Def. 7.1 Let M be a deterministic TM that halts on all inputs. The *running time* or *time complexity* of M is the function $f: \mathbb{N} \to \mathbb{N}$ where $f(n)$ is the maximum number of steps that M takes on any input of length n.

Def 7.2 Let f, g be functions $f, g: \mathbb{N} \to \mathbb{R}^+$

Then we say that $f(n) = O(g(n))$ if there exist $c, n_0 \in \mathbb{Z}_+$ s.t. $\forall n \geq n_0 \quad f(n) \leq c \cdot g(n)$

Def 7.7 Let $t: \mathbb{N} \to \mathbb{R}^+$ be a function.

The *time complexity class* $TIME(t(n))$ is the collection of all languages that are decided by an $O(t(n))$ time TM.

Note that we are only dealing with decision problems: given $w \in \Sigma^*$ does $w \in L$? when is the given language

Remark

Connection between decission and optimization variants of the same problem.

SpT_k: given $G=(V,E)$ and $w: E \to \mathbb{R}_+$, $K \in \mathbb{Z}_+$
Q: is there a spanning tree of weight $\leq K$?

optimization version is MST problem.

Given a MST we can answer SpT_k

Solving MST via $\log_2(opt)$ calls to SpT_k

$k=1$: SpT_1 ÷
$k=2$: SpT_2 ÷
⋮
$k=2^{r-1}$ SpT_k ÷
$k=2^r$ SpT_k + ⟶ binary search between 2^{r-1} and 2^r

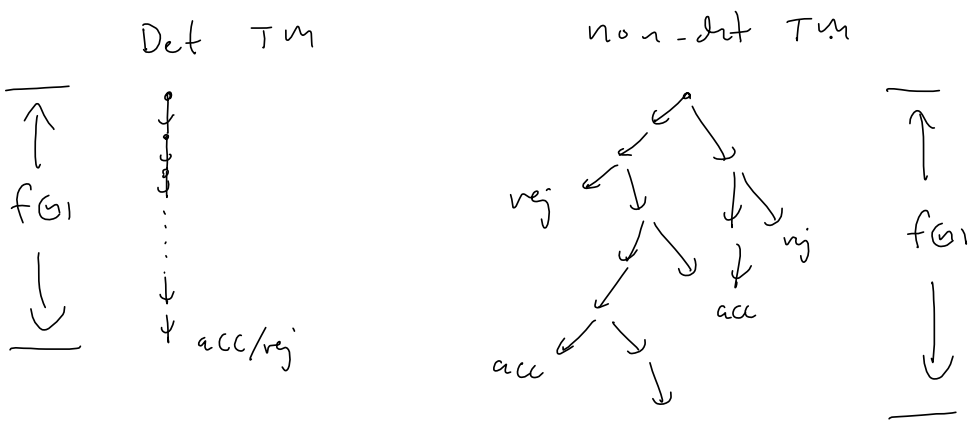
Recall from Chapter 3 (see page 10 in notes from Feb 27)

Thm 7.8

Let $t(n)$ be a function with $t(n) \geq n$.
Then every $t(n)$ time multitape TM has an equivalent $O(t^2(n))$ time single tape TM.

Def 7.9

Let M be a Non-det. TM that is a decider.
The running time of M is the function $f: \mathbb{N} \to \mathbb{N}$, where $f(n)$ is the maximum number of steps that M uses on any branch of its computation on any input of length n.



Recall notes from March:

Thm 7.11 Let $t(n)$ be a function with $t(n) \geq n$

Then every $t(n)$ time non deterministic TM has an equivalent $2^{O(t(n))}$ time equivalent deterministic single tape TM.

All reasonable deterministic computational models are polynomially equivalent.

I.e. any of them can simulate another with only a polynomial increase in running time.

We focus on aspects of time complexity that are unaffected by polynomial differences in running time

Our aim is to present fundamental properties of computation rather than properties of Turing machines.

Def 7.12

P is the class of languages that are decidable in polynomial time. That is,

$$\mathbf{P} = \bigcup_k \text{TIME}(n^k)$$

Notes

1. P is invariant for all models of computation that are polynomially equivalent to the deterministic single tape TM
2. P ~ class of problems that are realistically solvable on a computer.

Encodings of problems (denoted $\langle \dots \rangle$)

avoid unary 10 ~ 1111111111
as this is exponentially larger than any coding such as base k for any $k \geq 2$. E.g. 1000 uses only 10 bits in base 2.

Codings of graphs

1. list of vertices and edges + possible costs in binary
2. adjacency matrix: $V \times V$ matrix with entry $(i,j) = \begin{cases} 1 & \text{if edge from } i \text{ to } j \\ 0 & \text{otherwise} \end{cases}$

   possibly with cost $c(i,j)$ in stead of 1
   (in binary)

Thm 5.30

None of EQ_{TM} and $\overline{EQ_{TM}}$ are recognizable.

P: recall that if A is not recognizable and $A \leq_m B$ then B is not recognizable

We show that $A_{TM} \leq_m EQ_{TM}$ and $A_{TM} \leq_m \overline{EQ_{TM}}$

(note that $A \leq_m B \iff \overline{A} \leq_m \overline{B}$)

$A_{TM} \leq \overline{EQ_{TM}}$

Given $\langle M \rangle \langle w \rangle$ construct $\hat{M}_w$ such that $L(\hat{M}_w) = \begin{cases} \emptyset & \text{if } w \notin L(M) \\ \Sigma^* & \text{if } w \in L(M) \end{cases}$

then $\langle M \rangle \langle w \rangle \in A_{TM}$

$\Downarrow$

$\langle \hat{M}_w \rangle \langle M_\emptyset \rangle \in \overline{EQ_{TM}}$

if $\langle M \rangle$ not a TM then $\hat{M}_w = M_\emptyset$

$A_{TM} \leq_m EQ_{TM}$

$\langle M \rangle \langle w \rangle \longrightarrow \langle \hat{M}_w \rangle \langle M_{\Sigma^*} \rangle$

Examples of problems in P:

1. SpT_k from before (earlier slide)

2. $PATH = \{ \langle G, s, t \rangle \mid$ G is a digraph, s, t vertices and $\exists$ (s,t)-path in $G \}$

$PATH \in P$

Deciding membership of context-free L

Given $w \in \Sigma^*$ is $w \in L$?

Here L is given by a chomsky Grammar.

$(S \overset{*}{\Rightarrow} w \iff S \overset{2|w|-1 \text{ steps}}{\Longrightarrow} w$

method 1: try all derivations of length w

$\leq |R|^{2|w|-1}$ not polynomial

Dynamic programming solution!

$w = w_1 w_2 \cdots w_n \quad w_i \in \Sigma$

construct $n \times n$ matrix T where (in the end)

$T_{i,j}$ = set of variables that generate $w_i w_{i+1} \cdots w_j$

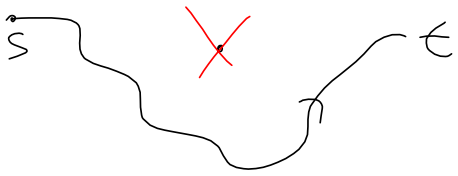
First $T_{i,i} = \{ A \mid A \Rightarrow w_i \} \quad i = 1, 2 \cdots n$

idea: If $A \to BC$ and $B \Rightarrow w_i \cdots w_j$ $C \Rightarrow w_{j+1} \cdots w_p$
then $A \Rightarrow w_i \cdots w_p$ so add A to $T_{i,p}$

Solution in step r $(r \in \{2, \cdots n-1\})$

$\forall A \to BC \in R$: if $i+r \le n$ and $\exists k \in \{i, \cdots i+r\}$ s.t $B \in T_{i,k}$, $C \in T_{k+1, i+r}$
then add A to $T_{i, i+r}$

$HAMPATH = \{(G,s,t) \mid$ G is a digraph, s,t are vertices of G, and $\exists$ (s,t)-path containing all vertices$\}$

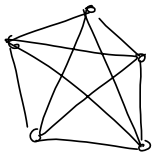


"Difficult" but easy if we can guess.

Given $s = v_1, v_2 \cdots v_{n-1}, v_n = t$

easy to check whether $v_i v_{i+1}$ is an arc for $i = 1,2 \cdots n-1$

$CLIQUE = \{(G,k) \mid$ G is a graph that has a complete subgraph of size $k\}$



Given $v_{i_1} v_{i_2} \cdots v_{i_k}$ we just need to check that $v_{i_p} v_{i_q} \in E(G)$ $\forall p \neq q$

called a 'certificate' for $(G,k) \in CLIQUE$

'proof'

remark if we just have G, k then (essentially) no better method is known than trying all n^k possible subsets.

For HAMPATH: could try all $(n-2)!$ permutations of $V - \{s,t\}$

Def 7.18

A verifier for a language A is an algorithm $\mathcal{A}$ s.t $A = \{w \mid \exists c \text{ (string) s.t } \mathcal{A} \text{ accepts } \langle w, c \rangle\}$

$\mathcal{A}$'s running time is measured in terms of $n = |w|$

$\mathcal{A}$ is a pol. verifier if it has running time $O(|w|^k)$

The string c is the certificate for $w \in A$

$|c(w)| \leq$ running time of $\mathcal{A}$

Def 7.19 $NP = \{ L \mid L \text{ has a pol time verifier} \}$

NDTM for HAMPATH

1. Guess $v_1, v_2, \ldots, v_n$
2. check for repetitions ($v_i = v_j$) and 'reject' if one is found
3. check $s = v_1, t = v_n$ if not 'reject'
4. check whether $v_i v_{i+1}$ is an arc $i = 1, 2, \ldots, n-1$ if so 'accept' otherwise 'reject'

Thm 7.20

$L \in NP \iff L$ is decided by some NDTM

Suppose $A \in NP$ and let $\mathcal{A}_A$ be a verifier for A s.t $\mathcal{A}_A$ runs in time $O(n^k)$

NDTM: on input w, $|w|=n$

1. nondet select a string c s.t $|c| \leq n^k$
2. Run $\mathcal{A}_A$ on $\langle w, c \rangle$
3. accept if $\mathcal{A}_A$ accepts else reject.

Conversely suppose A is decided by some NDTM N, construct $\mathcal{A}$ as follows

$\mathcal{A}$: on input $\langle w, c \rangle$

1. Simulate N on w using c to guide which branch to take (as in proof of Thm 3.16)
2. If this branch of N's computation results in accept, then $\mathcal{A}$ accepts $\langle w, c \rangle$ otherwise reject

$NTIME(t(n)) = \{ L \mid L$ is decided by an $O(t(n))$-time NDTM $\}$

Cor 7.22 $NP = \bigcup_k NTIME(n^k)$

Summarizing

$P = \{ L \mid L$ can be decided fast $\}$

$NP = \{ L \mid L$ can be verified fast $\}$

Open $P = NP$? solve and make 10^6 $

Know $NP \subseteq EXPTIME = \bigcup_k TIME(2^{n^k})$