



## Stof

- slides college 12
- boek 9.1, 9.3

## Twee pattern matching problemen

In de opgaven gebruiken we de twee pattern matching problemen, bestaande uit een tekst  $T$  en een string  $P$ . Probleem 1 is als volgt:

$$\begin{aligned} T &= aaabaadaabaaa \\ P &= aabaaa \end{aligned}$$

Probleem 2 is als volgt:

$$\begin{aligned} T &= 000010001010001 \\ P &= 000101 \end{aligned}$$

## Opgaven

1. (Dit is ongeveer opgave R-9.2.)  
Pas het brute-force pattern matching algoritme toe op het eerste pattern matching probleem.
2. Pas het brute-force pattern matching algoritme toe op het tweede pattern matching probleem.
3. (Dit is ongeveer opgave C-9.1.)  
De worst-case tijdscomplexiteit van het brute-force pattern-matching algoritme is in  $\mathcal{O}(m \cdot (n - m + 1))$ . Geef een voorbeeld van een tekst  $T$  ter lengte  $n$  en een patroon  $P$  ter lengte  $m$  zo dat het brute-force pattern matching algoritme inderdaad  $m \cdot (n - m + 1)$  vergelijkingen doet. Dit illustreert dat de afschatting van  $\mathcal{O}(m \cdot (n - m + 1))$  scherp is.
4. Stel we hebben een patroon  $P$  ter lengte  $m$  met alle  $m$  karakters verschillend. Kunnen we het brute-force pattern matching algoritme optimaliseren voor deze speciale situatie? Wat is de tijdscomplexiteit van het geoptimaliseerde algoritme?

5. \* Stel we staan in het patroon  $P$  het speciale symbool  $*$  toe, dat matcht met een willekeurige string (ter lengte 0 of groter). De  $*$  mag willekeurig vaak in het patroon  $P$  voorkomen, maar nooit in de tekst  $T$ .

Voorbeeld: Het patroon  $P = ab * ba * c$  zit als volgt in de tekst  $T = cabccbacbacab$ :

$$\begin{array}{rcccccccc} T & = & c & ab & cc & ba & cba & c & ab \\ P & = & & ab & * & ba & * & & c \end{array}$$

Geef een pattern matching algoritme voor zulke patronen en teksten, en analyseer de tijdscomplexiteit van je algoritme.

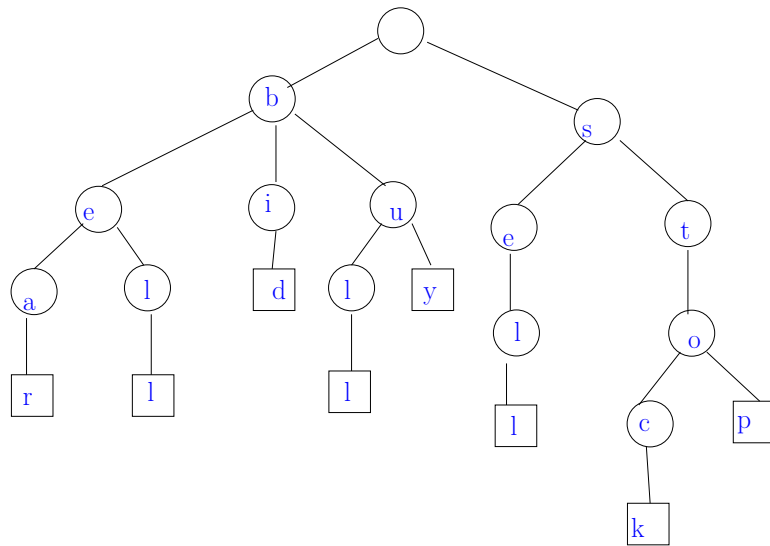
6. (Dit is ongeveer opgave R-9.3.)
- (a) Geef de *last* functie van het Boyer-Moore pattern matching algoritme voor het patroon  $P$  uit het eerste pattern matching probleem, en het alfabet  $\{a, b, c, d\}$ .
- (b) Pas het Boyer-Moore pattern matching algoritme toe op het eerste pattern matching probleem.
7. Herhaal (eventueel) de vorige opgave voor het tweede pattern matching probleem.
8. (Dit is ongeveer opgave R-9.6.)  
Gegeven is een alfabet  $\Sigma = \{0, 1, \dots, s-1\}$  en een patroon  $P$  ter lengte  $m$ . Geef een methode in  $\mathcal{O}(m + s)$  die voor deze input de *last* functie geeft.
9. Kun je het Boyer-Moore pattern matching algoritme aanpassen zó dat het *alle* (nul of meer) voorkomens van een patroon  $P$  in een tekst  $T$  vindt?
10. (Dit is ongeveer opgave R-9.4.)
- (a) Geef de *failure* functie van het Knuth-Morris-Pratt pattern matching algoritme voor het patroon  $P$  uit het eerste pattern matching probleem (de failure functie geeft aan hoeveel karakters hergebruikt kunnen worden in de vergelijking bij mismatch op  $P[i + 1]$ ).
- (b) Pas het Knuth-Morris-Pratt pattern matching algoritme toe op het eerste pattern matching probleem.
11. Herhaal (eventueel) de vorige opgave voor het tweede pattern matching probleem.
12. (Dit is ongeveer opgave C-9.2.)  
Leg uit waarom het algoritme om de failure functie te berekenen (zie boek) in  $\mathcal{O}(m)$  is, met  $m$  de lengte van het patroon.

13. \* Analyseer de tijdscomplexiteit van het Knuth–Morris–Pratt algoritme, gebruikmakend van aggregate analyse. Zie ook het boek (p427).
14. Als een codering van strings prefix-free is, dan is hij niet ambigu. Toon met een voorbeeld aan dat de omgekeerde implicatie niet geldt, dwz, geef een voorbeeld van een codering die niet prefix-free is maar wel non-ambigu.
15. Geef de frequentie-tabel en de Huffman–coderingsboom voor de volgende string (de spatie doet niet mee):

how much wood would a woodchuck chuck

16. Let uit waarom de tijdscomplexiteit van Huffman's coderingsalgoritme in  $\mathcal{O}(n + d \log(d))$  is. (Zie ook het boek.)
17. \* Pas Huffman's coderingsalgoritme aan zó dat het de symbolen 0, 1, 2 gebruikt om ternaire codewoorden te genereren.
18. (Dit is ongeveer opgave R-9.8.)

Een standard trie is een boom om strings in op te slaan, waarbij de wortel leeg is, elke knoop één karakter bevat, en er voor elke string precies één externe knoop is. Voorbeeld:



Geef een standard trie voor de strings  $\{abab, baba, ccccc, bbaaaa, caa, bbaacc, cbcc, cbca\}$ .