

Computational Complexity

A decidable problem is

- computationally solvable in principle,
- but not necessarily in practice.

Problem is resource consumption:

- time
- space

Let M be a deterministic TM that halts on all inputs. The *running time* of M is a function $f : \mathbb{N} \rightarrow \mathbb{N}$ where $f(n)$ is the maximum running time of M on input of length n .

- worst case: longest running time on input of given length
- average case: average running time on input of given length

Asymptotic Notation

The exact running time of most algorithms is quite complex so it is better to “estimate” it.

For functions f , g , we say that:

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

if there exist positive integers c_1 and c_2 such that:

$$f(n) \leq c_1 \cdot g(n), \text{ for } n \geq c_2$$

For example, consider $f(n) = 5n^3 + 2n^2 + 22n + 6$

To show that $f(n) = O(n^3)$, let $c_1 = 6$ and $c_2 = 6$.

Then $5n^3 + 2n^2 + 22n + 6 \leq 6n^3$ for every $n \geq 6$

Other examples:

- $617x^3 + 277x^2 + 720x + 7 = O(x^3)$
- $2 = O(1)$ (i.e. runs in *constant time*)
- $\sin(x) = O(1)$

Polynomial and Exponential

Definition: $\text{TIME}(f(n)) = \{L \mid L \text{ is a language decided by an } O(f(n))\text{-time TM}\}$

If there is an upper bound of n^k , where $k > 0$, then we say that it runs in *polynomial* time.

Definition: The class P is the set of languages decidable in polynomial time on deterministic single-tape TMs. This roughly corresponds to realistically solvable (*tractable*) problems:

$$P = \bigcup_k \text{TIME}(n^k)$$

If there is an upper bound of $2^{(n^k)}$, where $k > 0$, then we say that it runs in *exponential* time.

Definition: The class $EXPTIME$ is the set of languages decidable in exponential time on deterministic single-tape TMs. Problems belonging to this class are *intractable*:

$$EXPTIME = \bigcup_k \text{TIME}(2^{n^k})$$

Polynomial vs Exponential

The following table gives the running times for several hypothetical algorithms over input of size n . It is assumed that they are run on a computer which can carry out one million instructions per second.

	10	20	30	40	50	60
n	.00001 seconds	.00002 seconds	.00003 seconds	.00004 seconds	.00005 seconds	.00006 seconds
n^2	.0001 seconds	.0004 seconds	.0009 seconds	.0016 seconds	.0025 seconds	.0036 seconds
n^3	.001 seconds	.008 seconds	.027 seconds	.064 seconds	.125 seconds	.216 seconds
n^5	.1 seconds	3.2 seconds	24.3 seconds	1.7 minutes	5.2 minutes	13.0 minutes
2^n	.001 seconds	1.0 seconds	17.9 minutes	12.7 days	35.7 years	366 centuries
3^n	.059 seconds	58 minutes	6.5 years	3855 centuries	2×10^8 centuries	1.3×10^{13} centuries

Example

Consider the following language:

$$L = \{0^n 1^n \mid n \geq 0\}$$

Clearly this language is decidable, but how much time does a single-tape TM need to decide it?

Consider the following TM M_1 with 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 '0's and '1's appear on the tape:
 - (a) scan across the tape, crossing off a single '0' and single '1'
3. If '0's still remain after all the '1's have been crossed out, or vice-versa, reject. Otherwise, if the tape is empty, accept.

Example

Consider stages separately.

1. Scan across the tape and reject if a '0' is found to the right of a '1'.
 - Scanning requires n steps.
 - Repositioning head requires n steps.
 - Total is $2n = O(n)$ steps.
2. Repeat the following if both '0's and '1's appear on the tape
 - (a) scan across tape, crossing off a single '0' and a single '1'
 - Each scan requires $O(n)$ steps
 - because each scan crosses off two symbols, at most $n/2$ scans can occur.
 - Total is $(n/2)O(n) = O(n^2)$ steps.
3. If '0's still remain after all '1's have been crossed out, or vice-versa, reject. Otherwise, if the tape is empty, accept.
 - Single scan requires $O(n)$ steps.
 - Total is $O(n)$ steps.

Example

Total cost for stages:

1. $O(n)$
2. $O(n^2)$
3. $O(n)$

which is $O(n^2)$. Can we do any better?

Consider the following alternative TM M_2 with 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 '0's and '1's appear on the tape:
 - (a) scan across the tape, checking whether the total number of '0's and '1's is even or odd. If odd, reject.
 - (b) Scan across the tape, crossing off every other '0' (starting with the first), and every other '1' (starting with the first).
3. If no '0's or '1's remain, accept, otherwise reject.

Example

1. Scan across the tape and reject if a '0' is found to the right of a '1'.
 - As before, this requires $2n = O(n)$ steps.
2. Repeat the following if both '0's and '1's appear on the tape
 - (a) scan across the tape, checking whether the total number of '0's and '1's is even or odd. If odd, reject.
 - (b) Scan across the tape, crossing off every other '0' (starting with the first), and every other '1' (starting with the first).
 - Each scan requires $O(n)$ steps.
 - Because each scan crosses off half the '0's and '1's, at most $1 + \log_2 n$ scans can occur.
 - Total is $(1 + \log_2 n)O(n) = O(n \log n)$ steps.
3. If no '0's or '1's remain, accept, otherwise reject.
 - Single scan requires $O(n)$ steps.

This is therefore $O(n \log n)$.

A Two-Tape TM

Consider the following two-tape TM M_3 with input string w :

1. Scan across the tape and reject if a '0' is found to the right of a '1'.
2. Scan across '0's to the first '1', copying '0's to tape 2.
3. Scan across '1's on tape 1 until the end. For each '1', cross off a '0'. If no '0's left, reject.
4. If any '0's left, reject, otherwise accept.

In this case, the running time is $O(n)$.

Complexity

Deciding $\{0^n 1^n\}$:

- single-tape M_1 : $O(n^2)$
- single-tape M_2 : $O(n \log n)$ (fastest possible)
- two-tape M_3 : $O(n)$

- **Computability**: all “reasonable” models of computation are equivalent (Church-Turing thesis)
- **Complexity**: choice of model affects time complexity, but all “reasonable” models of computation are polynomially equivalent.

Thus the class P is invariant for all models of computation polynomially equivalent to a deterministic single-tape TM.

Question: is a problem solvable in

- linear time? model-specific
- polynomial time? model-independent

Problems Belonging to P

Given

- a directed graph G
- nodes s and t

is there a path from s to t ?

PATH = $\{\langle G, s, t \rangle \mid G \text{ has a directed path from } s \text{ to } t\}$

A possible solution:

- let m be the number of nodes in G
- any path from s to t need not repeat nodes
- examine each path in G of length $\leq m$
- check if it goes from s to t .

What is the complexity of this algorithm?

There are m^m possible paths, so this algorithm is exponential in the number of nodes

Problems Belonging to P

Another possible solution to PATH:

1. Place mark on s
2. Repeat until no additional nodes are marked:
 - Scan edges of G .
 - If there is an edge (a, b) where a is marked and b is unmarked, then mark b .
3. If t is marked, accept, otherwise reject.

What is the complexity of this algorithm?

- stages 1 and 3 run once.
- stage 2 runs at most m times, because each time (except the last) it marks a new node.

the total number of stages is polynomial.

- stages 1 and 3 are polynomial.
- stage 2 scans the input, so it is also polynomial.

The total time complexity is therefore polynomial.

Problems Belonging to P

Two numbers are relatively prime if 1 is the largest integer that evenly divides them both.

- 10 and 21 are relatively prime
- 10 and 22 are not

$\text{RELPRIME} = \{ \langle x, y \rangle \mid x \text{ and } y \text{ are relatively prime} \}$

A possible solution (the Euclidian algorithm): On input $\langle x, y \rangle$

1. Repeat until $y = 0$
 - $x \leftarrow x \bmod y$
 - exchange x and y
2. output x

If the result is 1, accept, otherwise reject.

Problems Belonging to P

What is the complexity of this algorithm?

- each execution of step 1 cuts x by at least half
- after each loop $x < y$
- values swapped
- number of steps $\min(\log_2 x, \log_2 y)$

The total time complexity is therefore polynomial.