P

Formalizing running time

We used ad-hoc metrics

- #comparison for comparison sort
- #multiplications for matrix multiplication

We need a generic notion of running time

- Back to Turing machine!

Formalized running time

Definition: Let $f: \{0,1\}^* \to \{0,1\}^*$ and let $T : \mathbb{N} \to \mathbb{N}$ be some function and let M be a Turing machine. We say that M *computes* f if for every $x \in \{0,1\}^*$, whenever M is initialized to the start configuration on input x, then it halts with $f(x)$ written on its output tape. We say M *computes* f *in* $T(n)$*-time* if its computation on every input x requires at most $T(|x|)$ steps.

Formalized running time

Definition: Let $f: \{0,1\}^* \to \{0,1\}^*$ and let $T : \mathbb{N} \to \mathbb{N}$ be some function and let M be a Turing machine. We say that M *computes* f if for every $x \in \{0,1\}^*$, whenever M is initialized to the start configuration on input x, then it halts with $f(x)$ written on its output tape. We say M *computes* f *in* $T(n)$*-time* if its computation on every input x requires at most $T(|x|)$ steps.

Robustness of definition

- non-binary alphabet Γ: encode a symbol using $\log|\Gamma|$ bits ($\log|\Gamma|$-X blowup)
- TM with k tapes: encode to one tape via interleaving, and simulate each step by sweeping back and forth ($T(n)$ steps for each original step, quadratic blowup)

Decision problems

Decision Problems:

- The output is *yes* or *no* (1 or 0)
- The function $f: \{0,1\}^* \to \{0,1\}$ is essentially a subset of $\{0,1\}^*$ and called a *language*
- Goal: decide if the given input belongs to the language

Definition: Let $L \subseteq \{0,1\}^*$ and let $T : \mathbb{N} \to \mathbb{N}$ be some function and let M be a Turing machine. We say that M *decides* L in $T(n)$ time if for every $x \in \{0,1\}^*$, whenever M is initialized to the start configuration on input x, then it halts in at most $T(|x|)$ steps, and accepts if and only if $x \in L$.

The class $\mathbf{P}$

Definition (class DTIME): Let $T : \mathbb{N} \to \mathbb{N}$ be some function. A language L is in $\mathbf{DTIME}(T(n))$ iff there is a Turing machine that decides L in time $O\big(T(n)\big)$.

Definition (class P): Let $\mathbf{P} = \bigcup_{c\geq 1} \mathbf{DTIME}(n^c)$

E.g., the graph connectivity (reachability) problem

Example: Graph Connectivity

Input: a directed graph G (with n nodes)

Output: decide if G is connected

Algorithm: compute the transitive (adjacency matrix multiplication, in $O(n^4)$ steps)

Complexity: $\mathbf{DTIME}(n^4)$, and also $\mathbf{P}$

Why Polynomial?

Cobham-Edmonds Thesis (1965):

$\mathbf{P} = \text{The collection of tractable computational problems}$

Platform-independence

- any problem in P can be solved in polynomial time on *any* reasonable computational model
- Strong Church-Turing Thesis: “Every physically realizable computation model can be simulated by a TM with polynomial overhead”

Encoding-independence

- if a problem is in P for one encoding, it will be in P even if input instances are encoded in a different manner

Low-order polynomial

- Most P problems in practice have low orders ($\Theta(n^3)$ or $\Theta(n^5)$)
- Even if the current best algorithm is $\Theta(n^{100})$, much better running time will likely soon be discovered

Closure properties

- The class P is closed under addition, multiplication, composition, etc.