Gianluca Della Vedova

- Elementi di Bioinformatica

Gianluca Della Vedova

- Elementi di Bioinformatica
- Ufficio U14-2041

Gianluca Della Vedova

- Elementi di Bioinformatica
- Ufficio U14-2041
- https://www.unimib.it/gianluca-della-vedova

Gianluca Della Vedova

- Elementi di Bioinformatica
- Ufficio U14-2041
- https://www.unimib.it/gianluca-della-vedova
- gianluca.dellavedova@unimib.it

Gianluca Della Vedova

- Elementi di Bioinformatica
- Ufficio U14-2041
- https://www.unimib.it/gianluca-della-vedova
- gianluca.dellavedova@unimib.it
- https://github.com/bioinformatica-corso/programmi-elementi-bioinformatica

Gianluca Della Vedova

- Elementi di Bioinformatica
- Ufficio U14-2041
- https://www.unimib.it/gianluca-della-vedova
- gianluca.dellavedova@unimib.it
- https://github.com/bioinformatica-corso/programmi-elementi-bioinformatica
- https://github.com/bioinformatica-corso/lezioni

Notazione

- simbolo: $T[i]$

Notazione

- simbolo: $T[i]$
- stringa: $T[1]T[2] \cdots T[l]$

Notazione

- simbolo: $T[i]$
- stringa: $T[1]T[2]\cdots T[l]$
- sottostringa: $T[i:j]$

Notazione

- simbolo: $T[i]$
- stringa: $T[1]T[2] \cdots T[l]$
- sottostringa: $T[i : j]$
- prefisso: $T[: j] = T[1 : j]$

Notazione

- simbolo: $T[i]$
- stringa: $T[1]T[2] \cdots T[l]$
- sottostringa: $T[i : j]$
- prefisso: $T[: j] = T[1 : j]$
- suffisso: $T[i :] = T[i : |T|]$

Notazione

- simbolo: $T[i]$
- stringa: $T[1]T[2] \cdots T[l]$
- sottostringa: $T[i : j]$
- prefisso: $T[: j] = T[1 : j]$
- suffisso: $T[i :] = T[i : |T|]$
- concatenazione: $T_1 \cdot T_2 = T_1 T_2$

Pattern Matching

Problema

Input: testo $T = T[1] \cdots T[n]$, pattern $P = P[1] \cdots P[m]$, alfabeto Σ
Goal: trovare *tutte* le occorrenze di P in T
Goal: trovare tutti gli i tale che $T[i] \cdots T[i + m - 1] = P$

Pattern Matching

Problema

Input: testo $T = T[1] \cdots T[n]$, pattern $P = P[1] \cdots P[m]$, alfabeto Σ
Goal: trovare *tutte* le occorrenze di P in T
Goal: trovare tutti gli i tale che $T[i] \cdots T[i + m - 1] = P$

Algoritmo banale

Tempo: $O(nm)$

Pattern Matching

Problema

Input: testo $T = T[1] \cdots T[n]$, pattern $P = P[1] \cdots P[m]$, alfabeto Σ
Goal: trovare *tutte* le occorrenze di P in T
Goal: trovare tutti gli i tale che $T[i] \cdots T[i + m - 1] = P$

Algoritmo banale

Tempo: $O(nm)$

Lower bound

Tempo: $O(n + m)$

Bit-parallel

Algoritmi seminumerici

Bit-parallel

Algoritmi seminumerici

- 25

Bit-parallel

Algoritmi seminumerici

- 25
- 25 = 00011001

Bit-parallel

Algoritmi seminumerici

- 25
- 25 = 00011001
- 25 = 00011001 =FFFTTFFT

Bit-parallel

Algoritmi seminumerici

- 25
- 25 = 00011001
- 25 = 00011001 =FFFTTFFT

Operazioni bit-level

Or: $x \vee y$, And: $x \wedge y$, Xor: $x \oplus y$
Left Shift: $x << k$, Right Shift: $x >> k$,

Bit-parallel

Algoritmi seminumerici

- 25
- 25 = 00011001
- 25 = 00011001 =FFFTTFFT

Operazioni bit-level

Or: $x \vee y$, And: $x \wedge y$, Xor: $x \oplus y$
Left Shift: $x << k$, Right Shift: $x >> k$,

- Tutte bitwise

Bit-parallel

Algoritmi seminumerici

- 25
- 25 = 00011001
- 25 = 00011001 =FFFTTFFT

Operazioni bit-level

Or: $x \vee y$, And: $x \wedge y$, Xor: $x \oplus y$
Left Shift: $x << k$, Right Shift: $x >> k$,

- Tutte bitwise
- Tutte in hardware

Dömölki / Baeza-Yates, Gonnet

Matrice M

$M(i, j) = 1$ sse $P[: i] = T[j - i + 1 : j]$
$0 \leqslant i \leqslant m, 0 \leqslant j \leqslant n$

Dömölki / Baeza-Yates, Gonnet

Matrice M

$M(i, j) = 1$ sse $P[: i] = T[j - i + 1 : j]$
$0 \leqslant i \leqslant m, 0 \leqslant j \leqslant n$

Occorrenza di P in T

$M(m, \cdot) = 1$

Dömölki / Baeza-Yates, Gonnet

Matrice M

$M(i, j) = 1$ sse $P[: i] = T[j - i + 1 : j]$
$0 \leqslant i \leqslant m, 0 \leqslant j \leqslant n$

Occorrenza di P in T

$M(m, \cdot) = 1$

- $M(0, \cdot) = 1, M(\cdot, 0) = 0$

Dömölki / Baeza-Yates, Gonnet

Matrice M

$M(i,j) = 1$ sse $P[:i] = T[j-i+1:j]$
$0 \leqslant i \leqslant m, 0 \leqslant j \leqslant n$

Occorrenza di P in T

$M(m,\cdot) = 1$

- $M(0,\cdot) = 1, M(\cdot,0) = 0$
- $M(i,j) = 1$ sse $M(i-1,j-1) = 1$ AND $P[i] = T[j]$

Esempio

Esempio

T=abracadabra
P=abr

10010101001
01000000100
00100000010 ← occorrenze

Esempio

Esempio

T=abracadabra
P=abr

10010101001
01000000100
00100000010 ← occorrenze

Matrice M

1 colonna = 1 numero

Colonne

$U[\sigma]$ = array di bit dove $U[\sigma, i] = 1$ sse $P[i] = \sigma$

C[j] da C[j − 1]

Colonne

$U[\sigma]$ = array di bit dove $U[\sigma, i] = 1$ sse $P[i] = \sigma$

C[j] da C[j − 1]

- Right shift di C[j − 1]

Colonne

$U[\sigma]$ = array di bit dove $U[\sigma, i] = 1$ sse $P[i] = \sigma$

C[j] da C[j − 1]

- Right shift di C[j − 1]
- 1 in prima posizione

Colonne

$U[\sigma]$ = array di bit dove $U[\sigma, i] = 1$ sse $P[i] = \sigma$

C[j] da C[j − 1]

- Right shift di C[j − 1]
- 1 in prima posizione
- AND con U[T[j]]

Colonne

$U[\sigma]$ = array di bit dove $U[\sigma, i] = 1$ sse $P[i] = \sigma$

C[j] da C[j − 1]

- Right shift di C[j − 1]
- 1 in prima posizione
- AND con U[T[j]]
- ω: word size

Colonne

$U[\sigma]$ = array di bit dove $U[\sigma, i] = 1$ sse $P[i] = \sigma$

C[j] da C[j − 1]

- Right shift di C[j − 1]
- 1 in prima posizione
- AND con U[T[j]]
- ω: word size
- C[j] = ((C[j − 1] >> 1) | (1 << (ω − 1))) & U[T[j]];

Note

- Tempo $O(n)$ se $m \leqslant \omega$

Note

- Tempo $O(n)$ se $m \leqslant \omega$
- Tempo $O(nm)$

Note

- Tempo $O(n)$ se $m \leqslant \omega$
- Tempo $O(nm)$
- No condizioni

Note

- Tempo $O(n)$ se $m \leqslant \omega$
- Tempo $O(nm)$
- No condizioni
- $\omega < m \leqslant 2\omega$?

Licenza d'uso