

Knuth-Morris-Pratt algorithm

Cosmin IVAN
Mihai MALOS
Livia TOMA

Cum funcționează algoritmul?

NAIVE VS KMP

Scopul proiectului

Crearea unui program care poate cenzura datele personale ce apar în componența unui document, în cazul necesității publicării documentului respectiv.

Cu această ocazie avem posibilitatea de a prezenta și o metodă eficientă de căutare a unei secvențe într-un text.

Aplicații ale algoritmului KMP

- >> Verificarea plagiaturii în documente
- >> Bioinformatică și secvantarea ADN-ului
- >> Criminalistica digitală
- >> Verificarea ortografiei
- >> Filtre de spam
- >> Motoare de căutare
- >> Sistem de detectare a intruziunilor

CONCLUZII

- >> Complexitatea algoritmului este O(n²)
- >> Având în vedere că este realizată pe parcursul unei permutări caracterelor – incrementarea lui i și comparația cu subsecvența începând cu pozitia i+1, apoi se mută j la (P[i]+1) și se repeta procesul și așa delung, împărțindu-se secvența în părți mai mici

Knuth, Morris și Pratt

Motivul pentru care acest algoritm a fost denumit după cei trei oameni ai cărornei Donald Knuth, Jim Morris și Vaughan Pratt este faptul că, individual, fiecare dintre autori programul au descoperit metoda, pornind de la ipoteze sau probleme de rezolvat diferite.

Caracteristicile algoritmului KMP

- >> Algoritmul de căutare a parcurgării literelor într-un text nu se găsește în text mult mai mult.
- >> "String matching" este o metodă care oferă o soluție rezabilă la problema de căutare a unor subsecvențe într-un alt text.
- >> Algoritmul KMP este o variantă a algoritmului de căutare a unor subsecvențe, bazată pe comparația caracterelor din secvență și subsecvență.
- >> Complexitatea este de ordinul O(n+m), unde n este lungimea subsecvenței și m este lungimea textului, faza de căutare are complexitatea de timp O(1).

Scopul proiectului

Crearea unui program care poate cenzura datele personale ce apar în componența unui document, în cazul necesității publicării documentului respectiv.

Cu această ocazie avem posibilitatea de a prezenta și o metodă eficientă de căutare a unei secvențe într-un text.

Knuth, Morris și Pratt

- Motivul pentru care acest algoritm a fost denumit după cei trei oameni ai științei Donald Knuth, Jim Morris și Vaughan Pratt este faptul că, individual, fiecare dintre autorii programului au descoperit metoda, pornind de la ipoteze sau probleme de rezolvat diferite.

Caracteristicile algoritmului KMP

- >> Algoritmii de căutare a sirurilor încearcă să găsească un loc în care un sir se găsește într-un text mai mare.
- >> „String matching” este o metodă care oferă o soluție fezabilă la o problemă de căutare a sirurilor.
- >> Knuth, Morris și Pratt au conceput un algoritm de potrivire exactă a sirurilor, bazat pe compararea caracterelor
- >> Comparațiile sunt efectuate caracter cu caracter, de la stânga la dreapta.
- >> Complexitatea pentru faza de preprocessare este $O(m)$
- >> Indiferent de sistemul de semne, faza de căutare are complexitate de timp $O(n + m)$.

Cum funcționează algoritmul?

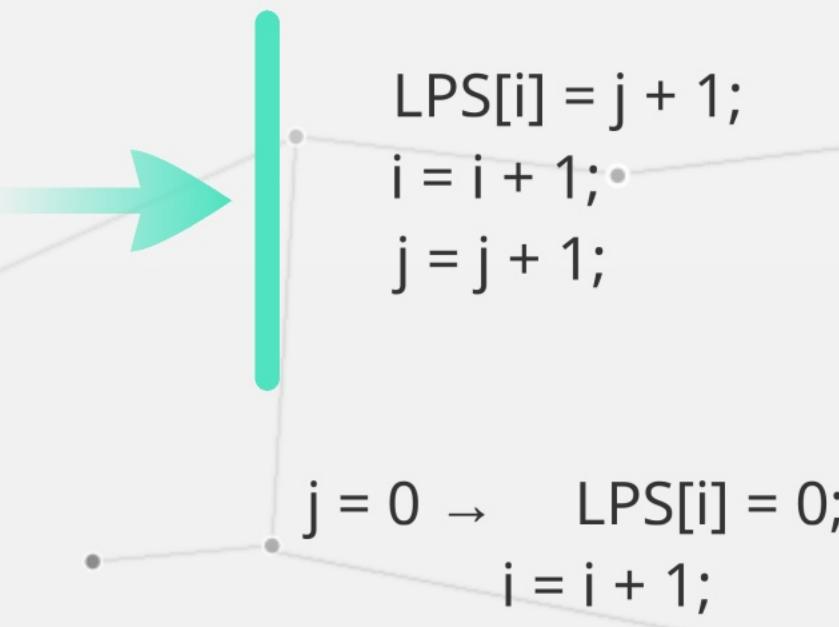
Partea de pre-procesare a algoritmului KMP: Vectorul LPS (longest proper prefix that is also a suffix) .

pattern - string de dimensiune m

Folosind poziția fiecărui caracter din cuvântul dat, în afară de elementul $LPS[0]$, se efectuează comparații între $pattern[i]$ și $pattern[j]$.

`pattern[j] = pattern[i]`

`pattern[j] ≠ pattern[i] &&`



Toate aceste condiții se repetă până când toate caracterele din „pattern” au fost parcurse.

	0	1	2	3	4	5	6	7	8
pattern	a	b	c	d	a	b	c	d	e
LPS									

Construirea vectorului LPS pentru cuvântul „abcdabcde”, de lungime m=9.

The KMP function

PATTERN

Size = m

TEXT

Size = n

```
j = m  
{  
    show i - m ;  
}
```



```
pattern[ j ] = text[ i ]  
{  
    i = i + 1;  
    j = j + 1;  
}
```

```
( i < n ) && ( pattern[j] ≠ text[i] ) &&  
    ( j = 0 ) → i = i + 1;  
    ( j ≠ 0 ) → j = LPS[j - 1];
```

n=21

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
text	a	b	a	b	c	d	a	d	e	g	r	a	b	c	d	a	b	c	d	e	t

m = 9

	0	1	2	3	4	5	6	7	8
pattern	a	b	c	d	a	b	c	d	e
LPS	0	0	0	0	1	2	3	4	0

Knuth-Morris-Pratt algorithm

Cosmin IVAN
Mihai MALOS
Livia TOMA

Cum funcționează algoritmul?

NAIVE VS KMP

Scopul proiectului

Crearea unui program care poate cenzura datele personale ce apar în componența unui document, în cazul necesității publicării documentului respectiv.

Cu această ocazie avem posibilitatea de a prezenta și o metodă eficientă de căutare a unei secvențe într-un text.

Aplicații ale algoritmului KMP

- >> Verificarea plagiaturii în documente
- >> Bioinformatică și secvantarea ADN-ului
- >> Criminalistica digitală
- >> Verificarea ortografiei
- >> Filtre de spam
- >> Motoare de căutare
- >> Sistem de detectare a intruziunilor

CONCLUZII

- >> Complexitatea algoritmului este O(n²)
- >> Având în vedere că este realizată pe parcursul a patru buclă: > patrula caracterelor – incrementarea lui i și patrula lui j și patrula lui k și patrula lui l și se compara

Knuth, Morris și Pratt

Motivul pentru care acest algoritm a fost denumit după cei trei oameni ai cărornei Donald Knuth, Jim Morris și Vaughan Pratt este faptul că, individual, fiecare dintre autori programul au descoperit metoda, pornind de la ipoteze sau probleme de rezolvat diferite.

Caracteristicile algoritmului KMP

- >> Algoritmul de căutare a patrurilor folosește algoritmul de căutare într-un text mai mare.
- >> "String matching" este o metodă care oferă o soluție rezabilă la problema de căutare a unor subsecvențe într-un alt text.
- >> Algoritmul KMP este o variantă a algoritmului de căutare într-un text, bazată pe comparația caracterelor consecutive exactă a patrurilor, bazată pe comparația caracterelor consecutive.
- >> Complexitatea este de $O(n + m)$, unde n este lungimea textului și m este lungimea subsecvenței.
- >> Complexitatea pentru lărgirea expresiunii este $O(m)$.
- >> Implementările sunt eficiente și ușoare de realizat.
- >> Folosirea unei tabele de căutare, baza de căutare are complexitatea de timp $O(1)$ + n.

NAIVE VS KMP

Folosind algoritmul de căutare Naive, se testează fiecare poziție posibilă a pattern-ului de string-uri, pattern [1 ... m], în textul dat, text [1 ... n].

Factorul care face această metodă ineficientă este faptul că se efectuează comparații care nu sunt necesare, încrucișând, în mod repetat, se caută o potrivire a elementelor deja parcuse.

No.	Algorithm	Best Case	Average Case	Worst Case
1	Naive String Searching	$O(n)$	$O((n-m+1)*m)$	$O(n*m)$
2	Rabin Karp	$O(n+m)$	$O(n+m)$	$O(n*m)$
3	KMP	$O(n+m)$	$O(n+m)$	$O(n+m)$

Asemănări și deosebiri

>> Pentru a se putea realiza comparatia se folosesc doi pointeri de tipul string
>> Se folosesc doi iteratori care compara caracter cu caracter de la stanga la dreapta.
>> Se identifică un pattern de caractere într-un text primit, de cele mai multe ori returnând poziția de început a pattern-ului găsit în text, iar dacă nu se găsește niciodată se returnează valoarea -1

KMP

Este eficient deoarece folosește preprocessarea pentru a evita munca suplimentară.

Are nevoie de memorie suplimentară și de preprocessare.

Complexitatea este mereu aceeași $O(n+m)$.

Este foarte complex, ceea ce prezintă o dificultate ridicată în a fi înțeles.

Naive

Este ineficientă datorită potrivirilor ce nu sunt salvate și refolosite, ajungând să muncească suplimentar.

Nu are nevoie de memorie suplimentară și nici de preprocessare.

Prezinta un cel mai rău și bun caz, complexitatea fiind variabilă după caz.

Este ușor de înțeles .

>> Pentru a se putea realiza comparatia se folosesc doi pointeri de tipul string

>> Se folosesc doi iteratori care compara caracter cu caracter de la stanga la dreapta.

>>Se identifică un pattern de caractere într-un text primit, de cele mai multe ori returnând poziția de început a pattern-ului găsit în text, iar dacă nu se găsește niciodată se returnează valoarea -1

Este eficiență
preprocesare
muncă

Are nevoie
suplimentară

Complexitate

Este foarte
rezistentă
prezintă o
a

KMP

Este eficient deoarece folosește preprocesarea pentru a evita munca suplimentară.

Are nevoie de memorie suplimentară și de preprocesare.

Complexitatea este mereu aceeași $O(n+m)$.

Este foarte complex, ceea ce prezintă o dificultate ridicată în a fi înțeles.

Naive

Este ineficientă datorită potrivirilor ce nu sunt salvate și refolosite, ajungând să muncească suplimentar.

Nu are nevoie de memorie suplimentară și nici de preprocesare.

Prezinta un cel mai rău și bun caz, complexitatea fiind variabilă după caz.

Este ușor de înțeles .

Asemănări și deosebiri

>> Pentru a se putea realiza comparatia se folosesc doi pointeri de tipul string
>> Se folosesc doi iteratori care compara caracter cu caracter de la stanga la dreapta.
>> Se identifică un pattern de caractere într-un text primit, de cele mai multe ori returnând poziția de început a pattern-ului găsit în text, iar dacă nu se găsește niciodată se returnează valoarea -1

KMP

Este eficient deoarece folosește preprocessarea pentru a evita munca suplimentară.

Are nevoie de memorie suplimentară și de preprocessare.

Complexitatea este mereu aceeași $O(n+m)$.

Este foarte complex, ceea ce prezintă o dificultate ridicată în a fi înțeles.

Naive

Este ineficientă datorită potrivirilor ce nu sunt salvate și refolosite, ajungând să muncească suplimentar.

Nu are nevoie de memorie suplimentară și nici de preprocessare.

Prezinta un cel mai rău și bun caz, complexitatea fiind variabilă după caz.

Este ușor de înțeles .

Knuth-Morris-Pratt algorithm

Cosmin IVAN
Mihai MALOS
Livia TOMA

Cum funcționează algoritmul?

NAIVE VS KMP

Scopul proiectului

Crearea unui program care poate cenzura datele personale ce apar în componența unui document, în cazul necesității publicării documentului respectiv.

Cu această ocazie avem posibilitatea de a prezenta și o metodă eficientă de căutare a unei secvențe într-un text.

Aplicații ale algoritmului KMP

- >> Verificarea plagiaturii în documente
- >> Bioinformatică și secvantarea ADN-ului
- >> Criminalistica digitală
- >> Verificarea ortografiei
- >> Filtre de spam
- >> Motoare de căutare
- >> Sistem de detectare a intruziunilor

CONCLUZII

- >> Complexitatea algoritmului este O(n²)
- >> Având în vedere că este realizată pe parcursul a patru buclă: > patrula caracterelor – incrementarea lui i și patrula lui j și patrula lui k și patrula lui l și se compara

Knuth, Morris și Pratt

Motivul pentru care acest algoritm a fost denumit după cei trei oameni ai cărornei Donald Knuth, Jim Morris și Vaughan Pratt este faptul că, individual, fiecare dintre autori programul au descoperit metoda, pornind de la ipoteze sau probleme de rezolvat diferite.

Caracteristicile algoritmului KMP

- >> Algoritmul de căutare a patrurilor folosește algoritmul de căutare într-un text mai mare.
- >> "String matching" este o metodă care oferă o soluție rezabilă la problema de căutare a unor subsecvențe într-un alt text.
- >> Algoritmul KMP este o variantă a algoritmului de căutare într-un text, bazată pe comparația caracterelor patrurilor.
- >> Comparările sunt efectuate caracter cu caracter, de la stânga la dreapta.
- >> Complexitatea pentru hărțu de expresiune constă O(m+n)
- >> Complexitatea pentru căutare într-un text, baza de căutare are complexitatea de timp O(1) + n

Eficiența algoritmului Kmp - distingerea cazurilor

Exemple

KMP este un algoritm mai stabil pentru fiecare situație, iar motivul pentru care îl folosim este acela de a evita cazurile nefavorabile pentru abordarea Naive.

input:

Se generează un text random pe care îl multiplicăm până atingem numărul de 1,5 milioane de cuvinte.

În acest caz, vom căuta cuvântul "Can", rezultatele fiind următoarele:

```
There are 1500000 words
```

```
KMP:2348 matches found  
It took 386 ms
```

```
Naive:2348 matches found  
It took 216 ms
```

Timpul în care KMP a găsit toate aparițiile cuvântului este aproape dublu decât timpul în care algoritmul Naive a rezolvat aceeași cerință.

input:

Se consideră un string compus din 10 caractere identice și ultimul diferit, pe care îl vom clona de 100.000 de mii de ori.

Rezultatele sunt următoarele:

```
There are 100000 words
```

```
KMP:0 matches found  
It took 197 ms
```

```
Naive:0 matches found  
It took 1408 ms
```

Timpul în care KMP a găsit toate aparițiile cuvântului este 7 ori mai mic decât timpul în care algoritmul Naive a rezolvat aceeași cerință.

Eficiența algoritmului Kmp - distingerea cazurilor

Exemple

KMP este un algoritm mai stabil pentru fiecare situație, iar motivul pentru care îl folosim este acela de a evita cazurile nefavorabile pentru abordarea Naive.

KMP este un algoritm mai stabil pentru fiecare situație, iar motivul pentru care îl folosim este acela de a evita cazurile nefavorabile pentru abordarea Naive.

Knuth-Morris-Pratt algorithm

Cosmin IVAN
Mihai MALOS
Livia TOMA

Cum funcționează algoritmul?

NAIVE
VS
KMP

Scopul proiectului

Crearea unui program care poate cenzura datele personale ce apar în componența unui document, în cazul necesității publicării documentului respectiv.

Cu această ocazie avem posibilitatea de a prezenta și o metodă eficientă de căutare a unei secvențe într-un text.

Aplicații ale algoritmului KMP

- >> Verificarea plagiaturii în documente
- >> Bioinformatică și secvantarea ADN-ului
- >> Criminalistica digitală
- >> Verificarea ortografiei
- >> Filtre de spam
- >> Motoare de căutare
- >> Sistem de detectare a intruziunilor

CONCLUZII

- >> Complexitatea algoritmului este O(n²)
- >> Având în vedere că este realizată pe parcursul unei permutări caracterelor – incrementarea lui i și comparația cu subsecvența începând de la pozitia i+1, apoi se mută j la (j+1) și se repeta procesul de la i+1 și se repeta procesul de la i+1 și se compara

Knuth, Morris și Pratt

Motivul pentru care acest algoritm a fost denumit după cei trei oameni ai cărora Donald Knuth, Jim Morris și Vaughan Pratt este faptul că, individual, fiecare dintre autori programul au descoperit metoda, pornind de la ipoteze sau probleme de rezolvat diferite.

Caracteristicile algoritmului KMP

- >> Algoritmul de căutare a stringurilor într-un text este deosebit de eficient.
- >> "String matching" este o metodă care oferă o soluție rezabilă la problema de căutare a unor subsecvențe într-un alt text.
- >> Algoritmul KMP este o variantă a algoritmului de căutare a stringurilor care nu folosește comparații de caractere între stringuri, ci folosește comparații de caractere între subsecvențe.
- >> Complexitatea este de $O(n+m)$, unde n este lungimea textului și m este lungimea stringului căutat.

Aplicații ale algoritmului KMP

- >> Verificarea plagiatului în documente
- >> Bioinformatică și secvențierea ADN-ului
- >> Criminalistica digitală
- >> Verificarea ortografiei
- >> Filtre de spam
- >> Motoare de căutare
- >> Sistem de detectare a intruziunilor

Knuth-Morris-Pratt algorithm

Cosmin IVAN
Mihai MALOS
Livia TOMA

Cum funcționează algoritmul?

NAIVE VS KMP

Scopul proiectului

Crearea unui program care poate cenzura datele personale ce apar în componența unui document, în cazul necesității publicării documentului respectiv.

Cu această ocazie avem posibilitatea de a prezenta și o metodă eficientă de căutare a unei secvențe într-un text.

Aplicații ale algoritmului KMP

- >> Verificarea plagiaturii în documente
- >> Bioinformatică și secvantarea ADN-ului
- >> Criminalistica digitală
- >> Verificarea ortografiei
- >> Filtre de spam
- >> Motoare de căutare
- >> Sistem de detectare a intruziunilor

CONCLUZII

- >> Complexitatea algoritmului este O(n²)
- >> Având în vedere că este realizată pe parcursul a patru buclă: > patrula caracterelor – incrementarea lui i și patrula lui j (j=0..n-i) și incrementarea lui k și patrula lui l și se compara

Knuth, Morris și Pratt

Motivul pentru care acest algoritm a fost denumit după cei trei oameni ai cărornei Donald Knuth, Jim Morris și Vaughan Pratt este faptul că, individual, fiecare dintre autori programul au descoperit metoda, pornind de la ipoteze sau probleme de rezolvat diferite.

Caracteristicile algoritmului KMP

- >> Algoritmul de căutare a patrulei literelor și ghoseșcă un lucru care nu se găsește în text mai mult.
- >> "String matching" este o metodă care oferă o soluție rezabilă la problema de căutare a unor subsecvențe într-un text.
- >> Algoritmul KMP este o variantă a algoritmului de patrulare exactă a patrulei, bazată pe comparația caracterelor care sunt efectuate caracter cu caracter, de la stânga la dreapta.
- >> Complexitatea pentru hărțu de expresiune constă O(m+n)
- >> Complexitatea pentru căutare este O(n), baza de căutare este compuzație de timp O(1) + n

CONCLUZII

- >> Complexitatea algoritmului este $O(m + n)$
- >> Există 3 operații ce pot fi realizate pe parcursul buclei:
 - > potrivirea caracterelor -- incrementarea lui i și j
 - > nepotrivirea caracterelor cu $j > 0$, așadar se mută j la $LPS[j-1]$ și se compară cu i
 - > nepotrivirea caracterelor cu $j = 0$, așadar se incrementează i și se compară

>>Componentele algoritmului KMP:

- >Prefixul
- >Sufixul
- >vectorul LPS

>>Componentele algoritmului KMP:

- >Prefixul
- >Sufixul
- >vectorul LPS

Knuth-Morris-Pratt algorithm

Cosmin IVAN
Mihai MALOS
Livia TOMA

Cum funcționează algoritmul?

NAIVE VS KMP

Scopul proiectului

Crearea unui program care poate cenzura datele personale ce apar în componența unui document, în cazul necesității publicării documentului respectiv.

Cu această ocazie avem posibilitatea de a prezenta și o metodă eficientă de căutare a unei secvențe într-un text.

Aplicații ale algoritmului KMP

- >> Verificarea plagiaturii în documente
- >> Bioinformatică și secvantarea ADN-ului
- >> Criminalistica digitală
- >> Verificarea ortografiei
- >> Filtre de spam
- >> Motoare de căutare
- >> Sistem de detectare a intruziunilor

CONCLUZII

- >> Complexitatea algoritmului este O(n²)
- >> Având în vedere că este realizată pe parcursul a patru buclă: > patrula caracterelor – incrementarea lui i și patrula lui j și patrula lui k și patrula lui l și se compara

Knuth, Morris și Pratt

Motivul pentru care acest algoritm a fost denumit după cei trei oameni ai cărornei Donald Knuth, Jim Morris și Vaughan Pratt este faptul că, individual, fiecare dintre autori programul au descoperit metoda, pornind de la ipoteze sau probleme de rezolvat diferite.

Caracteristicile algoritmului KMP

- >> Algoritmul de căutare a patrurilor folosește algoritmul de căutare într-un text mai mare.
- >> "String matching" este o metodă care oferă o soluție rezabilă la problema de căutare a unor subsecvențe într-un alt text.
- >> Algoritmul KMP este o variantă a algoritmului de căutare într-un text, bazată pe comparația caracterelor consecutive exactă a patrurilor, bazată pe comparația caracterelor consecutive.
- >> Complexitatea este de $O(n + m)$, unde n este lungimea patrurilor și m este lungimea textului.
- >> Complexitatea pentru lărgirea expresiei este $O(m)$.
- >> Implementarea este relativ simplă, baza de căutare este complexitatea de timp $O(1)$.