IX

Complexity Theory

- computability-in-principle vs computability-in-practice

computational / capability of a Turing machine in the absence of any proscribed limits on the length of computation or on the # squares that may be visited

vs

computational feasibility: which problems would be solvable on a Turing machine given the quantities on time & space likely to be available

- $f(n)$, $g(n)$: number-theoretic functions

- Def: $f(n)$ is said to be $O(g(n))$ if
$\forall\ n \geq n_0,\quad f(n) \leq c \cdot g(n)$

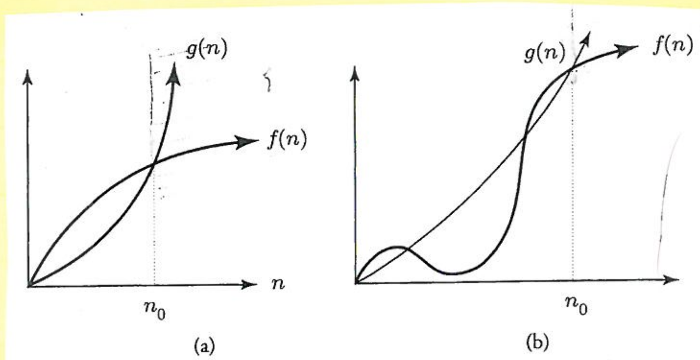

Figure 1.7.1 Function $f(n)$ is $O(g(n))$.

$g(n)$ is an upper-bound for $f(n)$

- O : read as "big-oh"
Worst case time analysis

TX

Complexity of Turing machine computations

DEFINITION 1.8: Let M be a Turing machine and let n be an arbitrary natural number. The unary number-theoretic function $time_M$ is defined by

$time_M(n)$ = the maximum number of "steps" in any terminating computation of M for an input of size n

- We note - the computation must be a halting one
  - a step is any instruction that is executed - i.e. either a move or write.

- time $_M(0)$

Turing machine

string length

An example

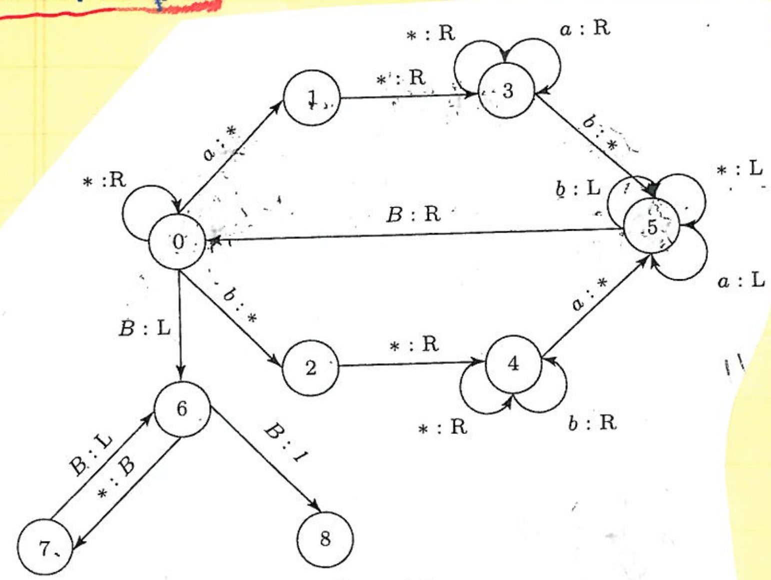

Figure 1.3.1 Same Number of *a*s and *b*s.

Ignore pencil marks - first time

space $_M(0) = 2$

- Suppose input string has length zero, i.e. $w = \epsilon$

$q_0 B \vdash B q_6 B$
$\quad \vdash B q_8 1$

We observe that two instructions have been executed

$\therefore$ time $_M(0) = 2$

Time Complexity - an example

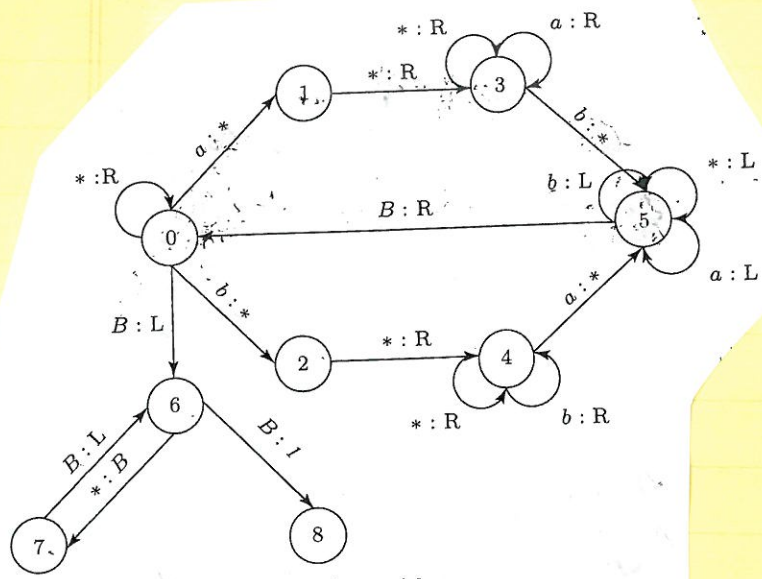

Figure 1.3.1 Same Number of *as* and *bs*.

- Next, we calculate $time_m(1)$
  - There are two strings of length one to consider i.e. $w = a$ and $w = b$

$w = a$

$$q_0 a \vdash q_1 * \vdash * q_3 B$$

$w = b$

$$q_0 b \vdash q_2 * \vdash * q_4 B$$

$space_m(1) = 2$

- In each case, two instructions were executed. Hence

$$time_m(1) = 2$$

- Next, we let string length = 2. There are four cases to consider

$w = aa$
$w = ab$
$w = ba$
$w = bb$

example con't.

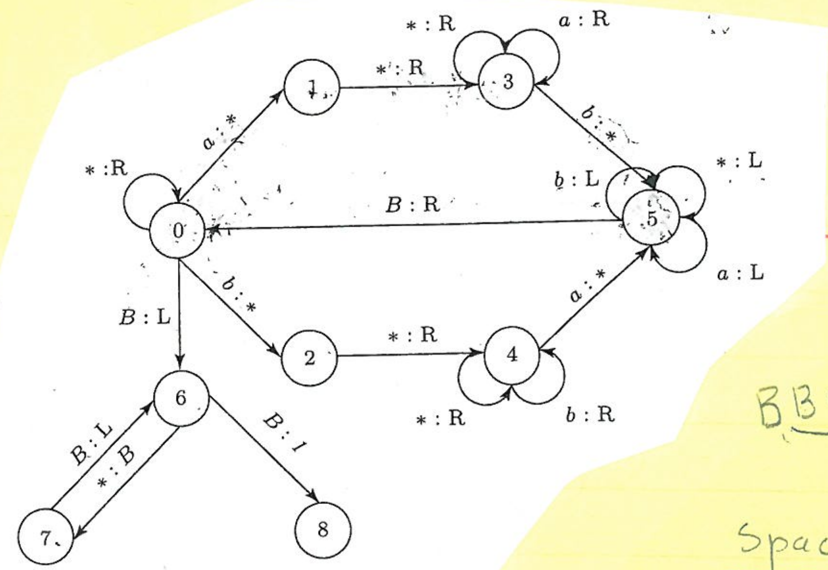

Figure 1.3.1 Same Number of as and bs.

BBabB

$\text{Space}_M(2) = 4$

case 1: w = aa : $q_0 aa \vdash q_1 {*}a \vdash {*}q_3 a \vdash {*}a q_3 B$

case 2: w = ab : $q_0 ab \vdash q_1 {*}b \vdash {*}q_3 b \vdash {*}q_5 {*}$
$\vdash q_5 {*}{*} \vdash q_5 B{*}{*} \vdash q_0 {*}{*}$
$\vdash {*}q_0 {*} \vdash {*}{*}q_0 B \vdash {*}q_6 {*}$
$\vdash {*}q_7 B \vdash q_6 {*} \vdash q_7 B$
$\vdash q_6 B \vdash q_8 1$

case 3: w = ba : $q_0 ba \vdash q_2 {*}a \vdash {*}q_4 a \vdash {*}q_5 {*}$
$\vdash q_5 {*}{*} \vdash q_5 B{*}{*} \vdash q_0 {*}{*}$
$\vdash {*}q_0 {*} \vdash {*}{*}q_0 B \vdash {*}q_6 {*}$
$\vdash {*}q_7 B \vdash q_6 {*} \vdash q_7 B$
$\vdash q_6 B \vdash q_8 1$

case 4: w = bb : $q_0 bb \vdash q_2 {*}b \vdash {*}q_4 b \vdash {*}b q_4 B$

- We notice that accepting computations require the most effort.
- Here, cases 2 and 3 each require 14 steps.

$\text{time}_M(2) = 14$

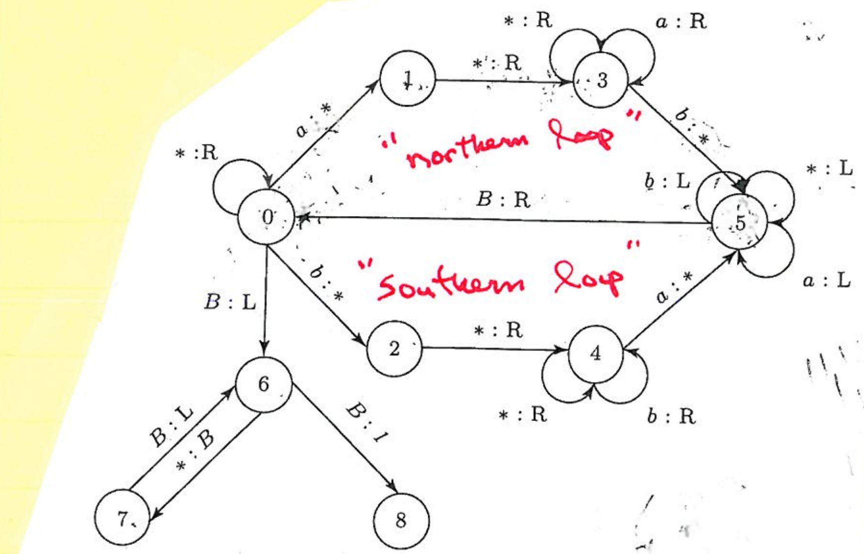

Figure 1.3.1 Same Number of *as* and *bs*.

- If $w \in L$ then $|w|$ is even
- Each loop (either "northern" or "southern") removes one 'a' and one 'b'.

① • Hence when $|w| = n$, $\frac{n}{2}$ loops required.

② • The maximum distance between an "a-b" pair is $\frac{n}{2}$ **why?**

$|\leftarrow \frac{n}{2} \rightarrow|$

B a ····· b ·····

w

③ • The Blank is being used as a left delimiter

④ • The last (and longest) trip is length n

B * * ····· * B

| n char. |

| n steps left |

$space_M(n) = n+2$

M computes in linear space

⑤ • Once w has been "starred out" we travel n+1 steps right (until the right blank)

⑥ • then 1 step left (facing "*" in state 6)

⑦ • 2n steps to erase stars and ⑧ • 1 step to write '1'.

Time complexity Analysis

Observation 1

$$\underset{\text{loops}}{\frac{n}{2}} \left[\begin{array}{l} \text{time to stan out an 'a' and 'b'} \\ \text{and travel} \end{array} \right] + \left[\begin{array}{c} \text{time to} \\ \text{clean} \\ \text{up} \\ \text{tape} \end{array} \right] + \begin{array}{c} \text{time} \\ \text{to} \\ \text{write} \\ \text{`1'} \end{array}$$

We will consider worst case as every trip

$$= \underset{\text{loops}}{\frac{n}{2}} \left[\begin{array}{l} \text{travel } n \text{ steps right} \\ \text{write two } *\text{'s} \\ \text{travel } n+1 \text{ steps right} \end{array} \right] + \left[\begin{array}{c} \text{time to} \\ \text{clean} \\ \text{tape} \end{array} \right] + \begin{array}{c} \text{time} \\ \text{to} \\ \text{write} \\ \text{`1'} \end{array}$$

observation 6

$$= \frac{n}{2} \left[n + 2 + n + 2 \right] + \left[1 + 2n \right] + 1$$

Observation 7

Observation 8

$$= \frac{n}{2} \left[2n + 4 \right] + \left[2n + 1 \right] + 1$$

$$= \frac{2n^2}{2} + \frac{4n}{2} + 2n + 2$$

$$= n^2 + 4n + 2$$

$$\text{time}_M(n) = n^2 + 4n + 2 = O(n^2)$$

Space Complexity

DEFINITION 1.9: Let M be a Turing machine and let n be an arbitrary natural number. The number-theoretic function $space_M$ is defined by

$$space_M(n) = \text{the maximum number of tape squares scanned over the course of any terminating computation of } M \text{ for arbitrary input of size } n$$

- Size of input will correspond to input word length
- We count the number of distinct squares visited
- If $|w| = 0$, we must still visit one square

$\therefore \; space_M(n) \geq 1$

return to page 2

IX

Significance of Time Complexity

DEFINITION 1.10: A language L over alphabet Σ is said to be *polynomial-time Turing-acceptable* if there exist both a deterministic Turing machine M and a polynomial $p(n)$ such that, for any $w \in \Sigma^*$, we have $w \in L$ if and only if M accepts w in $O(p(|w|))$ steps. Equivalently, language L over alphabet Σ is polynomial-time Turing-acceptable if there exists a deterministic Turing machine M such that M accepts L and M computes in $O(n^k)$ steps for some constant $k \in N$, where $n = |w|$. (As usual, we are writing $|w|$ for the length of word w.)

DEFINITION 1.11: The class of all languages that are accepted in polynomially bounded time by some (deterministic, single-tape) Turing machine is known as P.

Table 1.7.1 Comparison of the Growth Rates of Several Functions, Where We Are Assuming That Each Computation Step Requires One Microsecond[a]

	$n = 10$	$n = 20$	$n = 30$
$time_M(n) = n$	0.00001 seconds	0.00002 seconds	0.00003 seconds
$time_M(n) = n^2$	0.0001 seconds	0.0004 seconds	0.0009 seconds
$time_M(n) = n^3$	0.001 seconds	0.008 seconds	0.027 seconds
$time_M(n) = n^4$	0.01 seconds	0.16 seconds	0.81 seconds
$time_M(n) = 2^n$	0.001024 seconds	1.048576 seconds	17.8957 minutes
	$n = 40$	$n = 50$	$n = 60$
$time_M(n) = n^2$	0.00004 seconds	0.00005 seconds	0.00006 seconds
$time_M(n) = n^3$	0.0016 seconds	0.0025 seconds	0.0036 seconds
$time_M(n) = n^3$	0.064 seconds	0.125 seconds	0.216 seconds
$time_M(n) = n^4$	2.56 seconds	6.25 seconds	12.96 seconds
$time_M(n) = 2^n$	12.72583 days	35.67843 years	365.34711 centuries

[a] *One microsecond equals 0.000001 second.*

When $time_M(n) = O(2^n)$ for some Turing machine M it is not feasible to use M for even reasonable problem sizes, i.e. $n > 40$