

KARLSTADS UNIVERSITET
INSTITUTIONEN FÖR MATEMATIK OCH DATAVETENSKAP

TEORETISK DATALOGI
DVGA17

Komplexitetsteori

Skriven av:

Alexander FLOREAN
florean.alexander@gmail.com
Emanuel SVENSSON
emansven100@student.kau.se

Handledare:

Kerstin ANDERSSON

8 januari 2020



Innehåll

1	Inledning	2
1.1	Antaganden	2
2	Del 1: Turingmaskin	2
3	Del 2: Tornen i Hanoi	4
4	Sammanfattning	5

1 Inledning

1.1 Antaganden

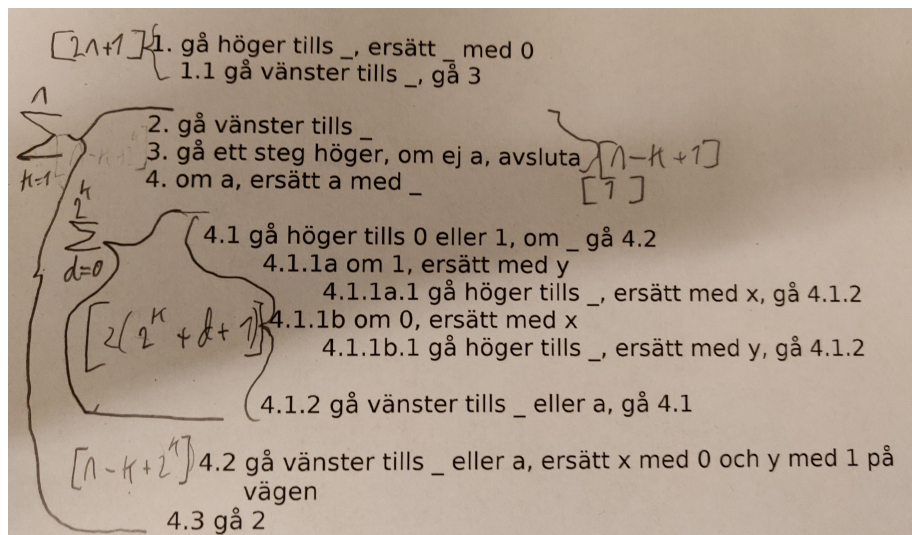
2 Del 1: Turingmaskin

Inledning Uppgiften går ut på att skapa och analysera en turingmaskin som efter antal $a:n$ i input lägger till inversen av en sträng med 0:or och 1:or till höger, startandes med 0 vilket ger 0 \rightarrow 01 \rightarrow 0110 osv.

Beskrivning av turingmaskinen Följande algoritm utvecklades för att lösa problemet.

1. gå höger tills $_$, ersätt $_$ med 0
 - 1.1 gå vänster tills $_$, gå 3
2. gå vänster tills $_$
3. gå ett steg höger, om ej a, avsluta
4. om a, ersätt a med $_$
 - 4.1 gå höger tills 0 eller 1, om $_$ gå 4.2
 - 4.1.1a om 1, ersätt med y
 - 4.1.1a.1 gå höger tills $_$, ersätt med x, gå 4.1.2
 - 4.1.1b om 0, ersätt med x
 - 4.1.1b.1 gå höger tills $_$, ersätt med y, gå 4.1.2
 - 4.1.2 gå vänster tills $_$ eller a, gå 4.1
- 4.2 gå vänster tills $_$ eller a, ersätt x med 0 och y med 1 på vägen
- 4.3 gå 2

Tidskomplexitet av algoritm



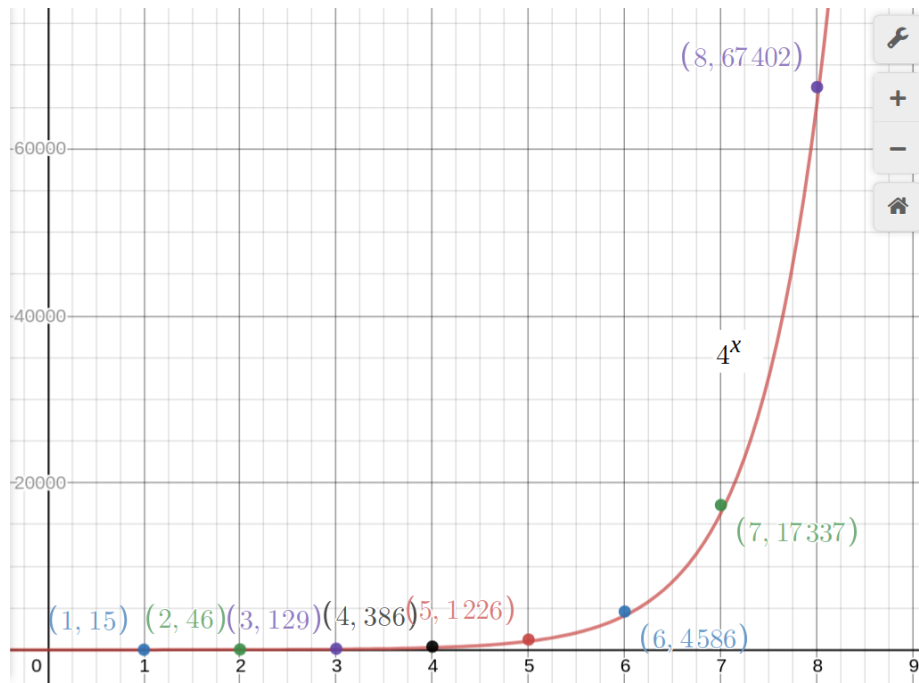
Figur 1: Analys av komplexitet per del

$$\begin{aligned}
 & [2n+1] + \sum_{k=1}^n ([n-k+1] + [1]) + \sum_{d=0}^{2^k} ([2(2^k+d+1)] + [n-k+2^k]) \\
 & = O(4^n)
 \end{aligned}$$

Figur 2: Uträkning av komplexiteten i $O()$

Ur analysen framkommer det att algoritmen har tidskomplexiteten $O(4^n)$

Tidskomplexiteten ritades på en linje och jämfördes med flertal datapunkter från när maskinen kördes i simulatoren.



Figur 3: Graf som jämför datapunkter och den uträknade tidskomplexiteten. Y-axeln står för antalet steg och x-axeln för antalet $a:n$ i indatan.

3 Del 2: Tornen i Hanoi

Inledning Vi har tre pinnar med givna respektive namn A, B, C, samt notationen N för antal ringar.

Regler:

1. Flytta en ring åt gången.
2. Lägg aldrig en större ring ovanpå en mindre.

Given rekursiv algoritm:

1. Flytta temporärt högen med $N - 1$ ringar från pinne A till pinne B.
2. Flytta den största ringen (den enda ringen kvar på pinne A) till pinne C.
3. Flytta $N - 1$ ringar från pinne B till pinne C.

Omskrivning görs för att komma fram till en tidskomplexitet av den rekursiva algoritmen till en rekursiv psudokod/funktion. Som kan beskrivas, funktionen $H(\text{antal ringar, som flyttas från pinne } x, \text{ till pinne } y)$

Tidsfunktion

```

H( N, A, C)
H( N - 1, A, B)      // kallar på sig självt
flytt(A,C)           // flyttar ringen ett steg
H( N - 1, B, C)      // kallar på sig självt

```

Med tiden i åtanke kan flytt(A,C) ses som en konstant ökning i tidskomplexiteten.

Därför kan ovan skrivas om till tidsfunktionen

$$\begin{aligned}
 T(N) &= T(N-1) + 1 + T(N-1) \\
 &= 2T(N-1) + 1
 \end{aligned}$$

2005/06/28ver : 1.3subfigpack

Med hjälp av inducering fås.

$$\begin{aligned}
 T(N) &= 2[2T(N-1-1) + 1] + 1 \\
 &= 2^2T(N-2) + 3 \\
 &= 2^2[2T(N-3) + 1] + 3 \\
 &= 2^3T(N-3) + 7
 \end{aligned}$$

2005/06/28ver : 1.3subfigpack

Då k är antal ringar för funktionen $T(N) = 2^kT(N-k) + 2^k - 1$
 När $k \rightarrow N$ så fås tidsfunktionen

$$\begin{aligned}
 T(N) &= 2^N T(N-N) + 2^N - 1 \\
 &= 2^N T(0) + 2^N - 1 \\
 &= 0 + 2^N - 1 = 2^N - 1
 \end{aligned}$$

2005/06/28ver : 1.3subfigpack

Detta ger Tornen i Hanoi tidskomplexiteten $\mathcal{O}(2^n)$ då konstanten kan bortses.

4 Sammanfattning

Referenser

- [1] Michael Sipser, *Introduction to the Theory of Computation Third Edition*, June 2012, Cengage Learning