Resource Bounded TM's

△ **A TM has unlimited resources: time and space.**

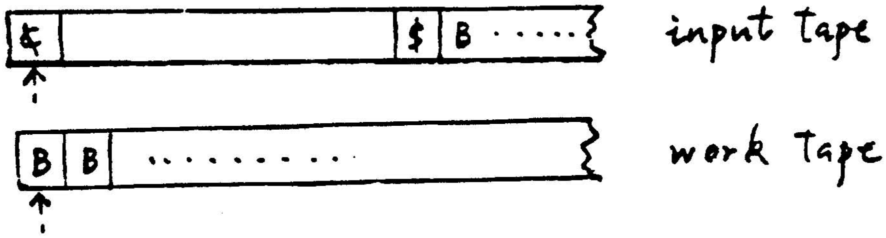
In practice, we need to restrict them.

△ **TMs with restricted resources are the topic of complexity theory. Here, we briefly introduce the basic concepts of P, NP, etc. .**

△ **We use the off-line DTMs and NTMs which have been introduced before. An off-line TM (DTM or NTM) has**

— a read-only input tape;

— a read-write work tape; (storage)



Note:

1. **There are two end-markers at the two ends of the input word. The read-only head never moves out of the two ends.**
2. **Both heads can make three kinds of moves, i.e., L, R, λ.**
3. **The work tape is initially blank.**

△ **Given an off-line DTM M, if each accepted word of length n causes M to visit at most $S(n)$ distinct cells on the work tape, then M is said to be an $S(n)$ space-bounded DTM.**
Note:

(1) We only consider accepted words.

(2) $S(n)$ is a function from N to N.

(3) We only count the cells used on the work tape.

△ **Given an off-line DTM M, if each accepted word of length n causes M to compute at most $T(n)$ steps before M accepts, then M is said to be a $T(n)$ time-bounded DTM.**

△ **Given an off-line NTM M, if each accepting configuration sequence visits at most $S(n)$ distinct cells on the work tape, then M is said to be an $S(n)$ space-bounded NTM.**

△ **Similarly, we define $T(n)$ time-bounded NTMs.**

$\triangle$ **A language L is of**

(1) deterministic–space complexity $S(n)$, if $L = L(M)$ and M is $S(n)$ space-bounded off-line DTM.

(2) deterministic-time complexity $T(n)$.

(3) nondeterministic-space complexity $S(n)$.

(4) nondeterministic-time complexity $T(n)$.

Attention has focused on polynomial functions, since these appear to be the the most practical.

Example:

n	1	10	20	...
$100n^2$	100	10000	40000	...
2^n	2	1024	1048576	...

We define the following
four families of languages:

(1) $\mathcal{L}_{DPSPACE} = \{L \mid L$ **has det.-space complexity** $S(n),$ **for some polynomial** $S(n)\}$

(2) $\mathcal{L}_{DPTIME} = \{L \mid L$ **has deterministic-time complexity** $T(n),$ **for some polynomial** $T(n)\}$

(3) $\mathcal{L}_{NPSPACE} = \{L \mid L$ **has nondet.-space complexity** $S(n),$ **for some polynomial** $S(n)\}$

(4) $\mathcal{L}_{NPTIME} = \{L \mid L$ **has nondet.-time complexity** $T(n),$ **for some polynomial** $T(n)\}$

$\triangle$ **We know that**

$$\mathcal{L}_{DPSPACE} \subseteq \mathcal{L}_{NPSPACE}$$

and

$$\mathcal{L}_{DPTIME} \subseteq \mathcal{L}_{NPTIME}$$

Why?

$\triangle$ **We also know that**

$$\mathcal{L}_{DPTIME} \subseteq \mathcal{L}_{DPSPACE}$$

and

$$\mathcal{L}_{NPTIME} \subseteq \mathcal{L}_{NPSPACE}$$

Why?

$\triangle$ **It has been proven that**

$$\mathcal{L}_{DPSPACE} = \mathcal{L}_{NPSPACE}$$

and that

$$\mathcal{L}_{NPSPACE} \subset \mathcal{L}_{REC}$$

$\triangle$ **So, we have**

$$\underbrace{\mathcal{L}_{DPTIME}}_{P} \subseteq \underbrace{\mathcal{L}_{NPTIME}}_{NP} \subseteq \overbrace{\mathcal{L}_{DPSPACE}}^{\mathcal{L}_{NPSPACE}} \subset \mathcal{L}_{REC}$$

Corresponding to these four families of languages, we have four <u>classes</u> of problems:

(1) PSPACE : the class of all problems that can be solved in deterministic polynomial space.
(2) P : the class of all problems that can be solved in deterministic polynomial time.
(3) NPSPACE : ... in nondeterministic polynomial space
(4) NP: ... in nondeterministic polynomial time.

$$P \subseteq NP \subseteq PSPACE = NPSPACE$$

problems $\Leftrightarrow$ decision problems $\Leftrightarrow$ languages

$$P? =?NP$$

$\triangle$ **We define a relation $\prec$ on problems (languages).**
For L_1, L_2 in $\mathcal{L}_{REC}$,

$$L_1 \prec L_2$$

If L_1 requires no more time than L_2 to accept.

$\triangle$ **$\prec$ is transitive and reflexive.**
It is a pre-order.

$\triangle$ **L is said to be NP-hard if $L' \prec L$ for all L' in $\mathcal{L}_{NPTIME}$.**

$\triangle$ **L is said to be NP-complete if it is also in $\mathcal{L}_{NPTIME}$.**

A clear concept of NP-completness was given by Steven Cook. He showed that satisfiability problem is NP-complete.

SATISFIABILITY PROBLEM (SAT)
INSTANCE : A set U of variables and a collection C of clauses over U.
QUESTION : Is there a satisfying truth assignment for C?

SAT is NP-complete.