
Uppgift 2: Suffixtabell

Bekanta dig noggrant med följande text och svara med hjälp av den de fem frågorna i slutet av uppgiften.

Att finna en given teckensträng (så kallad mönster) från en annan längre teckensträng (texten) är en grundläggande tillämpning i datavetenskap. Man kan lösa sökningen genom att gå igenom hela texten från början till slut och samtidigt kontrollera om mönstret man söker finns på den platsen. Det här tillvägagångssättet är emellertid långsamt om texten är mycket stor. Man kan göra sökningen snabbare genom att på förhand göra en indexkonstruktion utifrån texten. Med hjälp av indexkonstruktionen kan man undvika att behöva gå igenom hela texten. En enkel och mycket använd indexstuktur kallar suffixtabell. Metoden har allmänt används till exempel för effektiv sökning i stora DNA-datamassor, så som människans genom med ca. 3 miljarder tecken.

En teckensträng består av tecken efter varandra. Man kan hänvisa till en teckensträng med en teckensträngsvariabel. Hänvisning till vissa tecken i teckensträngen kan göras genom att ge tecknets index i hakparentes direkt efter teckensträngsvariabeln. Till exempel betyder $x[1]$ det första tecknet i teckensträngen x. Beteckningen $x[i..j]$ betyder den delteckensträng av teckensträngen x som börjar vid index i och slutar vid index j. En delteckensträng som fortsätter till slutet av teckensträngen kallas suffix. Suffixets startindex kallas suffixets index. Vi använder $<, \leq, =, \geq, >$ i samband med teckensträngarna x och y för att jämföra deras alfabetiska ordning. Till exempel betyder $x \leq y$ att x är mindre eller lika stor som y enligt alfabetisk ordning och $x = y$ betyder att x och y är samma teckensträng.

Exempel 1. Om teckensträngen x är ”dator” gäller att:

- $x[1] =$ “`d`”, $x[2] =$ “`a`” och $x[4] =$ “`o`”.
- $x[1..3] =$ “`dat`”, $x[3..3] =$ “`t`” och $x[2..5] =$ “`ator`”.
- Suffixen till teckensträngen x givna i ordning enligt suffixens index, dvs enligt startindexet 1, 2, 3, 4 och 5, är $x[1..5] =$ “`dator`”, $x[2..5] =$ “`ator`”, $x[3..5] =$ “`tor`”, $x[4..5] =$ “`or`” och $x[5..5] =$ “`r`”.

Vi hänvisar till en text med teckensträngsvariabeln t och till mönstret vi söker i texten med teckensträngsvariabeln p. Dessutom antar vi att längden på texten t är n.

Suffixtabellen S för texten t är en heltalstabell med n element och som innehåller textens suffixens index i alfabetisk ordning för suffixen. Tabellens S index i värde, som vi betecknar $S[i]$, anger från vilken plats i texten den i ordningen i minsta suffixet börjar: $S[1]$ anger index för det minsta suffixet

(enligt alfabetisk ordning) i texten, $S[2]$ index för följande suffix i alfabetisk ordning osv. Man kan beteckna saken också så att för index $i = 2, \ldots, n$ gäller villkoret $t[S[i-1]..n] \leq t[S[i]..n]$.

Exempel 2. Suffixen för texten $t =$ "`abababba`" är "`abababba`", "`bababba`", "`ababba`", "`babba`", "`abba`", "`bba`", "`ba`" och "`a`". Nedan är till vänster suffixen för texten t i alfabetisk ordning och till höger suffixtabellen S för texten t. Notera att suffixtabellens värden är desamma som suffixens index till vänster. Till exempel beskriver värdet $S[5] = 7$ att det i alfabetisk ordning femte största suffixet börjar från indexet 7, dvs. är $t[7..8] =$ "`ba`".

Suffix (alfabetisk ordning)	**Suffixets index**
`a`	8
`abababba`	1
`ababba`	3
`abba`	5
`ba`	7
`bababba`	2
`babba`	4
`bba`	6

Index i	$S[i]$
1	8
2	1
3	3
4	5
5	7
6	2
7	4
8	6

En grundläggande egenskap för suffix är att om mönstret p finns någonstans i texten, så finns p som början på suffixet som börjar på det stället i texten. Vi säger då att mönstret p stämmer överens med suffixet ifråga. Vi kan söka mönstret p från texten genom att söka ett sådant suffix som stämmer överens med p. Med hjälp av suffixtabellen kan denna sökning utföras effektivt genom användning av så kallad binärsökning.

Binärsökning upprätthåller information om det suffixtabellens intervall som i enlighet med den information vi har för tillfället skulle kunna innehålla ett suffix som stämmer överens med mönstret p. Vi använder för sökintervallets startindex beteckningen *start* och för slutindex beteckningen *slut*. Dessutom bestämmer vi det mittersta indexet *mitt* med formeln $mitt = (start + slut)/2$, som vi avrundar uppåt om summan $(start + slut)$ är udda. I början kan alla suffix vara möjliga så $start = 1$ och $slut = n$.

Binärsökningen jämför mönstret och suffixet i sökintervallets mitt $t[S[mitt]..n]$ med varandra. Om mönstret stämmer överens, kan sökningen avslutas[1]. Annars gäller antingen $p < t[S[mitt]..n]$ eller $p > t[S[mitt]..n]$.

Om $p < t[S[mitt]..n]$, dvs. mönstret är i alfabetisk ordning mindre än suffixet i index $S[mitt]$, så kan inga suffix i intervallet $mitt, \ldots, slut$ stämma överens med mönstret. Detta är en direkt följd av att suffixtabellen innehåller

[1] I denna uppgift koncentrerar vi oss på att hitta ett mönster; mönstret kan dock förekomma flera gånger i texten.

suffixen i alfabetisk ordning. Då uppdaterar binärsökningen sökintervallets övre gräns till $slut = mitt - 1$ och jämför mönstret i nästa omgång med suffixen i mitten av det nya sökintervallet.

Om $p > t[S[mitt]..n]$ kan på motsvarande sätt konstateras att inga suffix i intervallet $start, \ldots, mitt$ kan stämma överens med mönstret. Då kan vi uppdatera den undre gränsen till $start = mitt + 1$.

Om sökintervallet blir tomt under sökningen, dvs. kriteriet $start > slut$ uppfylls, avslutas sökningen utan resultat: texten innehåller inga fall av mönstret p. Nedan är sökningen av mönstret p i texten t med hjälp av suffixtabellen S beskriven i lite mer exakt steg för steg:

1. Ställ sökintervallets startindex $start = 1$ och slutindex $slut = n$.
2. Om $start > slut$ dvs. sökintervallet är tomt, sluta sökningen: mönstret p finns inte i texten.
3. Räkna mittindexet $mitt$ mellan start- och slutindex: $mitt = (start + slut)/2$, avrundat uppåt vid behov.
4. Jämför mönstret p med intervallets mittersta suffix $t[S[mitt]..n]$.
   - Om p stämmer överens med början av $t[S[mitt]..n]$, så avsluta sökningen med informationen att mönstret p hittades med start i textens index $S[mitt]$.
   - Om p inte stämmer överens med intervallets mittersta suffix $t[S[mitt]..n]$, så:
     - Om $p < t[S[mitt]..n]$, uppdatera $slut = mitt - 1$ och fortsätt sökningen genom att gå tillbaka till punkt .
     - Om $p > t[S[mitt]..n]$, uppdatera $start = mitt + 1$ och fortsätt sökningen genom att gå tillbaka till punkt 4.

Exempel 3. Mönstret $p =$ "`ababbaba`" sökning i texten $t =$ "`abbaababbababbab`".

Suffix	i	$S[i]$
`aababbababbab`	1	4
`ab`	2	15
`ababbab`	3	10
`ababbababbab`	4	5
`abbaababbababbab`	5	1
`abbab`	6	12
`abbababbab`	7	7
`b`	8	16
`baababbababbab`	9	3
`bab`	10	14
`bababbab`	11	9
`babbab`	12	11
`babbababbab`	13	6
`bbaababbababbab`	14	2
`bbab`	15	13
`bbababbab`	16	8

1. Inled med $start = 1$, $slut = 16$.
2. $mitt = (1 + 16)/2 = 9$,
   $p < t[S[9]..16]$, $slut = mitt - 1 = 8$.
3. $mitt = (1 + 8)/2 = 5$,
   $p < t[S[5]..16]$, $slut = mitt - 1 = 4$.
4. $mitt = (1 + 4)/2 = 3$,
   $p > t[S[3]..16]$, $start = mitt + 1 = 4$.
5. $mitt = (4 + 4)/2 = 4$,
   p stämmer överens med $t[S[4]..16]$ och sökningen avslutas.

I exemplets binärsökning måste vi kontrollera 4 textavsnitt. Binärsökningens fördel är att antalet steg i sökningen växer väldigt långsamt då textens storlek växer. Till exempel har människans genom ca. 3 miljarder tecken, och vid en binärsökning för en sådan textmängd skulle vi behöva kontrollera högst 32 textavsnitt.

Frågorna

Fråga 1. Ange stegen binärsökningen gör då den söker mönstret $p =$ "`babaa`" i texten i exemplet 3 $t =$ "`abbaababbababbab`". Ge ditt svar i samma form som i exemplet 3, dvs. ge för varje steg sökintervallets värden *start*, *slut* och *mitt*. (maximum antal poäng 3)

Fråga 2. Ge suffixtabellen för teckensträngen $t =$ "`yhteisvalinta`". (maximum antal poäng 4)

Fråga 3.

a) Ge suffixtabellen för teckensträngen $t =$ "`aacatcgatagctagaacat`". (maximum antal poäng 4)

b) Ange stegen binärsökningen gör då den söker mönstret $p =$ "`cga`" i texten i punkt a). Ge svaret i samma form som i exemplet 3, dvs. ge för varje steg sökintervallets värden *start*, *slut* och *mitt*.

(maximum antal poäng 3)

Fråga 4. Ge någon sådan teckensträng som består av tecken i svenska alfabetet som har en suffixtabell som den här nedan. I denna uppgift duger alltså tecken som hör till alfabetet "`a`", "`b`", "`c`", ..., "`ö`". (maximum antal poäng 4)

i	$S[i]$
1	8
2	7
3	6
4	5
5	4
6	3
7	2
8	1

Fråga 5. Ge en sådan teckensträng som innehåller underlineenbart tecknen "`a`" och "`b`" som har en suffixtabell som den här nedan.

(maximum antal poäng 7)

i	$S[i]$
1	16
2	13
3	3
4	14
5	11
6	9
7	4
8	6
9	15
10	12
11	2
12	10
13	8
14	5
15	1
16	7