Distanța Levenshtein

Buzatu Giulian

Ilie Dumitru

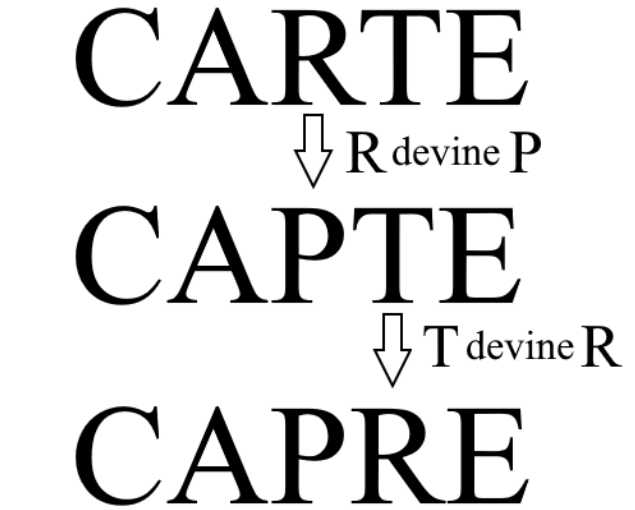
Coordonator: Lect. Dr. Dumitran Adrian Marius

Problemă

- Se dau două cuvinte, care este numărul minim de operații prin care putem ajunge la cel de-al doilea pornind de la primul?
- O operației reprezintă: inserarea unui caracter, ștergerea unui caracter sau înlocuirea unui caracter.

Exemplu

- De exemplu, pentru cuvintele carte și capre răspunsul este 2. Distanța se poate obține în două moduri:



Exemplu

- De exemplu, pentru cuvintele carte și capre răspunsul este 2. Distanța se poate obține în două moduri. Matricea obținută este:

		C	A	P	R	E
	0	1	2	3	4	5
C	1	0	1	2	3	4
A	2	1	0	1	2	3
R	3	2	1	1	1	2
T	4	3	2	2	2	2
E	5	4	3	3	3	2

Distanța Levenshtein

- Distanța Levenshtein este un tip de distanță de editare. Distanța Levenshtein este un număr care ne spune cât de diferite sunt două cuvinte.

Distanța Levenshtein

- Distanța Levenshtein este un număr care ne spune cât de diferite sunt două cuvinte. Considerăm cuvintele indexate de la 1 în momentul în care le reținem în memorie.

- $$lev_{a,b}(i,j) = \begin{cases} \max(i,j) & , \textit{dacă } \min(i,j) = 0 \\ min \begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & , \textit{altfel} \end{cases}$$

Distanța Levenshtein

- $lev_{a,b}(i,j) = \begin{cases} \max(i,j) & , dacă \min(i,j) = 0 \\ min \begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & , altfel \end{cases}$
- În relația de mai sus a și b reprezintă cele două cuvinte, iar i poziția curentă din cuvântul a și j poziția curentă din cuvântul b.

Distanța Levenshtein

- $lev_{a,b}(i,j) = \begin{cases} \max(i,j) & , \textit{dacă } \min(i,j) = 0 \\ min \begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & , \textit{altfel} \end{cases}$
- Prima ramură din recurență se întâmplă în momentul în care cel puțin unul dintre i și j este 0, adică unul dintre cuvinte este vid.

Distanța Levenshtein

- $lev_{a,b}(i,j) = \begin{cases} \max(i,j) & , dacă \min(i,j) = 0 \\ min \begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & , altfel \end{cases}$
- Primul termen din funcția de minim de pe a doua ramură se obține prin ștergerea caracterului i, deci rămânem cu un cuvânt cu $i-1$ litere și unul cu j litere, iar distanța de editare crește cu 1.
- Al doilea termen din funcția de minim de pe a doua ramură se obține prin adăugarea caracterului j, deci plecăm de la un cuvânt cu i litere și unul cu $j-1$ litere, iar distanța de editare crește cu 1.

Distanța Levenshtein

- $lev_{a,b}(i,j) = \begin{cases} \max(i,j) & , dacă \min(i,j) = 0 \\ min \begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & , altfel \end{cases}$
- Cel de-al treilea termen din funcția de minim se obține când nici nu ștergem și nici nu adăugăm vreun caracter, ci modificăm caracterul i în caracterul j, dacă este cazul. Dacă $a_i = b_j$, atunci este $lev_{a,b}(i-1,j-1)$, iar dacă $a_i \neq b_j$, atunci trebuie să adăugăm 1 la $lev_{a,b}(i-1,j-1)$.

Demonstrație

- Primul caz poate fi spart în 2 cazuri mai mici:

Demonstrație

- Primul caz poate fi spart în 2 cazuri mai mici:
  - Dacă i este 0, atunci vrem să obținem un șir de j caractere din șirul vid. Singurul mod de a obține un șir nevid dintr-unul vid este să adăugăm caractere. Trebuie să adăugăm j caractere, iar fiecare adăugare are costul 1. Avem deci costul total $j * 1 = j$.

Demonstrație

- Primul caz poate fi spart în 2 cazuri mai mici:
  - Dacă i este 0, atunci vrem să obținem un șir de j caractere din șirul vid. Singurul mod de a obține un șir nevid dintr-unul vid este să adăugăm caractere. Trebuie să adăugăm j caractere, iar fiecare adăugare are costul 1. Avem deci costul total $j * 1 = j$.
  - Dacă j este 0, atunci vrem să obținem un șir vid dintr-unul nevid. Singurul mod de a face asta este să ștergem pe rând caracterele. Trebuie să ștergem i caractere, iar fiecare ștergere are costul 1. Avem deci costul total $i * 1 = i$.

Demonstrație

- Al doilea caz are 3 subcazuri. Putem obține șirul de j caractere din cel de i caractere dacă:

Demonstrație

- Al doilea caz are 3 subcazuri. Putem obține șirul de j caractere din cel de i caractere dacă:
  - Ștergem caracterul i. Această operație are costul 1 și ne aduce în punctul în care vrem să ajungem de la șirul de $i-1$ caractere la cel de j caractere. Deci obținem costul 1 + costul de la $i-1$ la j, i.e. $lev_{a,b}(i-1, j)+1$.

Demonstrație

- Al doilea caz are 3 subcazuri. Putem obține șirul de j caractere din cel de i caractere dacă:
  - Ștergem caracterul i. Această operație are costul 1 și ne aduce în punctul în care vrem să ajungem de la șirul de $i-1$ caractere la cel de j caractere. Deci obținem costul 1 + costul de la $i-1$ la j, i.e. $lev_{a,b}(i-1, j)+1$.
  - Adăugăm caracterul j. Această operație are costul 1 și ne aduce în punctul în care vrem să ajungem de la șirul de i caractere la cel de $j-1$ caractere. Deci obținem costul 1 + costul de la i la $j-1$, i.e. $lev_{a,b}(i, j-1)+1$.

Demonstrație

- Al doilea caz are 3 subcazuri. Putem obține șirul de j caractere din cel de i caractere dacă:
  - Schimbăm caracterul i în caracterul j. Această operație are costul 1 și ne aduce în punctul în care vrem să ajungem de la șirul de $i-1$ caractere la cel de $j-1$ caractere. Deci obținem costul 1 + costul de la $i-1$ la $j-1$, i.e. $lev_{a,b}(i-1, j-1)+1$.
    - Dar, dacă caracterul i și caracterul j sunt egale? Atunci „înlocuirea" are costul 0, deoarece nu trebuie sa înlocuim efectiv caracterul. Deci costul este doar costul de la $i-1$ la $j-1$, i.e. $lev_{a,b}(i-1, j-1)$.

Algoritm

Algorithm 1 Edit distance

```
1:  function LEV(a, b, i, j)
2:      if min{i, j} = 0 then
3:          return max{i, j}
4:
5:      up ← lev(a, b, i − 1, j) + 1
6:      left ← lev(a, b, i, j − 1) + 1
7:      diag ← lev(a, b, i − 1, j − 1)
8:
9:      if  a_i ≠ b_j  then
10:         diag ← diag + 1
11:     return min{up, left, diag}
```

Reconstrucția pașilor

- Pe lângă distanța Levenshtein, ne dorim să știm pașii efectivi prin care transformăm primul cuvânt în cel de-al doilea.

Cum putem face asta?

Reconstrucția pașilor

- Pe lângă distanța Levenshtein, ne dorim să știm pașii efectivi prin care transformăm primul cuvânt în cel de-al doilea.
  - Știm că $lev_{a,b}(i,j)$ este numărul minim de operații astfel încât să convertim subșirul format din primele i caractere în subșirul format din primele j caractere. Deci $lev_{a,b}(|a|,|b|)$ este numărul minim de operații necesre pentru a transforma șirul a în șirul b.

Reconstrucția pașilor

- Pe lângă distanța Levenshtein, ne dorim să știm pașii efectivi prin care transformăm primul cuvânt în cel de-al doilea.
  - Știm că $lev_{a,b}(i,j)$ este numărul minim de operații astfel încât să convertim subșirul format din primele i caractere în subșirul format din primele j caractere. Deci $lev_{a,b}(|a|,|b|)$ este numărul minim de operații necesre pentru a transforma șirul a în șirul b.

Cum am obținut $lev_{a,b}(|a|,|b|)$?

Reconstrucția pașilor

- Cum am obținut $lev_{a,b}(|a|, |b|)$?
  - L-am obținut adăugând/șterg ând/înlocuind un caracter. Dacă ne putem da seama ce operație am folosit, atunci putem face reconstrucția.

Cum ne putem da seama de asta?

Reconstrucția pașilor

- Cum am obținut $lev_{a,b}(|a|, |b|)$?
  - L-am obținut adăugând/șterg ând/înlocuind un caracter. Dacă ne putem da seama ce operație am folosit, atunci putem face reconstrucția. Putem să ne dăm seama de asta luând cele trei posibilități și verificând care dintre acestea este cea care ne-a oferit costul curent. Astfel, vom pleca de la $lev_{a,b}(|a|, |b|)$ și ne vom muta înapoi până o să ajungem la transfromarea de la șirul vid la șirul vid (cazul trivial).

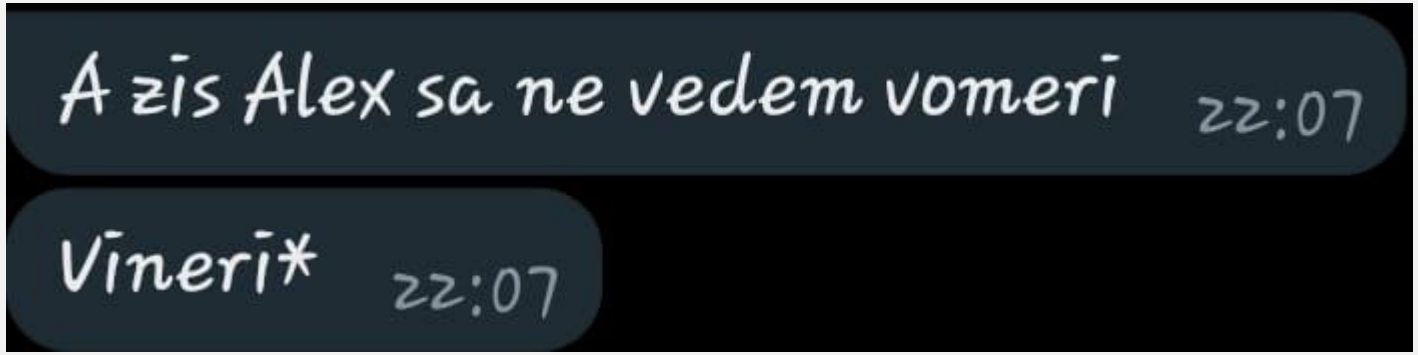
Cum ne putem da seama din ce caz venim?

Reconstrucția pașilor

- Cum am obținut $lev_{a,b}(|a|, |b|)$?
  - L-am obținut adăugând/șterg ând/înlocuind un caracter. Dacă ne putem da seama ce operație am folosit, atunci putem face reconstrucția. Putem să ne dăm seama de asta luând cele trei posibilități și verificând care dintre acestea este cea care ne-a oferit costul curent. Astfel, vom pleca de la $lev_{a,b}(|a|, |b|)$ și ne vom muta înapoi până o să ajungem la transfromarea de la șirul vid la șirul vid (cazul trivial).
    - Venim de sus ⇒ am șters caracterul i
    - Venim din stânga ⇒ am adăugat caracterul j
    - Venim pe diagonală ⇒ se efectuează o potențială înlocuire a caracterului i cu caracterul j.

Exemplu

Din grabă ați scris greșit cuvântul vineri. Totuși, după ce v-ați corectat greșeala, vă întrebați cât este distanța de editare între vomeri și vineri. Deci, cum o calculăm?



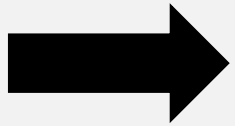
Exemplu

Din grabă ați scris greșit cuvântul vineri. Totuși, după ce v-ați corectat greșeala, vă întrebați cât este distanța de editare între vomeri și vineri. Deci, cum o calculăm?

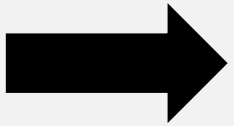
Observație: În rezolvare considerăm cuvintele având doar litere mici.

Exemplu

		v	i	n	e	r	i
v							
o							
m							
e							
r							
i							



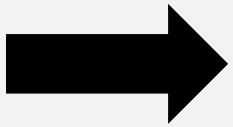
		v	i	n	e	r	i
	0	1	2	3	4	5	6
v	1						
o	2						
m	3						
e	4						
r	5						
i	6						



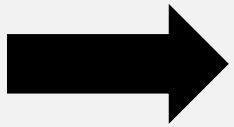
		v	i	n	e	r	i
	0	1	2	3	4	5	6
v	1	0					
o	2						
m	3						
e	4						
r	5						
i	6						

Exemplu

		v	i	n	e	r	i
	0	1	2	3	4	5	6
v	1	0	1				
o	2						
m	3						
e	4						
r	5						
i	6						



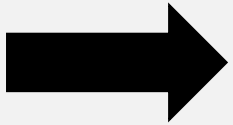
		v	i	n	e	r	i
	0	1	2	3	4	5	6
v	1	0	1	2			
o	2						
m	3						
e	4						
r	5						
i	6						



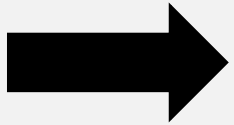
		v	i	n	e	r	i
	0	1	2	3	4	5	6
v	1	0	1	2	3		
o	2						
m	3						
e	4						
r	5						
i	6						

Exemplu

		v	i	n	e	r	i
	0	1	2	3	4	5	6
v	1	0	1	2	3	4	
o	2						
m	3						
e	4						
r	5						
i	6						



		v	i	n	e	r	i
	0	1	2	3	4	5	6
v	1	0	1	2	3	4	5
o	2						
m	3						
e	4						
r	5						
i	6						

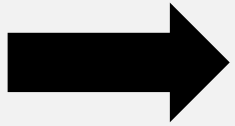


		v	i	n	e	r	i
	0	1	2	3	4	5	6
v	1	0	1	2	3	4	5
o	2	1					
m	3						
e	4						
r	5						
i	6						

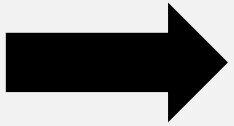
Exemplu

—

		v	i	n	e	r	i
	0	1	2	3	4	5	6
v	1	0	1	2	3	4	5
o	2	1	1				
m	3						
e	4						
r	5						
i	6						



		v	i	n	e	r	i
	0	1	2	3	4	5	6
v	1	0	1	2	3	4	5
o	2	1	1	2	3	4	5
m	3						
e	4						
r	5						
i	6						

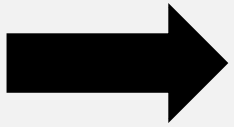


		v	i	n	e	r	i
	0	1	2	3	4	5	6
v	1	0	1	2	3	4	5
o	2	1	1	2	3	4	5
m	3	2	2	2	3	4	5
e	4						
r	5						
i	6						

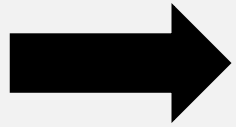
Exemplu

—

		v	i	n	e	r	i
	0	1	2	3	4	5	6
v	1	0	1	2	3	4	5
o	2	1	1	2	3	4	5
m	3	2	2	2	3	4	5
e	4	3	3	3	2	3	4
r	5						
i	6						

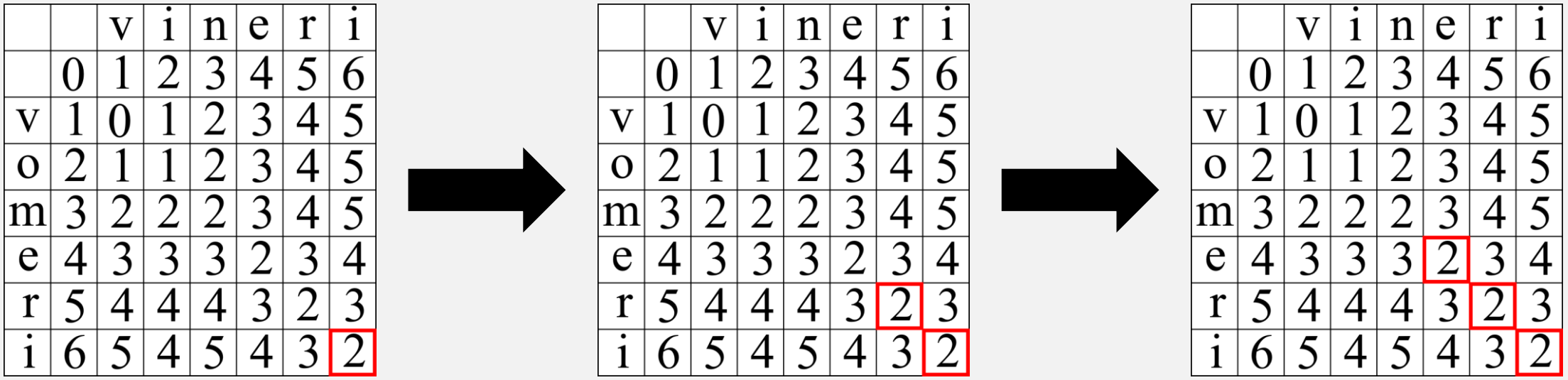


		v	i	n	e	r	i
	0	1	2	3	4	5	6
v	1	0	1	2	3	4	5
o	2	1	1	2	3	4	5
m	3	2	2	2	3	4	5
e	4	3	3	3	2	3	4
r	5	4	4	4	3	2	3
i	6						



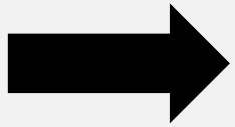
		v	i	n	e	r	i
	0	1	2	3	4	5	6
v	1	0	1	2	3	4	5
o	2	1	1	2	3	4	5
m	3	2	2	2	3	4	5
e	4	3	3	3	2	3	4
r	5	4	4	4	3	2	3
i	6	5	4	5	4	3	2

Reconstrucția pașilor



Reconstrucția pașilor

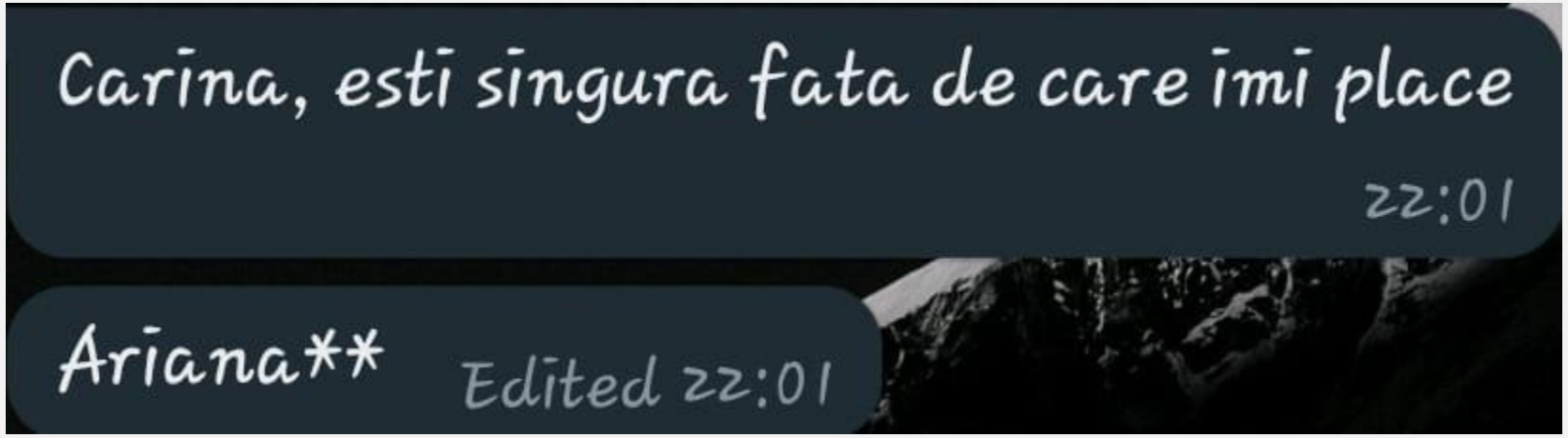
		v	i	n	e	r	i
	0	1	2	3	4	5	6
v	1	0	1	2	3	4	5
o	2	1	1	2	3	4	5
m	3	2	2	2	3	4	5
e	4	3	3	3	2	3	4
r	5	4	4	4	3	2	3
i	6	5	4	5	4	3	2



		v	i	n	e	r	i
	0	1	2	3	4	5	6
v	1	0	1	2	3	4	5
o	2	1	1	2	3	4	5
m	3	2	2	2	3	4	5
e	4	3	3	3	2	3	4
r	5	4	4	4	3	2	3
i	6	5	4	5	4	3	2

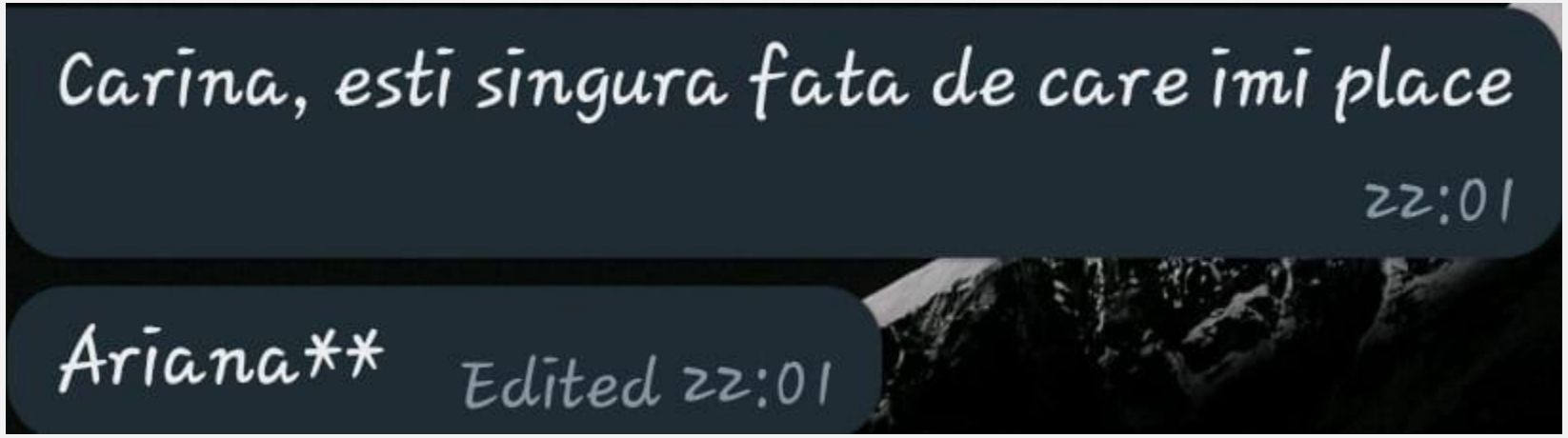
Exemplu 2

Sunteți într-un caz foarte nefericit și ați scris greșit numele iubitei voastre Ariana. Totuși, informaticieni fiind, vă întrebați cât este distanța de editare între Carina și Ariana. Deci, cum o calculăm?



Exemplu 2

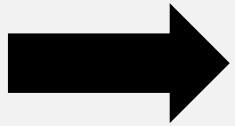
Sunteți într-un caz foarte nefericit și ați scris greșit numele iubitei voastre Ariana. Totuși, informaticieni fiind, vă întrebați cât este distanța de editare între Carina și Ariana. Deci, cum o calculăm?



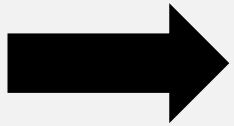
Observație: În rezolvare considerăm cuvintele având doar litere mici.

Exemplu 2

		a	r	i	a	n	a
c							
a							
r							
i							
n							
a							



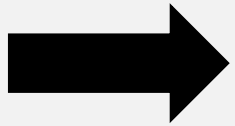
		a	r	i	a	n	a
	0	1	2	3	4	5	6
c	1						
a	2						
r	3						
i	4						
n	5						
a	6						



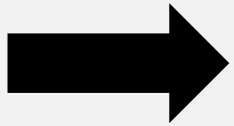
		a	r	i	a	n	a
	0	1	2	3	4	5	6
c	1	1	2	3	4	5	6
a	2						
r	3						
i	4						
n	5						
a	6						

Exemplu 2

		a	r	i	a	n	a
	0	1	2	3	4	5	6
c	1	1	2	3	4	5	6
a	2	1	2	3	3	4	5
r	3						
i	4						
n	5						
a	6						

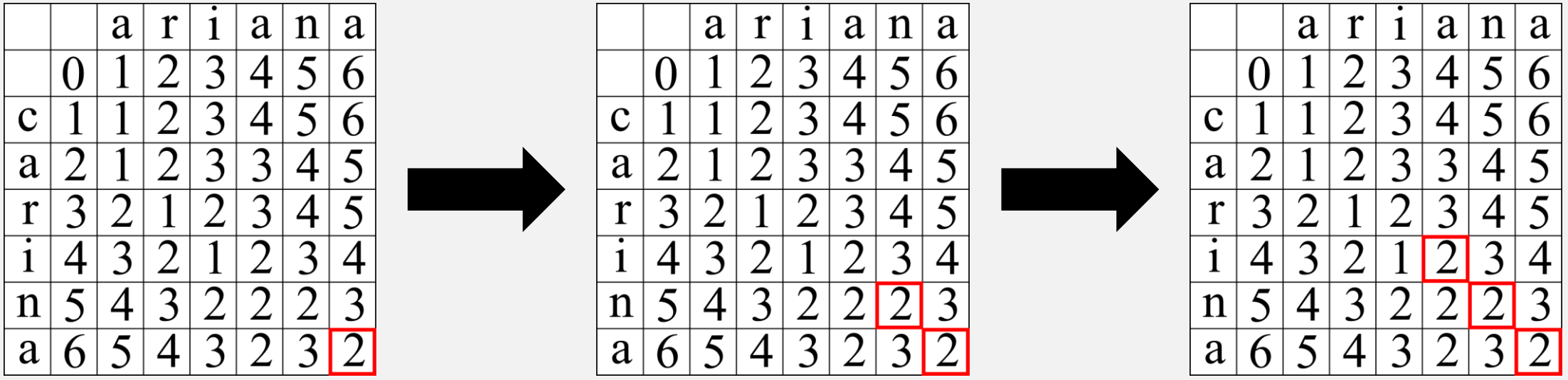


		a	r	i	a	n	a
	0	1	2	3	4	5	6
c	1	1	2	3	4	5	6
a	2	1	2	3	3	4	5
r	3	2	1	2	3	4	5
i	4	3	2	1	2	3	4
n	5						
a	6						

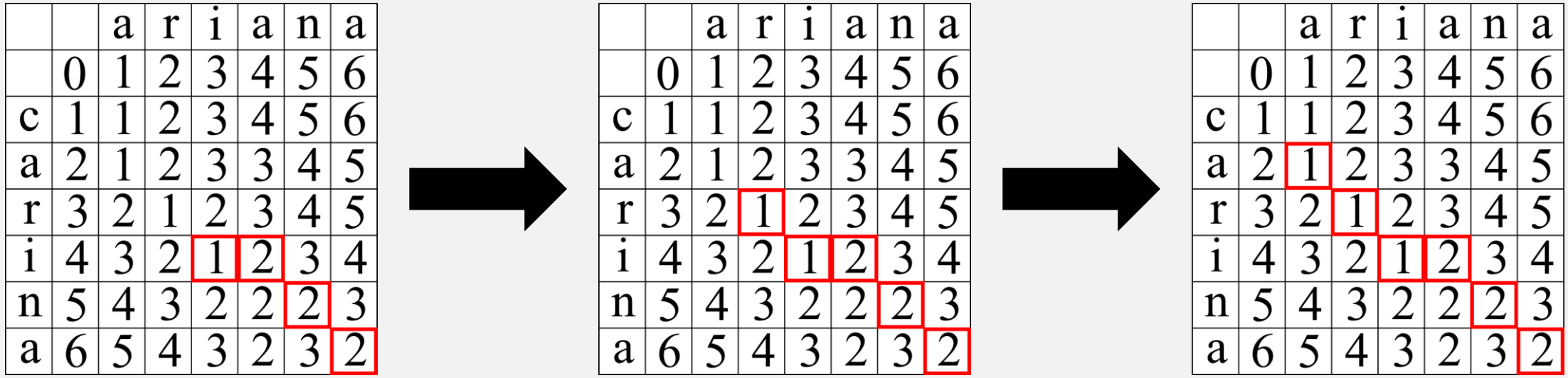


		a	r	i	a	n	a
	0	1	2	3	4	5	6
c	1	1	2	3	4	5	6
a	2	1	2	3	3	4	5
r	3	2	1	2	3	4	5
i	4	3	2	1	2	3	4
n	5	4	3	2	2	2	3
a	6	5	4	3	2	3	2

Reconstrucția pașilor

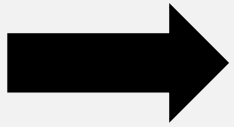


Reconstrucția pașilor



Reconstrucția pașilor

		a	r	i	a	n	a
	0	1	2	3	4	5	6
c	1	1	2	3	4	5	6
a	2	1	2	3	3	4	5
r	3	2	1	2	3	4	5
i	4	3	2	1	2	3	4
n	5	4	3	2	2	2	3
a	6	5	4	3	2	3	2



		a	r	i	a	n	a
	0	1	2	3	4	5	6
c	1	1	2	3	4	5	6
a	2	1	2	3	3	4	5
r	3	2	1	2	3	4	5
i	4	3	2	1	2	3	4
n	5	4	3	2	2	2	3
a	6	5	4	3	2	3	2

Complexitate

		a	r	i	a	n	a
	1.7k	2.4k	912	292	72	12	1
c	2.4k	1.7k	681	231	61	11	1
a	912	681	321	129	41	9	1
r	292	231	129	63	25	7	1
i	72	61	41	25	13	5	1
n	12	11	9	7	5	3	1
a	1	1	1	1	1	1	1

- Numărul în albastru reprezintă numărul de apeluri ale funcției $lev_{a,b}(i,j)$ După cum putem observa, valorile cresc destul de repede, ajungându-se la valori de ordinul miilor cu doar 6 caractere pe cel puțin un cuvânt. Pentru aproximativ 12 caractere vom obține un număr de apeluri de ordinul zecilor de milioane.

Complexitate

		a	r	i	a	n	a
	1.7k	2.4k	912	292	72	12	1
c	2.4k	1.7k	681	231	61	11	1
a	912	681	321	129	41	9	1
r	292	231	129	63	25	7	1
i	72	61	41	25	13	5	1
n	12	11	9	7	5	3	1
a	1	1	1	1	1	1	1

- Numărul în albastru reprezintă numărul de apeluri ale funcției $lev_{a,b}(i,j)$ După cum putem observa, valorile cresc destul de repede, ajungându-se la valori de ordinul miilor cu doar 6 caractere pe cel puțin un cuvânt. Pentru aproximativ 12 caractere vom obține un număr de apeluri de ordinul zecilor de milioane.

Cum putem îmbunătăți complexitatea?

Complexitate

- Putem observa faptul că $lev_{a,b}(i,j)$ nu depinde de ce s-a întâmplat în trecut.
- Prin urmare, putem calcula valoarea pentru $lev_{a,b}(i,j)$ o singură dată și după doar să folosim valoarea deja calculată.
- Complexitatea se îmbunătățește astfel de la o funcție exponențială la una polinomială, mai exact $\mathrm{O}(|a| * |b|)$.
- Tehnica folosită pentru îmbunătățirea complexității se numește memoizare.

Aplicații practice

- autocorector;
- verificarea plagiatului;
- sisteme de corecție pentru optical character recognition (OCR);
- software pentru natural-language translation;
- diferențele dintre două limbi;