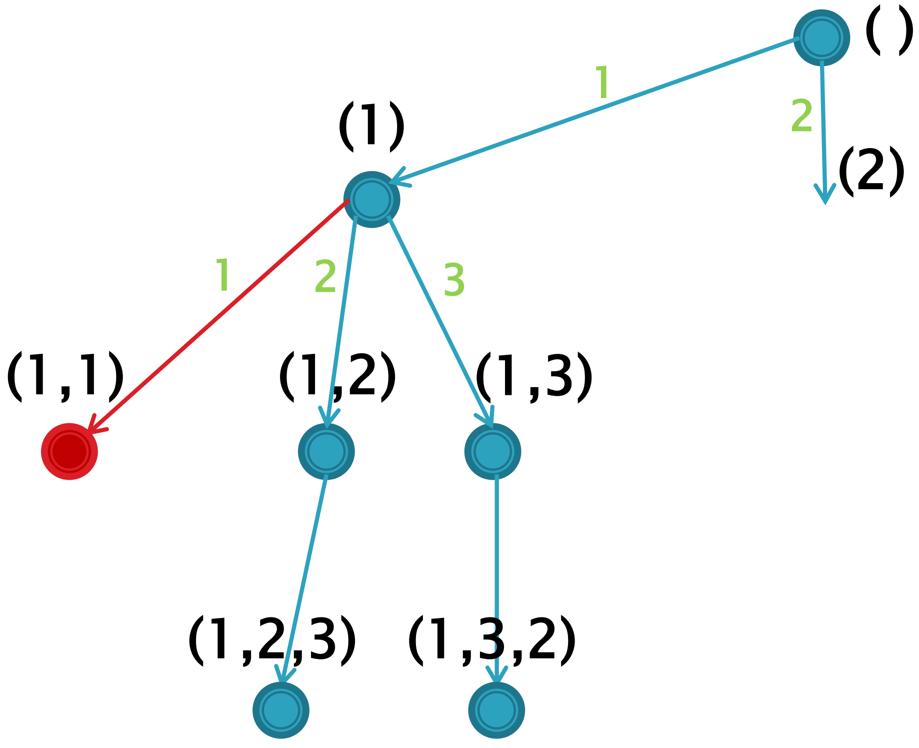
Exemplu

- Permutări `{1, 2, 3}`



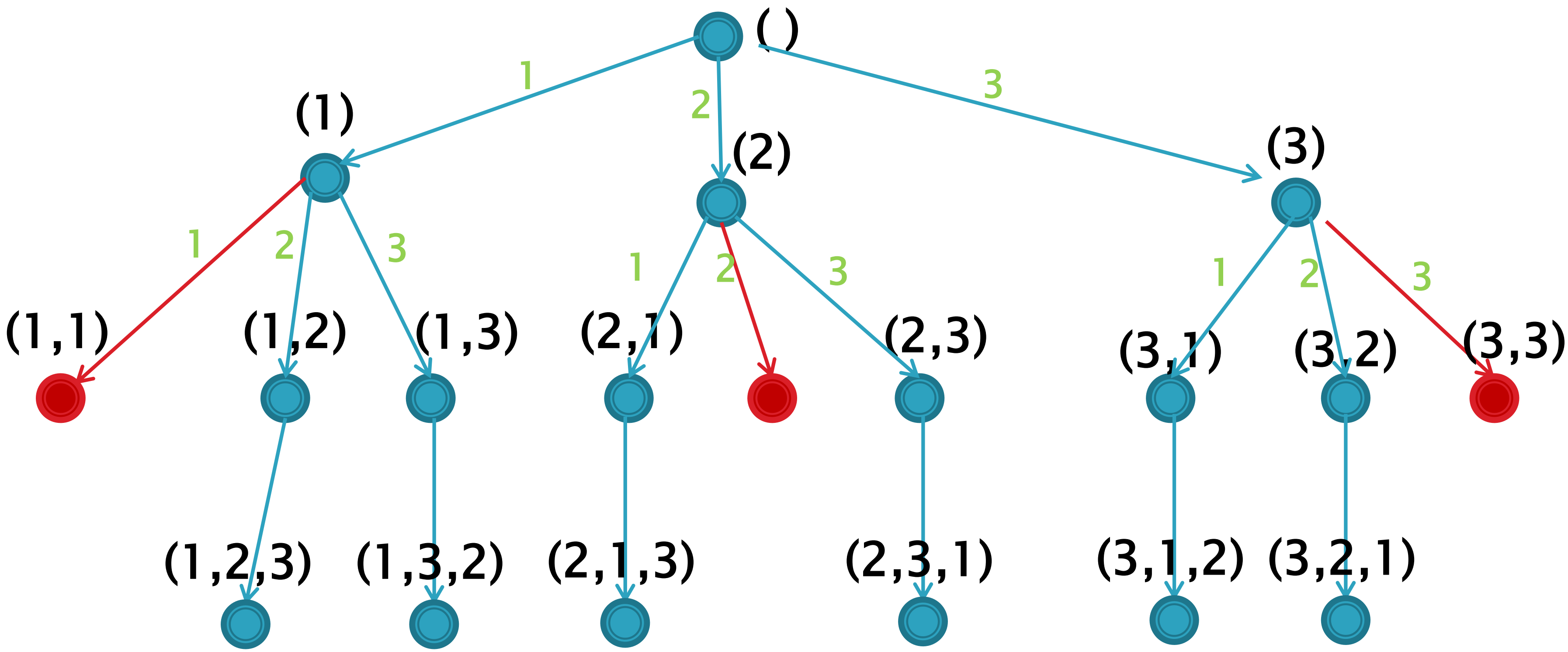
Exemplu

- Permutări `{1, 2, 3}`



Exemplu

- Permutări `{1, 2, 3}`



Metoda Backtracking

Condiții de continuare **pentru soluția parțială $y=x_1 \ldots x_k$** notate

continuare(x,k) = condiții de **continuare a completării soluției**

- **Condiţiile de continuare**
  - **rezultă de obicei din condițiile interne (finale)**
  - sunt **strict necesare**, **ideal fiind să fie şi suficiente**
  - sunt importante pentru micșorarea timpului de executare

```
def back(k):
    if k == n: #am completat x_0...x_{n-1} => tot x
        if test_solutie(): #daca este necesar
            retine_solutie()
    else:
        for i in Xk: #Xk=valori posibile pentru x[k]
            x[k]=i
            if continuare(x,k):
                back(k+1)
```

Observații

- În **`test_solutie()`** este posibil sa fie nevoie de testarea condițiilor finale, dacă pe parcurs condițiile de continuare nu au fost suficiente cât să garanteze obținerea unei soluții corecte

- Metoda poate fi folosită și în:
  - **probleme de optim** - atunci in retine_sol memorăm cea mai bună soluție generată până acum conform criteriului de optim
  - **probleme de numărare** - atunci in retine_sol contorizăm soluțiile generate până acum

Exemple

Exemple

- Permutări, combinări, aranjamente
- Produs cartezian
- Submulțimi
- Partiţiile unui număr n

Permutări

- Permutările mulțimii {1,2,... ,n}

Permutări

- Reprezentarea soluţiei

- Condiţii interne (finale)

- Condiţii de continuare (!!pentru x_k)

Permutări

- **Reprezentarea soluţiei**

  $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$, unde

  $\mathbf{x}_k \in \{1, 2, \ldots, n\}$ $(\mathbf{p}_k = 1,\ \mathbf{u}_k = \mathbf{n})$.

- **Condiţii interne (finale)**

- **Condiţii de continuare (!!pentru x_k)**

Permutări

- **Reprezentarea soluţiei**

  $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$, unde

  $\mathbf{x}_k \in \{1, 2, \ldots, n\}$ $(\mathbf{p}_k = 1, \mathbf{u}_k = \mathbf{n})$.

- **Condiţii interne (finale)**

  $\mathbf{x}_i \neq \mathbf{x}_j$ pentru orice $i \neq j$.

- **Condiţii de continuare (!!pentru x_k)**

Permutări

- **Reprezentarea soluţiei**

  $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$, unde

  $\mathbf{x}_k \in \{1, 2, \ldots, n\} \quad (p_k = 1,\ u_k = n)$.

- **Condiţii interne (finale)**

  $\mathbf{x}_i \neq \mathbf{x}_j$ pentru orice $i \neq j$.

- **Condiţii de continuare (!!pentru x_k)**

  $\mathbf{x}_i \neq \mathbf{x}_k$ pentru orice $i \in \{1, 2, \ldots, k-1\}$

Permutări, n=3

```
1
1 1
1 2
1 2 1
1 2 2
1 2 3
1 2 3 soluție
1 3
1 3 1
1 3 2
1 3 2 soluție
1 3 3
```

```
2
2 1
2 1 1
2 1 2
2 1 3
2 1 3 soluție
etc
```

Aranjamente

- Aranjamente de m elemente ale mulțimii {1,2,… ,n} (contează ordinea)

```
n = 5
m = 3:
      1 2 3
      1 2 4
      1 2 5
      1 3 2 ...
```

Aranjamente

- **Reprezentarea soluţiei**

  $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m)$, unde

  $\mathbf{x}_k \in \{1,2,\dots,n\}$ $(p_k = 1,\ u_k = n)$.

- **Condiţii interne (finale)** – ca la permutări

  $\mathbf{x}_i \neq \mathbf{x}_j$ pentru orice $i \neq j$.

- **Condiţii de continuare (!!pentru x_k)**

  $\mathbf{x}_i \neq \mathbf{x}_k$ pentru orice $i \in \{1,2,\dots,k-1\}$

Combinări

- Combinări de m elemente ale mulțimii {1,2,... ,n} (submulțimi cu m elemente)

```
n = 5
m = 3:
      1 2 3
      1 2 4
      1 2 5
      1 3 4
      1 3 5 ...
```

Combinări

- **Reprezentarea soluţiei**

  $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$, unde

  $\mathbf{x}_k \in \{1,2,\ldots,n\}$ $(\mathbf{p}_k = 1, \mathbf{u}_k = \mathbf{n})$.

- **Condiţii interne (finale)** – crescător=>distincte

  $x_i < x_j$ pentru orice $i < j$.

- **Condiţii de continuare (!!pentru x_k)**

  $\mathbf{x}_{k-1} < \mathbf{x}_k$

Submulțimi

- Submulțimile mulțimii {1,2,… ,n}

O submulțime – asociat un vector caracteristic v cu n elemente 0/1 ($v_i = 0 \Leftrightarrow$ i nu aparține submulțimii)

n = 5: {1,2,3,4,5}

Submulțimea {2,3,5} -> vectorul [0,1,1,0,1]

Generare de submulțimi = generare de șiruri binare de lungime n

Submulțimi

- **Reprezentarea soluţiei**

  $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$, unde

  $\mathbf{x}_k \in \{0,1\}$

- **Condiţii interne (finale)**

- **Condiţii de continuare (!!pentru x_k)**

Problemă

Un număr natural se numește k-echilibrat ($0 \leq k \leq 9$) dacă valoarea absolută a diferenței dintre oricare două cifre aflate pe poziții consecutive este mai mare sau egală decât k. De exemplu, numărul 2908 este 7-echilibrat, iar 152629096 este 3-echilibrat.

Scrieți un program Python care să citească de la tastatură numerele naturale k și m ($2 \leq m \leq 20$), după care afișează toate numerele naturale k-echilibrate formate din exact m cifre sau un mesaj corespunzător dacă nu există niciun astfel de număr.