# Potrivirea şirurilor

**Date de intrare**
Două şiruri: un model şi un text în care se caută modelul.
Deseori, acestea apar sub forma unor şiruri de caractere.

**Date de ieşire**
Toate poziţiile (deplasamente) la care modelul se potriveşte sau, cu alte cuvinte, „se suprapune perfect” cu textul.

**Convenţii**
`T` = text (text), indexat de la `1`
`P` = model (pattern), indexat de la `1`
`T, P` sunt dintr-un alfabet fixat
`n` = lungimea textului
`m` = lungimea modelului
$1 \le m \le n \le 2000000$

**Deplasament**
Modelul se află la deplasamentul `d` dacă potrivirea se verifică începând cu al `d+1`-lea element din text:

```
T = A B C A B C A B
    • • • C D F          e la deplasamentul 3
    • • • • • C D F      e la deplasamentul 5
```

**Subsecvenţă a unui şir**: elementele dintre doi indici din şir: $S_{2,3}(INFO) = NF$
**Prefixul de lungime k al unui şir**: primele $k$ elemente din şir: $P_2(CEX) = CE$
**Sufixul de lungime k al unui şir**: ultimele $k$ elemente din şir: $S_1(ONI) = I$
**Prefix-Sufix al unui şir**: prefix care este şi sufix în şir: `ABA` în `ABABA`.
**P / S / P-S strict al unui şir**: de lungime mai mică decât lungimea şirului.
**P / S / P-S maximal al unui şir**: nu există altul mai lung în şir.

| Algoritmi | Timp | Spaţiu | Avantaje | Dezavantaje | Recomandare |
|---|---|---|---|---|---|
| naiv | `m(n-m+1)` | `n+m` | uşor de scris şi intuitiv | extrem de lent | ☺☺ |
| Rabin Karp | `n+m` în medie | `n+m` | uşor de scris şi rapid | lent pentru unele intrări | ☺☺☺ |
| Knuth Morris Pratt | `n+m` | `n+m` | rapid întotdeauna | mai greu de scris | ☺☺☺☺☺ |

**Algoritmul naiv**
Verifică pentru fiecare deplasament dacă modelul se găseşte în text.

```
pentru d ← 0, n-m execută
    pentru p ← 1, m execută
        dacă T[d+p] ≠ P[p] atunci
            întrerupe
    dacă p = m+1 atunci
        scrie d, "e corect"
```

**Algoritmul Rabin Karp**
Calculează în $H_P$ pentru şirul model o funcţie $H$.
Pentru fiecare subsecvenţă $S$ de lungime $m$, calculează în $H_S$ funcţia $H$ şi testează potrivirea numai dacă $H_P=H_S$.

```
află H_P pt. modelul P
află H_S pt. subsecv. de la deplas. 0
pentru d ← 0, n-m execută
    dacă H_P = H_S atunci
•       pentru p ← 1, m execută
•           dacă T[d+p] ≠ P[p] atunci
•               întrerupe
•       dacă p = m+1 atunci
•           scrie d, "e corect"
    află H_S pentru subsecvenţa următoare
```

$H$ poate fi definită prin funcţiile $H_1, H_2, ..., H_k$ (în practică, $k \le 2$) şi se face testul de egalitate pe toate funcţiile $H_i$.
Dacă $H$ este bine aleasă, nu se mai execută •, ci direct •.

**Funcţia prefix**
Calculează în `L[i]` lungimea celui mai lung prefix-sufix strict maximal în prefixul de lungime `i` al modelului.

```
L[1] ← 0
pentru p ← 2, M execută
    L[p] ← L[p-1]
    cât timp L[p] şi P[L[p]+1] ≠ P[p] execută
        L[p] ← L[L[p]]
    dacă P[L[p]+1] = P[p] atunci
        L[p] ← L[p]+1
```

**Algoritmul Knuth Morris Pratt**
Folosind informaţiile reţinute în vectorul `L`, se sar deplasamente care nu ar duce la găsirea unei potriviri.

```
k ← 0
pentru t ← 1, N execută
    cât timp k şi P[k+1] ≠ T[t] execută
        k ← L[k]
    dacă P[k+1] = T[t] atunci
        k ← k+1
    dacă k = m atunci
        scrie t-m, "e corect"
        k ← L[k]
```

**Referinţe**
↗ Articol detaliat despre algoritmul Knuth Morris Pratt
↗ Probleme care se rezolvă cu algoritmi de potrivire a şirurilor
↗ Clasificare a algoritmilor de potrivire a şirurilor şi alte referinţe