KARLSTADS UNIVERSITY
DVGA 17 TEORETISK DATALOGI

INLÄMNINGSUPPGIFT

Handledare
Kerstin Andersson

Elev
Ahmad Shammout
shmoot001@gmail.com

3.2 Turingmaskiner

En Turingmaskin är en abstrakt modell av en dator [8]. Turingmaskinen består av en lång sträng med symboler, som kallades för bandet, och en läs- och skrivhuvud som kan flytta sig längs bandet. Bandet i en Turingmaskin består av en lång sträng med symboler, som kan vara bokstäver, siffror eller andra tecken. Bandet kan tänkas som en lång lista med element, där varje element motsvarar en symbol [8]. Huvudet på Turingmaskinen kan flytta sig längs bandet och läsa eller skriva symboler på bandet. Huvudet kan flytta sig en position åt gången åt höger eller vänster längs bandet, beroende på vilken instruktion som finns i algoritmen för att lösa problemet. När huvudet läser en symbol på bandet, tar den in symbolen och använder den för att ändra sitt tillstånd eller för att utföra en viss handling, såsom att skriva en ny symbol på bandet [8]. På så sätt kan Turingmaskinen utföra olika uppgifter genom att läsa och skriva symboler på bandet. Turingmaskinen kan användas för att lösa olika problem genom att följa en serie instruktioner, som kallas för en "algoritm". Genom att läsa symboler på bandet, ändra sin tillstånd och skriva nya symboler på bandet kan Turingmaskinen simulera hur en dator utför olika uppgifter. Turingmaskinen är en viktig del av datavetenskapen och har hjälpt till att förklara hur datorer fungerar och vad de kan göra. Den har också använts som ett verktyg för att studera olika problem inom matematik och teoretisk datavetenskap [8].

Problem :

Betrakta funktionen :

$$f : N \rightarrow \{0, 1\} +,$$

eller egentligen

$$f : \{a\} * \rightarrow \{0, 1\} +$$

eftersom vi här representerar ett naturligt tal som ett antal a:n, där

$$f(n) = Sn, n \geq 0.$$

S0, S1, S2, . . . utgör en talföljd bestående av binära tal, som definieras enligt

$$S_n = \begin{cases} 0, & n = 0 \\ S_{n-1}\overline{S}_{n-1}, & n > 0 \end{cases} .$$

Inputsträngar består av 0 eller flera a:n och outputsträngen bestråv av en eller flera 0:or och 1:or.

Vi ska gör en Turingmaskin som beräknar funktionen f, som löser det här problemet, några exempel på inputsträgar och outputsträngar som ska Turningmaskinen beskriva :

$$\epsilon \rightarrow 0$$
$$a \rightarrow 01$$
$$aa \rightarrow 0110$$
$$aaa \rightarrow 01101001$$

Beskrivning av en Turingmaskin och beräkning till funktionen f :

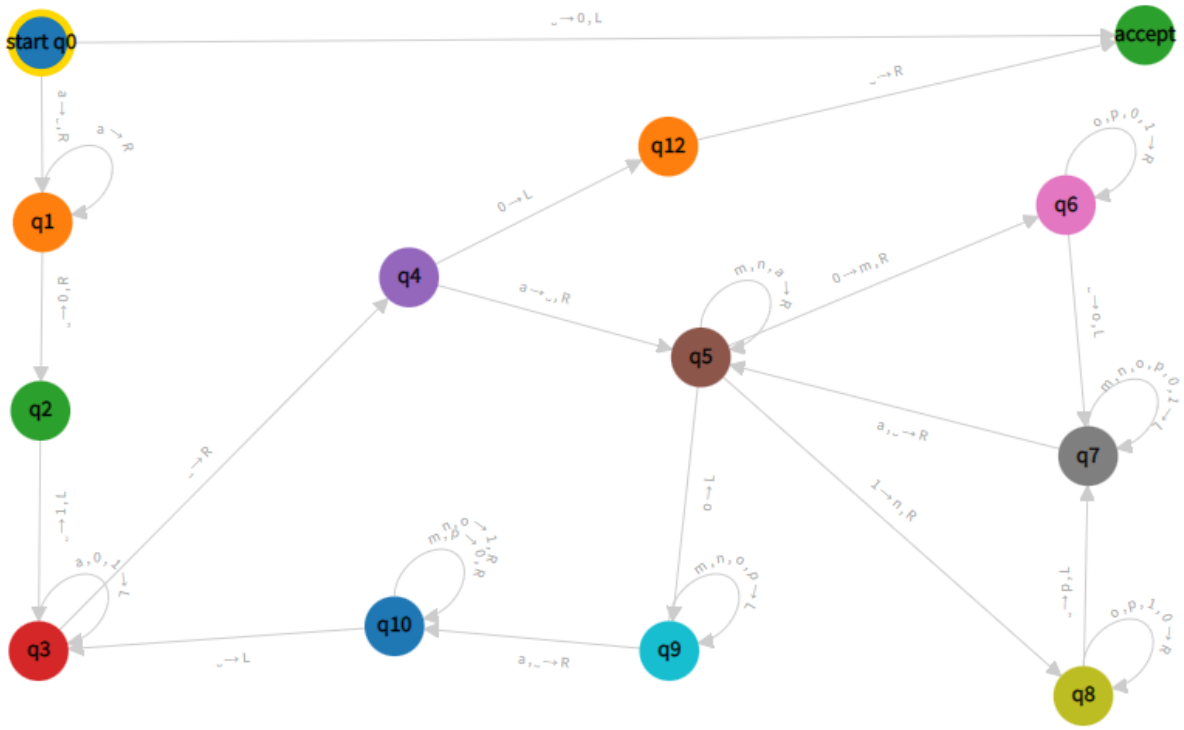
En Turing maskin är en 7-tuple som består av $f = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$

- $Q = \{startq_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11}, q_{12}, accept\}$
- $\Sigma = \{a\}$
- $\Gamma = \{0,1,m,n,o,p\}$
- $\delta = Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ Övergångsfunktionen
- $q_0 = \{startq_0\}$
- $q_{accept} = \{accept\}$
- $q_{reject} \in Q$ reject tillstånd ($\neq q_{accept}$)

Maskinen accepterar en tom sträng där skriv en 0 ut, språket som maskinen tar emot innehåller bara a:or, där när man skriver ett a som inputsträng så skriver maskinen ut 01, där ettan är inversen till 0, som funktionen f ovan beskriver. I början så pekar huvudet på elementet som är längst till vänster, det som händer när maskinen får ingen sträng så skrivs en nolla ut och sen blir det ett accept. När man skriver ett a så ersätter maskinen a till blankt symbol, sedan flyttas huvudet till sista strängen som ligger på bandet, det vill säga tills den hittar ett blankt symbol. När den hittar blankt symbol då skriver maskinen en nolla och flyttas huvudet ett steg åt höger, sedan skriver över en etta, i slutet så får man outputen som 01 när man skriver ett a på bandet. När man skriver fler a på bandet, till exempel aa då kommer maskinen efter att det har skrivit 01 för första a, så kommer huvudet sen att flyttas längst till höger. Sen när det hittar ett till a så byter den till ett blankt symbol och sedan om den hittar en 0 så skriver den över ett ‘m’ och sen går till sllutet av strängen och sedans kriver den i slutet ett ‘o’. Sedan körs huvudet mot väster till den hittar en etta, då ersätts det med ett ‘n’ och sedan går tillbaka till slutet av strängen åt högra sidan och skriver över den blanka symbolen ett ‘p’, Sist går den igenom hela sängen igen och byter ut ‘m’ och ‘p’ till nollor och ‘n’ och ‘o’ till ettor. Till slut får man rätt svar.

Simulering av Turingmaskin finns att kolla på och testa den, det kan man göra genom att trycka på länken nedan.

https://morphett.info/turing/turing.html?c706a78bb2b6b526858657d74f8bc40a



Figur 12 : Turingmaskin konstruktion

```
; Machine starts in state 0.

; State q0
0 _ 0 * q1
0 a _ r q2

; State q1
q1 0 0 * halt-accept
q1 1 1 * halt-accept

; State q2
q2 a a r q2
q2 _ 0 r q3

; State q3
q3 _ 1 l q4

; State q4
q4 0 0 l q4
q4 1 1 l q4
q4 a a l q4
q4 _ _ r q5

; State q5
q5 a _ r q6
q5 0 0 l q12
```

```
; State q6
q6 a a r q6
q6 w w r q6
q6 x x r q6
q6 0 w r q7
q6 1 x r q9
q6 y y l q10

; State q7
q7 w w r q7
q7 z z r q7
q7 0 0 r q7
q7 1 1 r q7
q7 x x r q7
q7 y y r q7
q7 _ y l q8

; State q8
q8 w w l q8
q8 x x l q8
q8 0 0 l q8
q8 1 1 l q8
q8 y y l q8
q8 z z l q8
q8 a a r q6
q8 _ _ r q6

; State q9
q9 z z r q9
q9 y y r q9
q9 0 0 r q9
q9 1 1 r q9
```

```
q9 _ z l q8

; State q10
q10 w w l q10
q10 x x l q10
q10 a a r q11
q10 _ _ r q11

; State q11
q11 w 0 r q11
q11 z 0 r q11
q11 y 1 r q11
q11 x 1 r q11
q11 _ _ l q4

; State q12
q12 _ _ r q1


accept * : r accept2
accept2 * ) * halt-accept

reject _ : r reject2
reject * _ l reject
reject2 * ( * halt-reject
```

Figur 13 : Övergångs Tabell och koden som är skriven i simulator

Tidskomplexitet är en mätning av hur lång tid det tar för en dator eller en algoritm att utföra en uppgift, och det är en viktig faktor att beakta när man utvecklar datorprogram och löser problem med hjälp av datorer [9]. Tidskomplexiteten för en algoritm kan mätas genom att räkna antalet steg eller operationer som algoritmen utför för att lösa problemet. Ju fler steg algoritmen behöver utföra, desto längre tid tar det att lösa problemet, och desto högre blir tidskomplexiteten. Tidskomplexiteten för en algoritm kan också beräknas genom att titta på hur mycket datainput algoritmen behöver för att lösa problemet. Ju mer data algoritmen behöver bearbeta, desto längre tid tar det att lösa problemet, och desto högre blir tidskomplexiteten [9].

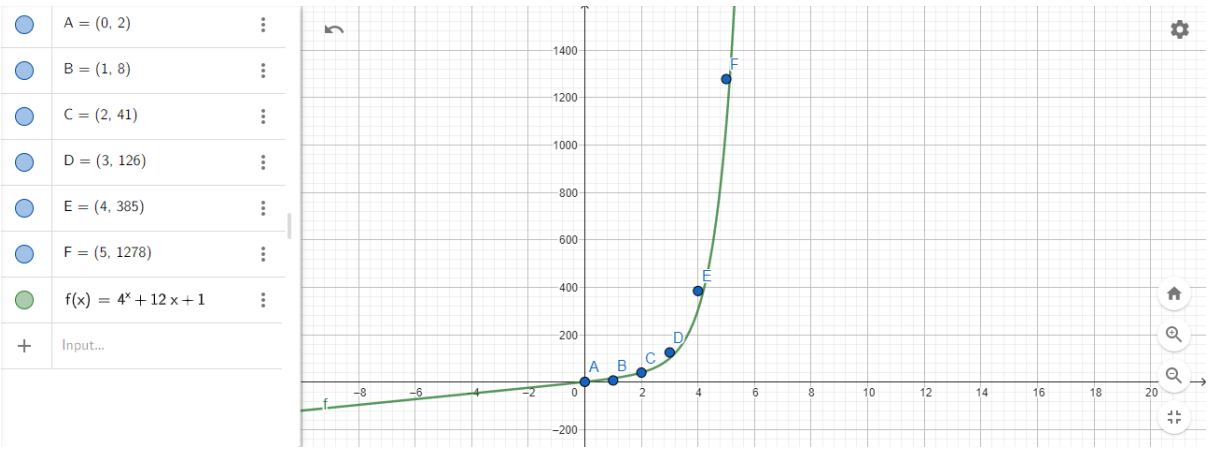
Tidskomplexitet för funktionen f :

Om inmatningssträngen är av längd 1 måste Turingmaskinen korsa den två gånger, dvs säga $a \rightarrow 01$. När inmatningssträngen är av längd 2 måste Turingmaskinen korsa den fyra gånger som till exempel : $aa \rightarrow 0110$. Om inmatningssträngen har längd 3 måste Turingmaskinen korsa den 8 gånger som till exempel $aaa \rightarrow 01101001$. Av detta kan man anta att Turingmaskinens tidskomplexitet är $O(2^n)$, där n är längden på inmatningssträngen.

Tabell 1 : Tabellen nedan visar hur många steg det tar när längden på input strängen ökar.

Längden på inputsträngen	Antal steg
0	2
1	8
2	41
3	126
4	385
5	1278

Värden i tabell 1 ovan plottats på en graf till att veta hur snabbt det ökar, i slutet kom man fram till att $f(x) = 4^x + 12x + 1$, där ser vi grafen nedan att antal stegar ökar väldigt mycket när man ökar på antal input strängar.



Figur 14 : Funktion antal steg av antal input strängar.

4. Sammanfattning

Under den här inlämningsuppgift så har man gått igenom flera området som handlar om i kursen där började vi med att beskriva reguljära uttrycken och hur de används i praktiken genom att testa flera Unix-verktygen. Sedan hade vi gått igenom vad lexikal analys och hur den används, där använde vi också reguljära uttrycken för att underlätta arbetet av att definiera strängar genom att beskriva mönstret mha reguljära uttryck. I uppgift 2 började med att komma in i området reu kursionssatsen, där skulle ett ämne väljas som handlar om reukursionssatsen, området som valdes var datorvirus. Vi förklarade vad datorvirus är och hur den används och vad den gör. Sedan kom Turingmaskin där beskrivs kortfattat vad en Turingmaskin och tidskomplexitet är. Därefter behövde man att konstruera en Turingmaskin och beräkna tidskomplexiteten till funktion f. Under den här arbetet har man lärt sig väldigt många saker som är relaterade till kursen och handlar om teoretisk datalogi.