**On Complexities of Models**

Complexities of algorithms under different models (RAM, TM, ...) differ somewhat, but "not too much"

Crucial distinction: What can be solved/computed in polynomial time vs. what requires super-polynomial time

At this level, Turing Machine and RAM are equally powerful models

---

**Refined Church-Turing Thesis**

Turing-solvability/computability in polynomial time $\equiv$ Algorithmic solvability/computability in polynomial time

Shown by simulations between models

![Simulation Diagram]

**Theorem** Let $M$ be a deterministic Turing Machine with time complexity $T(n)$. Then $M$ can be simulated by a RAM $M'$ with $T_{\text{unit}} = O(T(n))$ and $T_{\log} = O(T(n) \log T(n))$

**Proof** (sketch)

Let TM $M$ have $k$ tapes

Construct RAM $M'$ that simulates it

Store TM tape cells in RAM memory cells

How to represent contents of $k$ (unlimited) TM tapes in RAM memory?

Use $R_i$, $i = 1, \ldots, k$ for the address where the symbol at tape head $i$ is stored

Auxiliary variables: $R_{k+1}, \ldots, R_c$

Simulate tape cells with remaining registers:

Cells at distance $j$ from the start position are simulated by a block of $2k$ registers:

The cell at position $j$ (right) on tape $i$ is simulated by register $c + 2(jk + i)$, and cell $j$ (left) by register $c + 2(jk + i) + 1$
∼ Active cell of tape $i$ can be read (written) with instruction `load *i (store *i)`

Shift of tape head $i$ is simulated with incrementing/decrementing $R_i$ by $-2^k$
(or by $-1$, to pass the start position of the tape head)

Then it is straightforward to simulate transitions of TM with RAM instructions

Unit-cost time complexity of RAM $M'$ increases by a constant factor only wrt $T(n)$

Time complexity $T(n)$ implies space complexity $O(T(n))$ for TM $M$

∼ values of tape head addresses are $O(T(n))$

∼ $T_{log}$ of RAM $M'$ is $O(T(n) \ log \ T(n))$

\[\square\]

**Theorem** Let $M$ be a RAM whose logarithmic-cost time complexity is $T(n)$. Then it can be simulated with a TM $M'$ whose time complexity is $O((T(n))^3)$

**Proof** (sketch)

Let's first skip simulation of `mul` and `div` instructions

Tapes of TM $M'$:
1: input
2: output
3: auxiliary
4: simulation of RAM memory

Contents of tape 4, to simulate RAM with registers $i_1, ..., i_n$:

\[ \text{b # # ... # #} \]

(“Register addresses and contents in binary")

**TM simulation of `load *13` ($AC \gets R[R_{13}]$):

1. Write addr to auxiliary tape

2. Search from memory tape

3. Copy $< 13$ to aux tape

4. Repeat the search

5. Copy $<< 13 >>$ to aux tape

Other instructions simulated similarly
RAM $M$ with logarithmic-cost time complexity $T(n)$ can perform at most that many instructions, and manipulate at most that many bits

$\leadsto$ the "memory tape" fits in $O(T(n))$ cells

Each instruction can be simulated by scanning the memory tape a few times

$\leadsto$ TM $M'$ works in time $O(T(n)^2)$

TM can perform $\text{mul}$ and $\text{div}$ instructions in $O(l^2)$ wrt the length of their operands

$\leadsto$ Total time for simulation is $O(T(n)^3)$

- \hfill \Box

Observe that many combinations of polynomial computations are polynomial:

**Lemma** Let $p(n) = a_0 + a_1 n + \cdots + a_k$ and $q(n) = b_0 + b_1 n + \cdots + b_l$ be polynomials of degree $k$ and $l$, respectively ($a_k, b_l > 0$). Then

- (a) $p(n) + q(n)$ is a polynomial of degree $\max(k, l)$,
- (b) $p(n)q(n)$ is a polynomial of degree $k + l$, and
- (c) $p(q(n))$ is a polynomial of degree $kl$

For example, these take polynomial time:

- (a) polynomial-time executions in a row,
- (b) repeating polynomial-time computation polynomially many times;
- (c) applying polynomial-time algorithms to input that is polynomially larger ($q(n)$) than the original ($n$)

$\leadsto$ polynomial-time TM computation can be simulated with RAM in polynomial time, and vice versa

$\leadsto$ polynomial time RAM (or pseudocode) sufficient as specs for polynomial-time TMs