Universal Turing Machine (UTM)

\[ w \rightarrow M \rightarrow \text{Acc/Rej} \]

(particular TM)

Need to build a new TM for each new algorithm?
No!
One TM can do it all!

Set of Universal TM U:

\[ M \rightarrow W \rightarrow U \]

accept if \( M \) accepts \( W \)
reject if \( M \) rejects \( W \)
\( \text{run forever if } M \text{ on } W \text{ runs forever} \)

* \( U \) is a fixed TM (with some fixed alphabet, ...
But, \( U \) needs to simulate any given TM \( M \) (with any alphabet, \# tapes, etc.)

**Thm [Universal TM]:** There is a 2-tape UTM \( U \) that can simulate any given TM \( M \) on any given input \( w \).

Moreover, if \( M \) takes time \( t \), then \( U \) takes time \( O(\text{log} t) \).

**Attempt 1:** \[ M, w \]

\( M \) has \( K \) tapes

\[ U \]

\( \text{(K tracks)} \)

\( \times \) One step of \( M \) takes \( O(t) \) steps of \( U \)!
Attempt 2:

To update the tape after 1 step of M, need to shift each track. Takes time $O(t)$.

Keep the tracks "in sync".

Attempt 3:

Idea: Don't shift the entire track at a time, but just a portion!

1, 1, R, R
new symbol: \( b \) (new blank)

Set:
- \( R_i \) is full if it contains no \( b \)'s
- \( R_i \) is half-full if it contains \( 2^i \) \( b \)'s
- \( R_i \) is empty if it contains only \( b \)'s

Ex:

\[ R_i \]

\[
\begin{array}{ccccccc}
\text{empty} \\
\hline
b & b & b & b & b & b & b \\
\hline
\end{array}
\]
Invariant:

1. Position 0 contains non-0's in each track (symbols currently scanned by $M$)

2. For $0 \leq i \leq \log t$,
   - either both $L_i$ & $R_i$ half-full
   - or one full & the other empty.

Performing a shift (right to left)
Claim: After a shift of index \( i_0 \), we don't perform any shift of index \( i_0 \) (or greater) for at least \( 2^{i_0} \) shifts.

Time +
index \( i_0 \), shift + times
This shift takes time $O(2^i)$. The total time to simulate $t$ steps of $M$ on $w$ is

\[
\sum_{i=0}^{\log t} \frac{t}{2^i} \cdot O(2^i) = O(1) \cdot t \cdot \sum_{i=0}^{\log t} \frac{1}{2^i} = O(t \cdot \log t).
\]
k-tape TM \ t^2
\rightarrow
k\text{-tape TM } t^2
2
O(H \cdot \log t)

Linear Speedup Thm

Ignore any constant factors when talking TM's run times.

10 \cdot n \ vs. \ 100 \cdot n

\sum \in \Sigma^*
q_1, q_2, q_3, q_4, \ldots, q_k

k
\[ \sum_l = \sum_k \]

With \( \geq 6 \) moves, the new TM can simulate \( K \) moves of the original TM.

\[ \Rightarrow t \rightarrow t \frac{K}{k} = 3 + \sum a \rightarrow \text{const} \]

---

**Reductions**: less hard than

\[ L_1 \leq L_2 \]

polytime reducible

there is some efficient \( f: \Sigma^* \rightarrow \Sigma^* \)

\[ \forall x, x \in L_1 \Leftrightarrow f(x) \in L_2 \]
Example

3 SAT = \{ \psi \mid \psi \text{ is a satisfiable } 3 \text{- CNF} \}

IS = \{ \phi (y, k) \mid \text{\phi is a graph with independent set of size } \geq k \}

3 SAT \leq IS

m = \# clauses

\psi = (x \lor y \lor w \lor \overline{z}) \land (\overline{x} \lor u \lor v \lor z) \land \ldots

G_{\psi} \vdash x \lor \overline{x} \lor z \

k = m

Completeness

C complexity class
$\mathsf{L}$ is $C$-complete

- $L \in C$
- $\forall L' \in C, L \leq L'$

$C = \mathsf{NP}, \leq = \mathsf{polytime}$

$C = \mathsf{P}, \leq = \mathsf{logspace}$

$C = \mathsf{EXP}, \leq = \mathsf{polytime}$

Extended Church-Turing Thesis

Efficient $\equiv \mathsf{P}$