Universal Turing Machine (UTM)

Need to build a new TM for each new algorithm?

No!

One TM can do it all!

Set of Universal TM U:

- U is a fixed TM (with some fixed alphabet, # tapes, etc.)
- 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 $O(t \cdot \log t)$.

**Attempt 1**: $M$, $w$

$M$ has $K$ tapes

$U$ has $K$ tracks

One step of $M$ takes $O(t)$ steps

$U$!

**Attempt 2**: +
Attempt 2:

\[ \overbrace{U}^{\text{tracks}} \quad t \]

Keep the tracks "in sync".

To update the tape after 1 step of \( M \), need to \underline{shift} each track.

takes time \( O(t) \) \( \times \)

Attempt 3:

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

\[ \overbrace{L_1 \quad L_0 \quad R_0 \quad R_1}^{1 \text{ track}} \]

\[ \begin{array}{cccccccc}
\text{bbbab} & \text{1011011} \\
\end{array} \]

\[ \text{position} \quad \text{log} t \rightarrow \text{log} t \]

\[ |L_i| = |R_i| = 2^{i+1} \text{ cells} \]

New symbol: \( b \) (new blank)
**Def:**

- $R_i$ 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:**

\[
R_i = \overbrace{\underbrace{\ldots | \ldots}}^{2^{i+1} = 4} \quad \text{empty}
\]

\[
\begin{array}{cccc}
  b & b & b & b \\
\end{array} \quad \text{half-full}
\]

\[
\begin{array}{cccc}
  b & b & o & u \\
\end{array} \quad \text{full}
\]

**Invariant:**

1. position 0 contains non-$b$'s in each track (symbols currently scanned by $M$)
2. If $0 \leq i \leq \log t - 1$, either both $L_i$ & $R_i$ half-full, or one full & the other empty.

Performing a shift (right to left):

![Diagram of a shift](image)

Shift left of index $i_0$: 

![Diagram of a shift left](image)

Claim: After a shift of index $i_0$, we don't perform any shift of index $i_0$. 

Special case where $\# \text{ rooms} = 1 + \sum_{j=0}^{i_0} \frac{i_0 - j}{2^j}$

Where $i_0 = 2^i - 1$ and $j = 0, 1, \ldots, i_0$.
\[ i_0 \text{ (or greater) for at least } 2^{i_0} \text{ shifts.} \]

Time \( t \)

\[ \text{index } i \text{ shift at most } \frac{t}{2^{i_0}} \text{ times} \]

This shift takes time \( O(2^{i_0}) \)

The total time to simulate \( t \) steps of \( M \) on \( w \)

\[ \sum_{i=0}^{\log t} \frac{t}{2^i} \cdot O(2^i) \]

\[ = O(1) \cdot t \cdot \sum_{i=0}^{\log \frac{t}{2}} \frac{1}{2^i} \]
= \mathcal{O}(t \cdot \log t).

\begin{align*}
\text{k-tape TM} & \quad t \\
\Rightarrow & \\
\text{k-tape TM} & \quad t^2 \\
\text{2} & \quad \mathcal{O}(t \cdot \log t)
\end{align*}

\underline{Linear Speedup Thm}

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

// 10 \cdot n \text{ vs. } 100 \cdot n

\text{vs. } n

\exists \delta \in \mathbb{R}^+ \quad 0 < \delta
\[ \Sigma = \Sigma \]

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

\[ t \rightarrow \frac{t}{k} = \epsilon \cdot t + \beta \rightarrow 0 \text{ as } k \to \infty \]