Complexity Cheat Sheet

Alessandro Cheli - 2020 - University of Pisa

Turing Machines

Deterministic TMs are seen in the *Computability Theory Cheat Sheet*.

K-tapes Turing Machines

Let $k \geq 1$ be the number of tapes. A k-tape TM is a quadruple (Q, Σ, δ, q_0). Special symbols $\#, \triangleright, \in \Sigma$, while $L, R, - \notin \Sigma$. Since we are only considering decision problems, YES, NO $\notin Q$ are the halting states. The transition function differs but is subject to the same restrictions as a classical TM.

$$\delta : Q \times \Sigma^k \to Q \cup \{\text{YES}, \text{NO}\} \times (\Sigma \times \{L, R, -\})^k$$

IO Turing Machines

Are useful to study space complexity. Any k-tape TM $M = (Q, \Sigma, \delta, q_0)$ is of IO type $\Leftrightarrow$ δ is subject to the following constraints. Consider $\delta(q_1, \sigma_1, \ldots, \sigma_k) = (q', (\sigma'_1, D_1), \ldots, (\sigma'_k, D_k))$

$\sigma'_1 = \sigma_1$	First tape is read-only
$D_k = R$ or $(D_k = -) \Rightarrow \sigma'_k = \sigma_k$	Last tape is write-only
$\sigma_1 = \# \Rightarrow D_1 \in \{L, -\}$	Input tape is right-bounded

Nondeterministic Turing Machines

A **nondeterministic** TM is a quadruple $N = (Q, \Sigma, \Delta, q_0)$ where Q, Σ, q_0 are the same as in other turing machines. They can be of IO and k-tape type. The difference rilies in the fact that

$$\Delta \subseteq (Q \times \Sigma) \times ((Q \cup \{\text{YES}, \text{NO}\}) \times \Sigma \times \{L, R, -\})$$

Is not a transition function but a **transition relation**. The same syntax as for other TMs is used for nondet. TMs configurations and computations. A nondeterministic TM N decides I if and only if $x \in I \Leftrightarrow$ there exists **at least** a computation such that $(q_0, \underline{\triangleright}, x, \triangleright, \ldots, \triangleright) \to^*_N (\text{YES}, w_1, \ldots, w_k)$

Note. A single computation on a nondeterministic TMs is not a tree. Many computations together, can be arranged in a tree.

The **degree** d of nondeterminism of a NDTM $N = (Q, \Sigma, \Delta, q_0)$ can now be defined as $d = \max\{\deg(q, \sigma) \mid q \in Q, \sigma \in \Sigma\}$, where $\deg(q, \sigma) = \#\{(q', \sigma', D) \mid ((q, \sigma), (q', \sigma', D)) \in \Delta\}$

Computation Table

A computation table T_M is a square matrix where rows represent the configuration at current computation step number (index i) of a polynomial time deterministic, 1-tape TM M on an input x. The column (index j) represent the j-th slot on tape. Since M decides I in time $\mathcal{O}(|x|^k)$, and space is limited by time, the matrix is of size $|x|^k$. To define it, the definition of the deterministic TM has to be slightly changed. First we use $\Sigma' = \Sigma \cup \{\sigma_q \mid \sigma_q \in \Sigma \times (Q \cup \{\epsilon\})\}$ as the alphabet, enriching symbols with a subscript containing the current state. These *subscript* symbols keep track of the head position and current state at time i. A new machine M' can be built from M, and this new construction does not take M' out of $\mathcal{P}$. Computation tables follow these rules:

$\lvert x\rvert \geq 2$	$\forall i. T(i, 1) = \triangleright$
$\forall j.\ 2 \leq j \leq \lvert x\rvert + 1$	$T(1, j)$ contains the $j-1$-th symbol of x
$\forall j.\ \lvert x\rvert + 2 \leq j \leq \lvert x\rvert^k$	$T(1, j)$ contains $\#$
$\forall i. T(i, \lvert x\rvert^k) = \#$	Always use less space than time
$\forall i \geq 2, j \geq 2.\ T(i, j)$	depends only on $T(i-1, j-1), T(i-1, j)$ and $T(i-1, j+1)$.

Initial configuration starts from $\triangleright \sigma_{q_0}$ and not from $\underline{\triangleright} \sigma_{q_0}$

The head never positions on column 1 (on $\triangleright$).

If $T(i, j) \in \{\sigma_{\text{YES}}, \sigma_{\text{NO}}\}$, add an auxiliary state that moves the tape head to $T(i, 2)$. M accepts $x \Leftrightarrow \exists i.\ T(i, 2) = \sigma_{\text{YES}} = T(|x|^k, 2)$, therefore all lines between that index i and $|x|^k$ are equal to line i.

Time and Space Measures

The size of a datum x, is a total (easily) computable function $|x|$, which returns $n \in \mathbb{N}$, computed from memory usage, number of components of the datum, or other criteria. Given a TM M (the same applies for IO and k-tape machines), t is the time needed by a to decide $x \in I$, given by

$$(q_0, \underline{\triangleright} x) \to^t (H, w) \text{ with } H \in \{\text{YES}, \text{NO}\}$$

Deterministic Time Measures

A measuring function $f : \mathbb{N} \to \mathbb{N}$ is said to be **appropriate** $\Leftrightarrow$ it is strictly increasing and $\exists$ a TM M that $\forall x.\ M(x) = \diamond^{f(|x|)}$, in time $\mathcal{O}(f(|x|) + |x|)$ and space $\mathcal{O}(f(|x|))$.

M decides I in *deterministic time* f if $\forall x$, the time t required to compute $M(x)$ is $\leq f(|x|)$. A **Complexity Class** can now be defined:

$$\text{TIME}(f) = \{I \mid \exists M \text{ deciding } I \text{ in deterministic time } f\}$$

Asymptotic notation is commonly used to denote time complexity and to simplify. Constants are commonly ignored (treated as r)

$$\mathcal{O}(f) = \{g \mid \exists r \in \mathbb{R}^+.\ g(n) < rf(n) \text{ almost everywhere}\}$$

Nondeterministic Time Measures

A nondet. TM N decides I in **nondeterministic time** $f(n) \Leftrightarrow N$ decides I and $\forall x \in I$. $\exists t$ such that N *halts and accepts* x in t steps, with $t \leq f(|x|)$. Because of nondeterminism, there can be many computations that satisfy (or don't) those requirements. We consider only the shortest accepting computation.

$$\text{NTIME}(f) = \{I \mid \exists N \text{ deciding } I \text{ in non deterministic time } f\}$$

Deterministic and Nondeterministic Space Measures

To measure space, we would have to edit the definition of TMs, by adding an additional *right bumper* $\triangleleft$ that denotes the end of the tape. This way (k-tape and IO) TMs will remember *how many slots* were visited. $\forall$ computation halting in a state $(H, w_1, \ldots, w_n)$ with $H \in \{\text{YES}, \text{NO}\}$, the space required by a k-tape IO TM is $\sum_{i=2}^{k-1} |w_i|$. M decides I in deterministic space $f(n)$ if $\forall x$ the spaced required to compute $M(x)$ is $\leq f(|x|)$, and it follows that $I \in \text{SPACE}(f(n))$. The same idea can be applied to nondeterministic machines, considering the additional restraint on accepting, to define NSPACE.

Complexity Classes

$$\mathcal{P} = \bigcup_{k \geq 1} \text{TIME}(n^k) \qquad \mathcal{NP} = \bigcup_{k \geq 1} \text{NTIME}(n^k)$$

$$\text{PSPACE} = \bigcup_{k \geq 1} \text{SPACE}(n^k) \qquad \text{NPSPACE} = \bigcup_{k \geq 1} \text{NSPACE}(n^k)$$

$$\text{EXP} = \bigcup_{k \geq 1} \text{TIME}(2^{n^k}) \qquad \text{LOGSPACE} = \bigcup_{k \geq 1} \text{SPACE}(k \times \log(n))$$

$\mathcal{P}$, PSPACE, LOGSPACE are closed with respect to model transformations and are therefore robust classes of problems. $\mathcal{NP}$ can also be defined as the set of problems that allow *verification in polynomial time*

Hierarchy

$$\text{LOGSPACE} \subseteq \mathcal{P} \subseteq \mathcal{NP} \subseteq \text{PSPACE} = \text{NPSPACE} \subset \text{EXP} \subset R \subset RE$$

Since LOGSPACE $\subsetneq$ PSPACE, at least one of those inclusions is strict:

$$\text{LOGSPACE} \subseteq \mathcal{P}, \mathcal{P} \subseteq \mathcal{NP}, \mathcal{NP} \subseteq \text{PSPACE}$$

It is not yet know which one of those is a strict inclusion.

Theorem. LOGSPACE $\subseteq \mathcal{P}$. *Proof.* Let $I \in$ LOGSPACE. There $\exists$ a TM that computes $x \in I$ in $\mathcal{O}(\log|x|)$ space. M can range through $\mathcal{O}(|x| \log|x| \#Q \#\Sigma^{\log|x|})$ configurations. A halting computation cannot cycle on configurations. So M computes in $\mathcal{O}(|x|^k)$ for some k.

Theorem. (Savitch) NPSPACE = PSPACE.

Theorem. Hierarchy of time and space. If f is appropriate, then $\text{TIME}(f(n)) \subsetneq \text{TIME}((f(2n+1))^3)$, and $\text{SPACE}(f(n)) \subsetneq \text{SPACE}(f(n) \times \log f(n))$. **Corollary.** $\mathcal{P} \subsetneq \text{EXP}$

Proof. $\mathcal{P} \subset$ EXP is obvious because 2^n grows faster than any polynomial. It is strict because $\mathcal{P} \subseteq \text{TIME}(2^n) \subsetneq \text{TIME}((2^{(2n+1)})^3) \subseteq \text{TIME}(2^{n^2})$. This corollary, together with the fact that $\text{NTIME}(f(n)) \subseteq \text{TIME}(c^{f(n)})$ lets us conclude that $\mathcal{NP} \subseteq$ EXP.

Theorem. Hierarchy 2. Let f be an appropriate measuring function, and k a constant. Then

$\text{SPACE}(f(n)) \subseteq \text{NSPACE}(f(n)) \qquad \text{TIME}(f(n)) \subseteq \text{NTIME}(f(n))$

$\text{NSPACE}(f(n)) \subseteq \text{TIME}(k^{\log n + f(n)})$

Theorem. Arbitrarily Hard Problems: $\forall g$ total computable func. $\exists I \in \text{TIME}(f(n))$ and $I \notin \text{TIME}(g(n))$, with $f(n) > g(n)$ almost everywhere.

Using arbitrary measuring functions entails bizarre consequences:

Theorem. Blum speedup $\forall h$ total computable func. $\exists$ a problem I such that $\forall M$ algorithm deciding I in time f, $\exists M'$ deciding I in time f' such that $f(n) > h(f'(n))$ almost everywhere. This theorem guarantees that there are problems that have no optimal algorithm. Although given an h, the problems built for this theorem are artificial and not useful, and we do not know how to construct them.

Theorem. Borodin's Gap There exists f total computable such that $\text{TIME}(f(n)) = \text{TIME}(2^{f(n)})$

These last two theorems are also valid for *space measures*.

Theorems on Complexity of Turing Machines

Theorem. ***Reduction of tapes*** *Let M be a k-tape TM, deciding I in deterministic time f, then $\exists$ a 1 tape TM M' that decides I in time $\mathcal{O}(f^2)$*

Proof. (Only a draft): Build a 1 tape TM M' by introducing two symbols $\triangleright'$ and $\triangleleft'$ to denote the start and end of the k-th tape. Introduce $\#\Sigma$ new symbols $\overline{\sigma_i}$ to remember each tape's head location. To generate the initial configuration $(q, \triangleright \triangleright' x \triangleleft' (\triangleright' \triangleleft')^k)$, $2k + \#\Sigma$ states and $\mathcal{O}(|k|)$ steps are needed. To simulate a move of M, M' iterates the input datum from left to right, and back, 2 times: find the marked $\overline{\sigma_i}$ symbols, the second time write the new symbols. If a tape has to be extended, the $\triangleleft'$ parens have to be moved and a cascade happens. This takes $\mathcal{O}(f(|x|))$ time, for each move of M. Since M takes $\mathcal{O}(f(|x|))$ time to compute an answer, M' will take $\mathcal{O}(f(|x|)^2)$ $\square$

Corollary. *Parallel machines are polinomially faster than sequential machines.*

$\boxed{\text{A machine cannot use more space than time}}$

Theorem. Linear speedup If $I \in \text{TIME}(f)$, then $\forall \epsilon \in \mathbb{R}^+.\ I \in \text{TIME}(\epsilon \times f(n) + n + 2)$. *Proof.* (Draft): Build M' from M solving I, compacting m symbols of M into 1 of M', in function of ϵ. The states of M' will be triples $[q, \sigma_{i_1}, \ldots, \sigma_{i_m}, k]$ meaning the TM is in state q and has cursor on k-th symbol of $\sigma_{i_1}, \ldots, \sigma_{i_m}$. M' then needs 6 steps to simulate m steps of M. Therefore, M' will take $|x| + 2 + 6 \times \lceil \frac{f(|x|)}{m} \rceil$. Then $m = \lceil \frac{6}{\epsilon} \rceil$. $\square$

Same principle can apply to SPACE with **linear space compression**. If $I \in \text{SPACE}(f(n))$, then $\forall \epsilon \in \mathbb{R}^+.\ I \in \text{SPACE}(\epsilon \times f(n) + 2)$

Theorem. *For each k-tape TM M that decides I in deterministic time f there exists an IO TM M' with k+2 tapes that computes I in time $c \times f$ for some constant c.*

Proof. M' copies the first M tape to the second tape, in $|x| + 1$ steps. Then, M' operates identically to M. If and when M halts, M' copies the result to the tape $k + 2$, in at most $f(|x|)$ steps. M' computation was $2f(|x|) + |x| + 1$ steps long. $\square$

Theorem. ***Exponential loss in determinization, or bruteforce*** *If $I \in NTIME(f(n))$ and is computed by k-tape N, it can also be computed by a deterministic TM M with $k+1$ tapes in time $\mathcal{O}(c^{f(n)})$ with an exponential loss of performance. In short,* $\text{NTIME}(f(n)) \subseteq \text{TIME}(c^{f(n)})$

Proof. Let d be the degree of nondeterminism of N. $\forall q \in Q, \sigma \in \Sigma$ sort $\Delta(q, \sigma)$ lexicographically. Every computation of length t carried by N is a sequence of choices that can be seen as a sequence of natural numbers $(c_1, \ldots, c_t)$ with $c_i \in [0..d-1]$. The det. TM M considers these successions in an increasing order, visiting the tree of nondeterministic computations, one at a time. Therefore $M(x) \downarrow \Leftrightarrow N(x) \downarrow$, also, all the possible simulations can be $\mathcal{O}(d^{f(n)+1})$. $\square$

Problems in P and NP

A problem I *reduces efficiently* to I' ($I \leqslant_{\text{logspace}} I'$) if $\exists f \in$ LOGSPACE such that $x \in I \Leftrightarrow f(x) \in I'$. Let $\mathcal{D}, \mathcal{E} \in \{\mathcal{P}, \mathcal{NP}, \text{EXP}, \text{PSPACE}, \text{NPSPACE}\}$ and $\mathcal{D} \subseteq \mathcal{E}$, then $\leqslant_{\text{logspace}}$ classifies {LOGSPACE} and $\mathcal{E}$. Also, $\leqslant_{\text{logspace}}$ and $\leqslant_{\mathcal{P}}$ classify $\mathcal{D}$ and $\mathcal{E}$. (See the *computability theory cheat sheet*).

Note. A TM operating in logarithmic space, composed to another LOGSPACE machine is still in LOGSPACE.

NP complete problems

Traveling Salesman Problem Let G be a directed weighted graph with n vertices, all connected to each other. Let $d(i, j)$ be the cost function for (weight) on edge (i, j). The problem consists in finding the *hamiltonian cycle* (permutation of nodes) with the minimum cost: $\sum_{1 \leq i \leq n-1} d(i, i+1)$. To see this as a decision problem we have to check if the total cost of the path is $\leq$ of a treshold B. A *deterministic* TM that bruteforces the problem first builds *all permutations* of $[1..n]$ in $\frac{(n-1)!}{2}$ steps, and then searches for the first permutation with cost $\leq B$ with costs $\mathcal{O}(n^3)$. A *nondeterministic* TM solves the problem in $\mathcal{O}(n^3)$ by first generating all sequences of numbers $[1..n]$ of length n *nondeterministically*, and then polynomially verifying that one of the sequence is: a path (in $\mathcal{O}(n^2)$) and costs less than B (in $\mathcal{O}(n^3)$).

SAT or Satisfiability Problem: Given a boolean expression B, the problem consists in deciding if there $\exists$ a boolean assignment $\mathcal{V} \vDash B$. We consider only boolean expressions given in conjunctive normal form $((a \vee b) \wedge (c \vee d \vee e) \ldots)$, which is guaranteed to exist for any boolean expression.

Theorem. (Cook) SAT is $\mathcal{NP}$ complete. *Proof.* Since CIRCUIT SAT $\leqslant_{\text{logspace}}$ SAT, proving that CIRCUIT SAT is $\mathcal{NP}$-complete is enough. Then, $\forall I \in \mathcal{NP}, I \leqslant_{\text{logspace}}$ CIRCUIT SAT. Let $I \in \mathcal{NP}$, solved by N in time n^k, we build $f \in$ LOGSPACE such that $x \in I \Leftrightarrow f(x)$ is a satisfiable boolean formula. We assume that N has degree of nondeterminism $d = 2$ (if not, an equivalent machine can be built), so we can encode a succession of choices of length $\leq |x|^k$, as a sequence of binary bits B. Only if B is fixed, we can build the *computation table* of $(N(x), B)$. As seen in the proof of $\mathcal{P}$-completeness of CIRCUIT VALUE, one can build a boolean circuit C for each cell, but it also depends on b_{i-1} since Δ_N is not a function and $d = 2$. Analogously, a circuit for the complete computation of C_N, can be built with $(|x|^k - 1)(|x|^k - 1)$ copies of C, but each copy will have $3m + 1$ inputs. $\square$.

CLIQUE Problem: Determine if a undirected graph $G = (V, E)$ has a clique of degree k. **Theorem: SAT $\leqslant_{\text{logspace}}$ CLIQUE.** *Proof.* Given a bool. expr. B in CNF $\bigwedge_{1 \leq k \leq n} C_k$, build $f(B) = (V, E)$ such that V is the set of literal occurences in B and $E = \{(i, j) \mid i \in C_k \Rightarrow (j \notin C_k \wedge i \neq \neg j)\}$. $(\mathcal{V} \vDash B) \Leftrightarrow f(B)$ has a clique of degree k. $f \in$ LOGSPACE since 2 counters are needed. $\square$

HAM or Hamiltonian Path: The problem consists in deciding if a direct graph there exists a path, called *hamiltonian*, that goes through every vertex a single time. (A variant called *hamiltonian cycle* requires the path to go back to the starting node).

Theorem: HAM $\leqslant_{\text{logspace}}$ SAT. *Proof.* Given the direct graph G, we construct $f \in$ LOGSPACE such that $f(G)$ is a boolean formula in conjunctive normal form that is satisfiable $\Leftrightarrow G$ has an hamiltonian path. If G has n vertices, $f(G)$ has n^2 variables, written as $x_{i,j}$ with $1 \leq i, j \leq n$, which represents if the i-th element of the path is the j-th vertex in the graph. $f(G)$ will be the conjuction of these clauses:

$(\neg x_{i,j} \vee \neg x_{k,j})\ i \neq k$	Same node appears once in the path
$(\neg x_{i,j} \vee \neg x_{i,k})\ j \neq k$	Two nodes cannot be the i-th
$(x_{i,1} \vee \ldots \vee \neg x_{i,n})\ 1 \leq i \leq n$	Some node is the i-th
$(x_{1,j} \vee \ldots \vee \neg x_{n,j})\ 1 \leq j \leq n$	Every node is in path
$(\neg x_{k,i} \vee \neg x_{k+1,j})$	$\forall k.\ 1 \leq k \leq n-1$ and $\forall (i, j) \notin A$. If (i, j) is not an edge of G, it must not appear in the path.

It is then immediate to see that $(\mathcal{V} \vDash f(G)) \Leftrightarrow G$ has an hamiltonian path, and that $\mathcal{V}$ represents a permutation of nodes of G. To verify that $f \in$ LOGSPACE, we build a k-tape IO TM computing f that writes on the output tape the first 4 clauses. To do so, $\Sigma = \{tt, ff, \vee, \wedge, \neg, (,), 0, 1\}$ and it needs a work tape containing n, and 3 work tapes containing a binary representation of the 3 variables (counters) seen in the clauses: i, j, k. This requires $\mathcal{O}(\log n)$ bits on work tapes. $\square$

CIRCUIT SAT Problem: Let C a *boolean circuit*, or a direct acyclic graph (V, E) where the n vertices are called ports and edges are sorted pairs. Ports have 0,1 or 2 inputs, and are of sort $s(i) \in \{tt, ff, \neg, lor, \wedge\} \cup X$ where X is the set of variables. Circuit input ports are only variables or truth values and have no inputs. The circuit output is by convention the last port n. A denotational semantic $[\![i]\!]_{\mathcal{V}}$ given a boolean assignment is straightforward to define. The *CIRCUIT SAT* problem consists in deciding if an assignment $\mathcal{V}$ exists such that $\mathcal{V}(C) = tt$. A nondeterministic TM solving the problem will first generate all possible assignments at the same time and then verifies *polinomially* that an assigment satisfies C by letting the truth values *flow* through the circuit.

Theorem. CIRCUIT SAT $\leqslant_{\text{logspace}}$ SAT: *Proof.* Given $C = (V, E)$, with variables in X, build $f \in$ LOGSPACE such that $\exists \mathcal{V}.\ [\![C]\!]_{\mathcal{V}} = tt \Leftrightarrow \exists \mathcal{V}'.\ \forall x \in X.\ V'(x) = V(x) \wedge (V' \vDash f(C))$. For each port g in C build the conjuncts of $f(C)$ as follows:

g	g is the output port in C
$(\neg g \vee x) \wedge (g \vee \neg x)$	if $s(g) = x \in X$, $g \Leftrightarrow x$
$(\neg g \vee \neg h) \wedge (g \vee x)$	if $s(g) = \neg$, $(h, g) \in E$, then $g \Leftrightarrow \neg h$
$(\neg h \vee g) \wedge (\neg k \vee g) \wedge (h \vee k \vee \neg g)$	if $s(g) = \vee$ and $(h, g), (k, g) \in E$, then $g \Leftrightarrow (h \vee k)$
$(\neg g \vee h) \wedge (\neg g \vee k) \wedge (\neg h \vee \neg k \vee g)$	if $s(g) = \wedge$ and $(h, g), (k, g) \in E$, then $g \Leftrightarrow (h \wedge k)$

See the reduction from HAM to SAT to verify that this reduction is logarithmic in space.

Corollary. CIRCUIT VALUE $\leqslant_{\text{logspace}}$ SAT, CLIQUE

P Complete Problems

Theorem. CIRCUIT Value is $\leqslant_{\text{logspace}}$-complete for $\mathcal{P}$: CIRCUIT Value problems are specific cases of CIRCUIT SAT problems where the input ports are only truth values $s(i) \in \{tt, ff\}$ and *can not be variables.*

Proof. For an arbitrary $I \in \mathcal{P}$, build the computation table for the machine M deciding it. Encode each $\rho \in \Sigma'$ as a string of binary bits of length $m = \log_2(\#\Sigma')$. With this representation the computation table will be rectangular, with width $m \times |x|^k$. Since the transition function δ_M is fixed, as in the computation table we can build the boolean representation $S_{i,j}$ of $T_{i,j}$ that depends only on the values in the previous line $i-1$ and columns corresponding to T's $j-1, j, j+1$. We can then build a boolean circuit C, with $3m$ inputs that computes the transition function and outputs m bits corresponding to $S_{i,j}$. To build f, transform T into a circuit C_I that is composed by copies of C for each cell i, j. $(|x|^k - 1)(|x|^k - 2)$ copies are enough because the first and last columns are fixed, and the first row contains the input on tape. By simple induction it is provable than $C_{i,j}$ has $S_{i,j}$ as output $\Leftrightarrow T(i, j) = \rho$, that is encoded as $S_{i,j}$. Since we need a single copy of C for each index in the table i, j, f can be defined in logarithmic space because a TM computing it only needs these index counters on work tapes, stored in base 2. $\square$

Theorem. CIRCUIT VALUE $\leqslant_{\text{logspace}}$ CIRCUIT SAT: *Proof.* Obvious generalization.