TUTORIAT 9

Linkuri utile:

- https://www.python.org/
- https://docs.python.org/3/
- https://www.codecademy.com/learn/learn-python-3
- https://www.learnpython.org/
- https://stanfordpython.com/#/

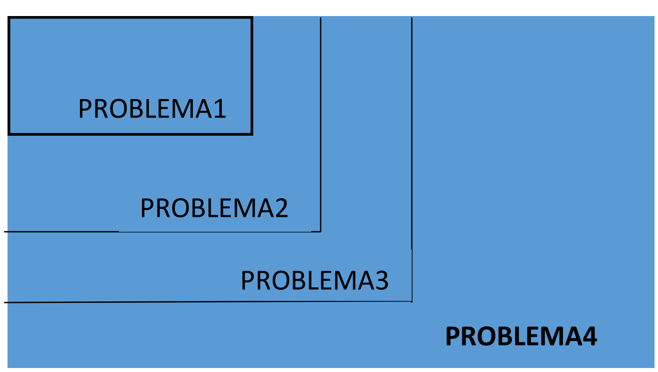
Comentariu multiplu PyCharm: CTRL + /

Ce conține tutoriatul ?

Metoda programării dinamice

- a fost dezvoltată de Richard Bellman în anul 1950
- cuvântul programare din „programare dinamică” se referă la planificare, nu la programare în sens informatic
- cuvântul dinamică face referire la modul de construire al „tabelelor” în care se rețin informațiile referitoare la soluțiile parțiale (dinamic în timp, adică deciziile care conduc la obținerea rezultatului se pot lua pas cu pas, pe baza deciziilor de la pasul sau pașii anteriori)
- de cele mai multe ori este utilizată pentru rezolvarea problemelor care solicită determinarea unui optim(minim sau maxim), în urma unui proces decizional care se desfășoară în mai multe etape: se pornește de la o stare inițială și la fiecare pas se ia o decizie care determină o nouă stare, până când se ajunge la soluția finală
- demonstrarea corectitudinii se realizează, de obicei, prin inducție matematică
- problemele rezolvate cu această metodă se pot descompune în subprobleme care se suprapun
- particularitate: fiecare subproblemă se rezolvă o singură dată, iar soluția ei este stocată pentru a putea fi ulterior folosită în rezolvarea problemei inițiale

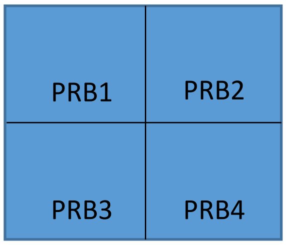
Descompunerea problemei în subprobleme suprapuse



Soluția problemei4 conține soluțiile problemelor(subproblemelor) 1, 2, 3

SOLUȚIE 1			
SOLUȚIE 2			
SOLUȚIE 3			
SOLUȚIE 4			

TEHNICA DIVIZĂRII	TEHNICA PROGRAMĂRII DINAMICE
subproblemele în care se împarte problema inițială sunt **INDEPENDENTE**, astfel, soluția unei subprobleme nu poate fi utilizată în construirea soluției altei subprobleme	subproblemele în care se împarte problema inițială sunt **DEPENDENTE**(se suprapun), astfel, soluția unei subprobleme se utilizează în construirea soluțiilor altor subprobleme(de aceea este important ca soluția fiecărei probleme să fie memorată)



sol1	**sol2**	**sol3**	**sol4**

- se poate aplica în mai multe abordări:
  - **înainte**: pentru rezolvarea problemei se pleacă de la starea finală
  - **înapoi**: pentru rezolvarea problemei se pleacă de la starea inițială
  - **mixtă**: o combinație între cele două
- problemele rezolvate prin metoda programării dinamice au două proprietăți:
  - **substructură optimală** – problema poate fi descompusă în subprobleme, iar soluțiile optime ale subproblemelor determină soluția optima a problemei inițiale(**!ATENȚIE!** se pot aplica și metodele Greedy, Divide et Impera)
  - **subprobleme superpozabile** – subproblemele nu sunt independente, ci se suprapun (în acest caz, nu se poate aplica metoda Divide et Impera)

Pași de rezolvare a unei probleme prin programare dinamică

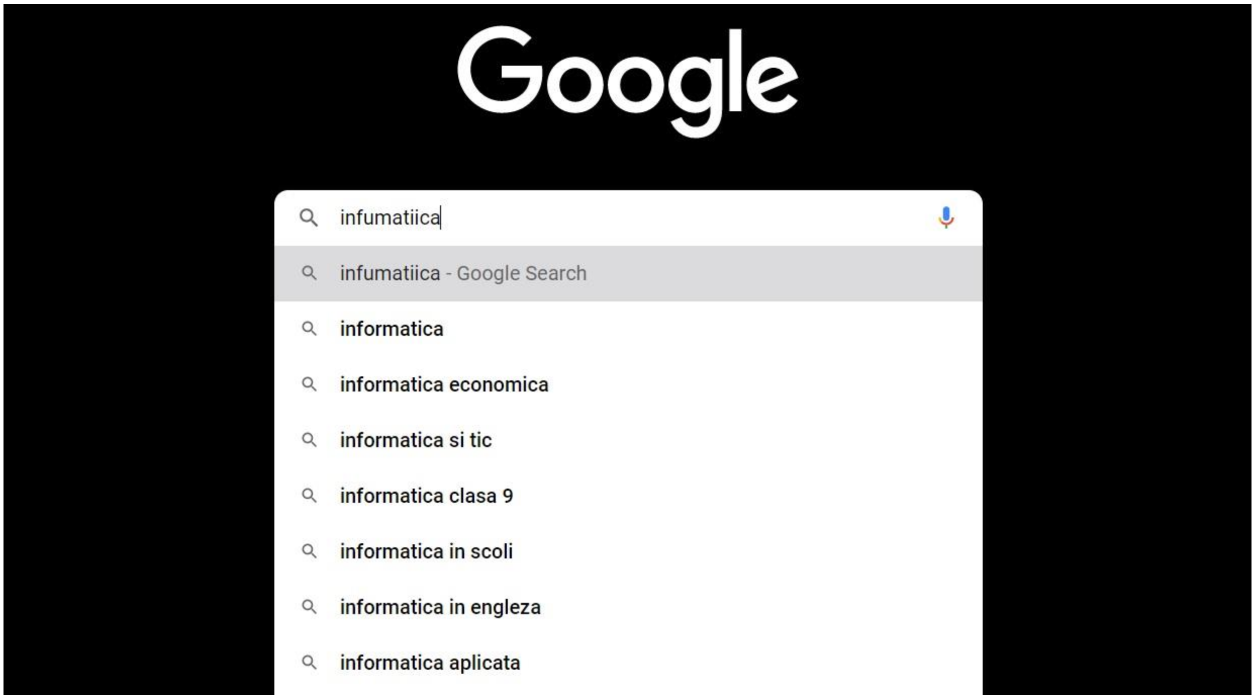
PAS1: **identificarea subproblemelor** (identificarea problemei generice; forma generală a problemei inițiale și a fiecărei subprobleme)

PAS2: **alegerea unei structuri** care să rețină soluțiile subproblemelor

PAS3: **dezvoltarea relației de recurență** (exprimă legătura dintre soluția problemei și soluțiile subproblemelor)

PAS4: **rezolvarea recurenței** în mod bottom-up (în ordinea crescătoare a dimensiunilor subproblemelor) pentru determinarea soluției optime a problemei inițiale

MOTIVAȚIE



Cum se decide faptul că s-a intenționat scrierea cuvântului informatica în loc de infumatiica?

Răspuns: se evaluează o măsură a disimilarității dintre cuvinte.

Evaluarea disimilarității dintre cuvinte

Pot apărea mai multe tipuri de erori la tastarea unui cuvânt:

1. înlocuirea unei litere cu o altă literă
   informatica -> infurmatica
2. absența unei litere
   informatica -> infomatica
3. introducerea unei litere suplimentare
   informatica -> informatiica

Distanța dintre două cuvinte reprezintă numărul minim de operații de înlocuire/ inserarție/ștergere unei litere care asigură transformarea unuia dintre cuvinte în cel corect.

Exemplu:

infumatiica -> infOmatiica -> infoRmatiica -> informatIca

înlocuire inserție ștergere

Problema 1(Calculul distanței de editare/ distanța Levenshtein)

Se consideră două șiruri de caractere(cuvinte), a[12,…m], b[12…n]. Se cere numărul minim de operații de inserție, înlocuire, ștergere caracter care permite transformarea șirului a în șirul b.

Ideea de rezolvare: se analizează cazuri particulare, apoi se încearcă extinderea soluției pentru cazul general

Se notează D[i, j] distanța de editare dintre șirurile parțiale(prefixele) a[12…i] și b[12…j].

Analiza cazurilor particulare:

1. a este șirul vid (i = 0): D[0,j] = j (sunt necesare j inserări pentru a trasforma șirul a în șirul b)
2. b este șirul vid (j = 0): D[i,0] = i (sunt necesare i ștergeri pentru a trasforma șirul a în șirul b)

Analiza cazurilor posibile:

1. a[i] = b[j], atunci D[i, j] = D[i-1, j-1]
2. a[i] != b[j], se disting trei cazuri posibile
    - a[i] se înlocuiește cu b[j]: D[i,j] = D[i-1,j-1] + 1
    - a[i] se șterge: D[i,j] = D[i-1,j] + 1
    - b[j] se inserează după a[i]: D[i,j] = D[i,j-1] + 1
3. Se va alege varianta care conduce la cel mai mic număr de operații:
   D[i,j] = min{ D[i-1, j-1], D[i-1,j], D[i,j-1]} + 1

Exemplu:

a = carte   b = caiete

$$D[i,j] = \begin{cases} i, & j = 0 \\ j, & i = 0 \\ D[i-1,j-1], & a[i] = b[j] \\ \min\{ D[i-1, j-1], D[i-1,j], D[i,j-1]\} + 1 & a[i] \neq b[j] \end{cases}$$

distanța de editare este 3

		c	a	i	e	t
0	0	1	2	3	4	5
c 1	1	0	1	2	3	4
a 2	2	1	0	1	2	3
r 3	3	2	1	1	2	3
t 4	4	3	2	2	2	2
e 5	5	4	3	3	2	3

determinarea secvenței de operații de editare:

carte -> cart -> caet -> caiet

eliminare   înlocuire   inserție

deplasări: sus diagonală stânga

Problema 2(Lăcusta)

Se consideră o matrice A cu m linii și n coloane, cu componente numere naturale.Traversăm matricea de la colțul stânga sus la coțul dreapta jos, făcând câte un salt pe orizontală și un pas pe verticală. Un salt înseamnă că putem trece de la o celulă la alta aflată pe aceeași linie, iar un pas înseamnă că putem trece de la o celulă la alta aflată imediat sub ea. Astfel, traversarea va consta din vizitarea a 2m celule.

Se cere să se găsească traversarea pentru care suma celulelor vizitate să fie minimă.

Exemplu:

4 5 suma minimă a traseului: 28

3 4 5 7 9 drumul este: (1,1)-> (1,3) -> (2,3) ->(2,2)->(3,2)->(3,3)->(4,3)->(4,5)

6 6 3 4 4

6 3 3 9 6

6 5 3 8 2

Modalitate de rezolvare

PAS1: identificarea subproblemelor

Problema cere determinarea sumei minime ce se poate obține din colțul stânga sus (A[0][0]) până în colțul dreapta jos (A[m-1][n-1]), traversând matricea în condițiile specificate.

O subproblemă a problemei date constă în determinarea sumei minime, care se poate obține pornind din colțul stânga sus până la fiecare element A[i][j] al matricei, oricare i,j.

PAS2: stabilirea structurilor de date

Se va construi o matrice S, cu m linii și n coloane, cu semnificația S[i][j] -> suma minimă care se poate obține din colțul stânga sus până la elementul A[i][j].

Soluția problemei date este:

$$\min\{S[m-1][j] \mid 0 \le j < n - 1\} + A[m-1][n-1]$$

PAS3: determinarea relației de recurență

- $S[0][0] = A[0][0]$
- $S[0][i] = \infty$, oricare $0 < i < n$ (nu se poate ajunge la elementul A[0][i] făcând un salt și un pas)
- $S[1][0] = \infty$ (nu se poate ajunge la elementul A[1][0] făcând un salt și un pas)
- $S[1][j] = A[0][0] + A[0][j] + A[1][j]$, oricare $0 < j < n$ (facem un salt până în elementul A[0][j], apoi un pas până la A[1][j])
- $S[i][j] = A[i][j] \mid A[i-1][j] + \min\{S[i-1][k] \mid 0 \le k < n, k \neq j\}$

Pentru a ajunge în A[i][j] am făcut un pas de pe elementul A[i-1][j], iar pentru a ajunge pe elementul A[i-1][j] am făcut un salt.Elementul A[i-1][kmin] de pe care se face saltul este ales astfel încât $S[i-1][kmin] = \min\{S[i-1][k] \mid 0 \leq k < n, k \neq j\}$.

PAS4: rezolvarea relației de recurență

Relația de recurență trebuie rezolvată în mod bottom-up (subproblemele problemei nu sunt independente), determinând suma minimă ce poate fi obținută pe un traseu într-o matrice de două linii, trei linii, etc.

În calculul lui S[i][j] este posibil să apară o problemă: minimul de pe linia i-1 să se afle chiar pe coloana j. În această eventualitate, vom calcula două valori minime de pe linia i-1.