Complexity Theory

Up to now we investigated whether a problem is in principle solvable algorithmically, that is, we asked the question whether a particular language is,

Decidable: machine halts on all inputs (total computable functions)

Turing-Recognizable: machine loops forever on some inputs (partially computable functions)

However, we did not investigate the cost of the computation itself - the amount of resources the computation absorbs (time, space, *etc.*)

In the following we discuss *time complexity* and *space complexity*.

Furthermore, we assume that we are dealing with total computable functions, that is, the respective language is decidable.

Time Complexity

Definition: Let M be a deterministic TM that halts on all inputs. The ***running time*** or ***time complexity*** of M is the function $f : \mathbb{N} \rightarrow \mathbb{N}$, where $f(n)$ is the *maximum number of steps* that M uses on any input of length n.
If $f(n)$ is the running time of M, then we say that M runs in time $f(n)$ and that M is an $f(n)$ time TM. Customarily we use n to represent the length of the input.

Our time complexity analysis is called ***worst case analysis*** because we only consider the ***maximum number of steps*** a machine uses on input n.

Big-O Notation

Asymptotic analysis or *big-O notation*.

Definition: Let f and g be functions $f, g : \mathbb{N} \to \mathbb{R}^+$. Say that

$$f(n) = O(g(n))$$

if positive integers c and n_0 exist such that for every integer $n \geq n_0$,

$$f(n) \leq c\, g(n).$$

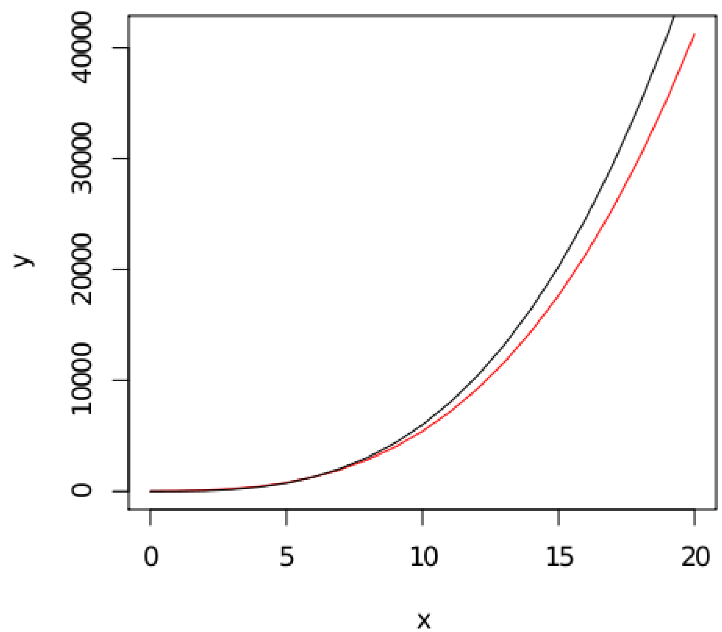
When $f(n) = O(g(n))$ we say that $g(n)$ is an ***(asymptotic) upper bound*** for $f(n)$.

Big-O Notation

Example: Let $f(n) = 5n^3 + 2n^2 + 22n + 6$, then only considering the highest order term and disregarding all constants and coefficients we get

$$f(n) = O(n^3).$$

We can show that this satisfies our formal definition of asymptotic analysis by letting $c = 6$ and $n_0 = 10$. Then for any $n > 10$ we have $f(n) \leq 6n^3$.



Big-O Notation

Notes:

- Let $f(n) = 3n \log_2 n + 5n + 3$, then $f(n) = O(n \log n)$. Notice that we dropped the base subscript because $\log_b n = \frac{1}{\log_2 b} \log_2 n$ for any base b, that is different logarithms are related to each other by a constant factor.
- $f(n) = O(n^2) + O(n) \Rightarrow f(n) = O(n^2)$.
- $f(n) = 2^{O(n)} \Rightarrow f(n) \leq 2^{cn}$ for some c and some value n_0 such that $n > n_0$.

Bounds of the form $O(n^k)$ where $k > 0$ are called ***polynomial bounds***. Bounds of the form $2^{O(n^k)}$ where $k > 0$ are called ***exponential bounds***.

Analyzing Algorithms

Example: Given the decidable language $A = \{o^k 1^k | k \geq 0\}$ and a TM M_1 that decides it, where

M_1 = "On input string w:

1. Scan across the tape and *reject*if a 0 is found to the right of a 1.
2. Repeat the following if both 0s and 1s remain on the tape.
3. Scan across the tape, crossing off a single 0 and a single 1.
4. If neither 0s nor 1s remain on the tape *accept*, otherwise *reject*."

To analyze the time complexity of this machine we analyze each stage separately:

stage 1: The machine scans across the tape to verify that the input is of the form 0^*1^*. Performing this scan uses n steps where n is the length of the input. Repositioning the head to the beginning of the tape takes another n steps. Therefore, this stage takes $2n$ or $O(n)$ steps.

stage 2,3: Here the machine scans the input repeatedly. Each scan takes $O(n)$ steps. Since each scan crosses off two symbols at a time, at most $n/2$ scans can occur. That is $(n/2)O(n) = O(n^2)$.

stage 4: The machine makes a single scan over the input to decide whether to accept or to reject - $O(n)$ steps.

Total time complexity of M_1 on input n is $2O(n) + O(n^2) = O(n^2)$. Can we find a faster algorithm or computational model?

Analyzing Algorithms

Consider the 2-tape machine M_2,

M_2 = "On input string w:

1. Scan across tape 1 and *reject*if a 0 is found to the right of a 1.
2. Scan across the 0s on tape 1 until the first 1. At the same time copy the 0s to tape 2.
3. Scan across the 1s on tape 1 until the end of the input. For each 1 read on tape 1 cross off a 0 on tape 2. If all 0s are crossed off before all 1s are read, *reject*.
4. If all 0s have now been crossed off, *accept*. If any 0s remain, *reject*."

It is easy to see that each stage of this machine takes $O(n)$ steps - time complexity is $O(n)$ or linear!

Discussion: It is interesting to note that even though computability did not depend on the precise model of computation chosen – time complexity does! We saw that both our machines, M_1 and M_2, decide the language A, but M_1 did it in time complexity $O(n^2)$ and M_2 decided the language in time complexity $O(n)$.