

Preliminaries for Complexity Theory

Complexity theory is the study of the efficiency of computation, in particular, *time-complexity* and *space-complexity*. *Tractable* problems are those that can be solved efficiently.

Assumptions:

1. Model = multitape TM \approx multivar. program.
 2. Numbers are encoded in binary.
 3. n = size of input (number of elements or bits)
 4. Interested in worst-case
-

Order Notation

$f(n)$ is $O(g(n))$ (f has order at most g) if there are constants c and k such that:

$$f(n) \leq c g(n) \text{ for all } n \geq k$$

$f(n)$ is $\Omega(g(n))$ (f has order at least g) if there are constants c and k such that:

$$f(n) \geq c g(n) \text{ for all } n \geq k$$

$f(n)$ is $\Theta(g(n))$ (f has the same order as g) if there are constants c_1 , c_2 , and k such that:

$$c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ for all } n \geq k$$

Linear Search Example

Problem: Determine if the key x is in a set of n numbers x_1, x_2, \dots, x_n .

Linear Search Algorithm: Compare x to each of the n numbers.

Single-tape solution: Must go back and forth on the tape for each bit comparison. This is $O(m^2)$, where m is the number of bits.

Two-tape solution: Copy key to the second tape. Read-write head on first tape never has to go left. This is $O(m)$, where m is the number of bits.

$a^n b^n$ Example

Problem: Recognize $\{a^n b^n : n \geq 1\}$.

Single-tape solution: Must go back and forth to match each a with a b . This is $O(n)$.

Two-tape solution: Copy all the a 's to a second tape, then match them against the b 's on the first. This is $O(n)$.

Context-Free Grammar Example

Problem: Determine if w is in a given context-free language.

Exhaustive Search Algorithm: Generate all possible derivations. This is $O(n^M)$ where $n = |w|$ and $M =$ number of symbols.

CYK Algorithm (Section 6.3): $O(n^3)$

Nondeterministic Algorithm: Guess derivation of w . This is $O(n)$ if grammar is in Chomsky normal form.

Satisfiability Problem

Problem: Given a Boolean expression in conjunctive normal form using variables x_1, \dots, x_n , determine if there is any Boolean assignment to the variables that make the expression true.

In conjunctive normal form, the expression is a conjunction of terms: $t_1 \wedge t_2 \wedge \dots \wedge t_k$.

Each term t_i is a disjunction of literals:

$$l_{i1} \vee l_{i2} \vee \dots \vee l_{ip}.$$

Each literal is a variable or its negation.

Evaluating an assignment: If we use one tape to store a value assignment, then it is at least $O(n)$ to find the value of a single variable. Evaluating an assignment is $O(mn)$ where m is the length of the expression.

Exhaustive search: There are 2^n possible assignments. This approach is $O(m2^n)$.

Nondeterministic Algorithm: It is $O(mn)$ to guess an assignment and evaluate the expression.