Complexity Theory

→ What we will see next in this course is an introduction to the area of complexity theory.

→ There are several more models of computation where resources come into play. Most important resources are time, space, randomness, circuits, parallelism, interaction etc.

→ The goal of complexity theory is to understand computational problems in terms of resources needed. We will see "complexity classes" which are classes based on computational problems.

→ In this course, we will see two resources.

- Time Complexity
- Space Complexity

→ From now on, we will only study decidable languages.

→ But we will do a more careful analysis of the resources used in the computation.

Time Complexity

no. of steps

Def 7.1: **Running time** or **time complexity** of a Turing machine M is the function $f: \mathbb{N} \to \mathbb{N}$ where $f(n)$ is the maximum time taken by M to accept/reject an input of length n.

Def 7.2: **Landau's O-notation**.

If $f, g: \mathbb{N} \to \mathbb{R}^+$, we say that

$f(n) = O(g(n))$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)} \leq c$ for some constant c.

$f = 3n^2 + 2n + 5$

$f = O(n^2)$

$f = o(n^3)$

$f(n) = o(g(n))$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$

$f(n) = \Omega(g(n))$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)} \geq c$ for some const. c.

We say $f(n) = \Theta(g(n))$ if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

$f = 3n^3 + 5n^4 + 10$

$f = \Theta(n^4)$

Exercise: Familiarise yourself with these symbols.

Def 7.7: $\mathrm{DTIME}(t(n))$ ($\mathrm{TIME}(t(n))$ in the book) is the set of languages that are decided by a DTM in time $O(t(n))$. (time = no. of steps)

TM is deterministic, and may be multitape.
Usually we care if $t(n)$ is a polynomial, or not.

$n, n^2, n^3 \ldots$

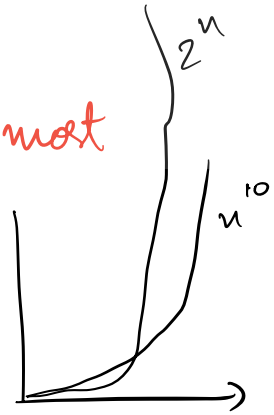
Def 7.12: $P = \bigcup_{k=1}^{\infty} \mathrm{DTIME}(n^k)$

Why should we study the class P?

→ Stands for all efficient / practical algorithms.

→ A robust class, independent of most models of computation

→ Exponential vs. Polynomial



2^n, 3^n

$f(n) = 3^n + n^2 + 5n^3$

n^2, n^3, n^{10}

$f(n) = n^6 + 5n^2$

PATHS, TREES, AND FLOWERS

1963?

JACK EDMONDS

1. Introduction. A *graph* G for purposes here is a finite set of elements called *vertices* and a finite set of elements called *edges* such that each edge

2. Digression. An explanation is due on the use of the words "efficient algorithm." First, what I present is a conceptual description of an algorithm and not a particular formalized algorithm or "code."

For practical purposes computational details are vital. However, my purpose is only to show as attractively as I can that there is an efficient algorithm. According to the dictionary, "efficient" means "adequate in operation or performance." This is roughly the meaning I want—in the sense that it is conceivable for maximum matching to have no efficient algorithm. Perhaps a better word is "good."

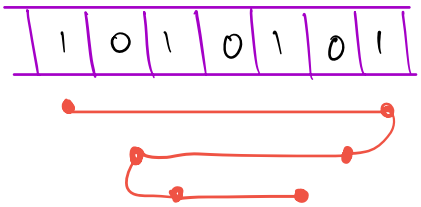
I am claiming, as a mathematical result, the existence of a *good* algorithm for finding a maximum cardinality matching in a graph.

There is an obvious finite algorithm, but that algorithm increases in difficulty exponentially with the size of the graph. It is by no means obvious whether *or not* there exists an algorithm whose difficulty increases only algebraically with the size of the graph.

The mathematical significance of this paper rests largely on the assumption that the two preceding sentences have mathematical meaning. I am not prepared to set up the machinery necessary to give them formal meaning, nor is the present context appropriate for doing this, but I should like to explain the idea a little further informally. It may be that since one is customarily concerned with existence, convergence, finiteness, and so forth, one is not inclined to take seriously the question of the existence of a *better-than-finite* algorithm.

PALINDROME = $\{ w \mid w = w^R \}$

Algorithm 1

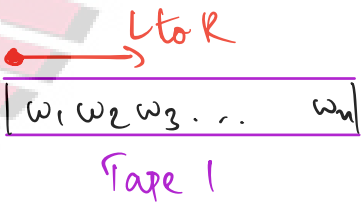


Let $n = |w|$.

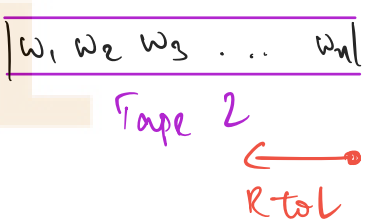
1. Check $w_1, w_n, w_2, w_{n-1}, \ldots$
2. Accept if for all $i \leq n/2$, $w_i = w_{n+1-i}$
3. Else reject.

Have to make $(n-1) + (n-2) + (n-3) + \ldots = O(n^2)$ moves on the tape.

Algorithm 2



1. Copy w to tape 2. $\rightarrow O(n)$



2. Move head of tape 2 to the right most end. $\rightarrow O(n)$
3. As head of tape 1 moves from L to R, move head of tape 2 from R to L. $\rightarrow O(n)$
4. Accept if $w = w^R$.

In this algorithm, tape heads move $O(n)$ steps.
But we need 2 tapes.

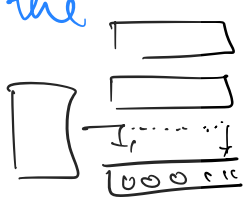
PALINDROME $\in P$

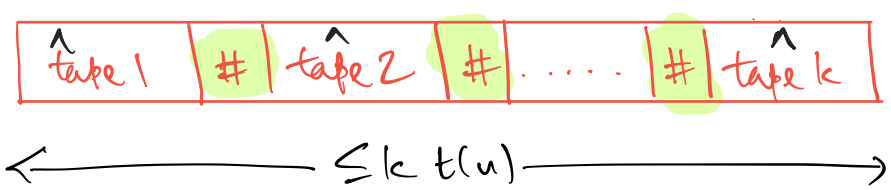
Theorem 7.8: Let $t(n)$ be such that $t(n) \geq n$.

Then every multitape TM running in $t(n)$ time has an equivalent single tape TM running in time $O((t(n))^2)$.

Proof: (Theorem 3.13 from lecture 29)

We do a careful calculation of the approach followed in the proof of Theorem 3.13. Let k be the number of tapes. We simulate the k tapes in a single tape as follows.





1. Put the tape in the above format.
   Initially tape 1 has the input, while

the other tapes are empty.

2. Make 1 pass for the head locations.
3. Simulate the transition of the k-tape machine.
4. Make another pass to update.

For each step of the k-tape machine, the single tape machine needs to make 2 passes – this takes $2 * k\,t(n)$ time. The k-tape machine needs to make $t(n)$ steps.

Total time needed for the single tape machine $= 2\,k\,t(n) * t(n)$

$= O((t(n))^2)$

Class P: n^k for some constant k.

Polynomial : $O(n^k)$

Exponential : $O(k^n)$

Examples:

(1) CONNECTED : Given a graph G, is it connected? $\in P$.

Do BFS / DFS on the graph.

(2) 3-COLORABLE : Given a graph G, can we colour it using 3 colours? (with each adjacent pair of vertex receiving distinct colors)

Not known to be in P.

(3) RELPRIME : Given integers x, y, are they relatively prime? $\in P$.

Is $\gcd(x,y) = 1$?

(4) PRIME : Given integer N, is it prime? $\in P$

$\sim \sqrt{N}$

(5) $A \times B$ for two $n \times n$ matrices. $\in P$

(6) TSP : Travelling Salesman Problem.

Not known to be in P

(7) SUBSET-SUM : Given a set S of integers,

and a target sum t, is there a subset of S that sums to t. Not known to be in P.

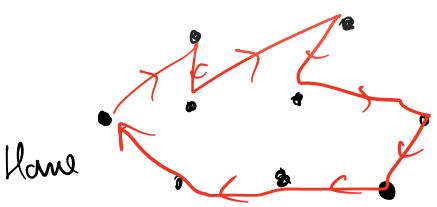
$$S = \{1, 3, 7, 10, 16\}$$

Is there a subset that sums to $t = 12$?

For PRIME, trying out all prime factors upto $\sqrt{N}$ is NOT a poly time approach. This is because $\sqrt{N}$ is not polynomial in the length of the input $\approx \log N := n$.

Then $\sqrt{N} = \sqrt{2^n} = 2^{n/2}$

Exponential in n, which is the length of the input.



TSP