

Def: DTIME, NTIME, DSPACE, measured on **Multi-tape Turing Machines**.

Th: $\mathbf{DTIME}[t(n)] \subseteq \mathbf{RAM-TIME}[t(n)] \subseteq \mathbf{DTIME}[(t(n))^3]$

$$\mathbf{L} \equiv \mathbf{DSPACE}[\log n]$$

$$\mathbf{P} \equiv \mathbf{DTIME}[n^{O(1)}] \equiv \bigcup_{i=1}^{\infty} \mathbf{DTIME}[n^i]$$

$$\mathbf{NP} \equiv \mathbf{NTIME}[n^{O(1)}] \equiv \bigcup_{i=1}^{\infty} \mathbf{NTIME}[n^i]$$

$$\mathbf{PSPACE} \equiv \mathbf{DSPACE}[n^{O(1)}] \equiv \bigcup_{i=1}^{\infty} \mathbf{DSPACE}[n^i]$$

Th: For $t(n) \geq n, s(n) \geq \log n,$

$$\mathbf{DTIME}[t(n)] \subseteq \mathbf{NTIME}[t(n)] \subseteq \mathbf{DSPACE}[t(n)]$$

$$\mathbf{DSPACE}[s(n)] \subseteq \mathbf{DTIME}[2^{O(s(n))}]$$

Cor: $\mathbf{L} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE}$

Definition 6.1 The **busy beaver function** $\sigma(n)$ is the maximum number of one's that an n state TM with alphabet $\Sigma = \{0, 1\}$ can leave on its tape and halt when started on the all 0 tape. \square

Note: In this example, to be consistent with literature, e.g., Hopcroft article, we are taking our tape alphabet $\Sigma = \{0, 1\}$.

Note that $\sigma(n)$ is **well defined**:

There are only finitely many n -state TMs, with $\Sigma = \{0, 1\}$.

M_0 M_1 M_2 M_3 \dots M_{289} M_{290} \dots M_{3042} M_{3043} \dots $M_{t(n)}$

Some finite subset of these eventually halt on input 0.

M_1 M_3 M_{289} M_{3043}

One of these prints the max # of 1's = $\sigma(n)$.

1 17 22,961 407

	q_1	q_2	q_3
0	$q_2, 1, \rightarrow$	$q_3, 0, \rightarrow$	$q_3, 1, \leftarrow$
1	$h, 1, -$	$q_2, 1, \rightarrow$	$q_1, 1, \leftarrow$

$$\sigma(3) \geq 6$$

q_1	0	0	0	0	0	0	0
q_2	0	1	0	0	0	0	0
q_3	0	1	0	0	0	0	0
q_3	0	1	0	1	0	0	0
q_3	0	1	1	1	0	0	0
q_1	0	1	1	1	0	0	0
q_2	1	1	1	1	0	0	0
q_2	1	1	1	1	0	0	0
q_2	1	1	1	1	0	0	0
q_3	1	1	1	1	0	0	0
q_3	1	1	1	1	0	1	0
q_3	1	1	1	1	1	1	0
q_1	1	1	1	1	1	1	0
h	1	1	1	1	1	1	0

Is $\sigma(n) \in O(n^2)$?

$O(n^3)$?

$O(2^n)$?

$O(n!)$?

$O(2^{2^n})$?

$O(\text{hyperexp}(n))$?

$O(\text{hyperexp}(\text{hyperexp}(n)))$?

$$\text{hyperexp}(n) = 2^{\left. 2^{\dots 2} \right\} n}$$

States	Max # of 1's	Lower Bound for $\sigma(n)$
3	$\sigma(3)$	6
4	$\sigma(4)$	12
5	$\sigma(5)$	17
6	$\sigma(6)$	35
7	$\sigma(7)$	22,961
8	$\sigma(8)$	$3^{92} \sim 7.9 \times 10^{43}$

Theorem 6.2 *Let $f : \mathbf{N} \rightarrow \mathbf{N}$ be a total, recursive function.*

$$\lim_{n \rightarrow \infty} \left(\frac{f(n)}{\sigma(n)} \right) = 0$$

That is, $\sigma(n)$ is eventually bigger than any total, computable function!

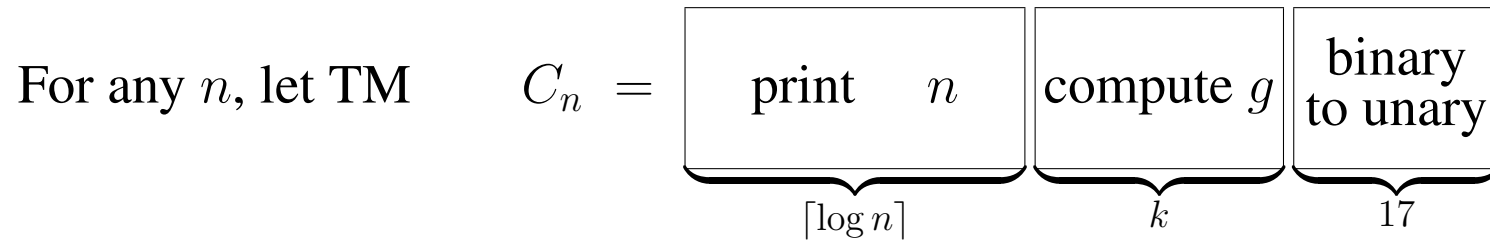
Proof:

Let $g(n) = n \cdot (f(n) + 1)$ g is total, recursive.

Note: $\lim_{n \rightarrow \infty} \left(\frac{f(n)}{g(n)} \right) = 0$

We'll show that, $\sigma(n) \geq g(n)$, for all sufficiently large n .

$g(n)$ is computed by some k -state TM.



C_n has $\lceil \log n \rceil + k + 17$ states.

C_n prints $g(n)$ 1's.

For $n \geq \lceil \log n \rceil + k + 17$ states,

$$\sigma(n) \geq \sigma(\lceil \log n \rceil + k + 17) \geq g(n)$$

□

From Hw2, we have a pairing function:

$$P : \mathbf{N} \times \mathbf{N} \xrightarrow[\text{onto}]{1:1} \mathbf{N}$$

$$P(L(w), R(w)) = w$$

$$L(P(i, j)) = i$$

$$R(P(i, j)) = j$$

We can use the pairing function to think of a natural number as a pair of natural numbers.

Thus, the input to a Turing machine is a single binary string which may be thought of as a natural number, a pair of natural numbers, a triple of natural numbers, etc.

Turing machines can be encoded as **character strings** which can be encoded as **binary strings** which can be encoded as **natural numbers**.

TM_n	1	2	3	4
0	1, 0, \rightarrow	3, \sqcup , \rightarrow	0, 0, $-$	0, 0, $-$
1	1, 1, \rightarrow	4, \sqcup , \rightarrow	0, 1, $-$	0, 1, $-$
\sqcup	2, \sqcup , \leftarrow	0, \sqcup , $-$	1, 0, \leftarrow	1, 1, \leftarrow
\triangleright	1, \triangleright , \rightarrow	0, \triangleright , $-$	0, \triangleright , $-$	0, \triangleright , $-$

ASCII: 1, 0, \rightarrow ; 1, 1, \rightarrow ; 2, \sqcup , \leftarrow ; 1, \triangleright , \rightarrow ; ; \cdots 0, \triangleright , $-$

$\{0, 1\}^*$: w

\mathbf{N} : n

There is a simple, countable listing of all TM's: M_0, M_1, M_2, \cdots

Encoding (transition function to binary string) and
Decoding (binary string to transition function) are trivial to compute.

Theorem 6.3 *There is a Universal Turing Machine U such that,*

$$U(P(n, m)) = M_n(m)$$

Proof: Given $P(n, m)$, compute n and m . n is a binary string encoding the state table of TM M_n . We can simulate M_n on input m by keeping track of its state, its tape, and looking at its state table, n , at each simulated step.



□

Theorem 6.3 is perhaps the most fundamental fact about computation.

$$\text{HALT} = \{P(n, m) \mid \text{TM } M_n(m) \text{ eventually halts}\}$$

Corollary 6.4 (Unsolvability of the Halting Problem) *HALT is r.e. but not recursive.*

Proof:

$$\begin{aligned} \text{HALT} &= \{w \mid U(w) \text{ eventually halts}\} \\ &= \{w \mid U'(w) = 1\} \end{aligned}$$

$$U' = \begin{array}{|c|c|c|} \hline U & \text{erase tape} & \text{print 1} \\ \hline \end{array} \quad \text{Simulate } U; \text{ if it eventually halts, then output 1}$$

Suppose HALT were recursive. Then $\sigma(n)$ would be a total recursive function: Cycle through all n -state TMs, M_i , and if $P(i, 0) \in \text{HALT}$, then count the number of 1's in $M_i(0)$. Return the maximum of these.

$\Rightarrow \Leftarrow$

□

$$W_i = \{n \mid M_i(n) = 1\} = \mathcal{L}(M_i)$$

The set of all r.e. sets = W_0, W_1, W_2, \dots

n	0	1	2	3	4	5	6	7	8	\dots	W_n
0	0	0	0	0	0	0	0	0	0	\dots	W_0
1	1	1	1	1	1	1	1	1	1	\dots	W_1
2	1	0	1	0	1	0	1	0	1	\dots	W_2
3	0	1	0	1	0	1	0	1	0	\dots	W_3
4	1	0	0	0	0	0	0	0	0	\dots	W_4
5	0	1	1	0	1	0	0	0	1	\dots	W_5
6	1	0	0	1	0	0	1	0	0	\dots	W_6
7	1	1	0	0	0	0	0	0	0	\dots	W_7
8	0	1	0	0	0	0	0	0	0	\dots	W_8
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\dots	\vdots
	0	1	1	1	0	0	1	0	0	\dots	K
	1	0	0	0	1	1	0	1	1	\dots	\overline{K}

$$\begin{aligned}
 K &= \{n \mid M_n(n) = 1\} \\
 &= \{n \mid U(P(n, n)) = 1\} \\
 &= \{n \mid n \in W_n\}
 \end{aligned}$$

Theorem 6.5 *K is r.e., but \bar{K} is not r.e.*

Proof: $\bar{K} = \{n \mid n \notin W_n\}$

Suppose \bar{K} were r.e. Then for some c ,

$$\bar{K} = W_c = \{n \mid M_c(n) = 1\}$$

$$c \in K \Leftrightarrow M_c(c) = 1 \Leftrightarrow c \in W_c \Leftrightarrow c \in \bar{K}$$

$\Rightarrow \Leftarrow$

□

Corollary 6.6 *$K \in \text{r.e.} - \text{Recursive}$*

“Anyone familiar with the theory of computability will be aware that practical conclusions from the theory must be drawn with caution. If a problem can theoretically be solved by computation, this does not mean that it is practical to do so. Conversely, if a problem is formally undecidable, this does not mean that the subcases of primary interest are impervious to solution by algorithmic methods.”

Albert Meyer and Dennis Ritchie
“The Complexity of Loop Programs,”
Proc. 22nd National ACM Conference,
(1967), Washington, DC, 465-470.