CS 1010 Notes

lecture 14: Time Complexity

Example language: \( L = \{ w, 13, 27 \mid w \text{ has a substring of length } 13 \text{ that occurs } \underbrace{27 \text{ times}}_{\text{without overlap}} \} \)

Language and 437 in \( L \)

Brute Algorithm: For every possible start position of a length 13 substring (there are \( 13 \cdot 13 + 1 \) positions), count how many times the length 27 substring that starts at \( p \) occurs. If count \( \geq 13 \), accept.

TM version: Set aside enough space on the tape to count.
Mark a position (position one at the input) and its neighbors (to denote a candidate substring), takes \( 13 \cdot n \) steps. (\( 13 \cdot n \) steps).
For every subsequent position, see if it matches the substring. Also \( 13 \cdot n \) steps for each position, so \( 13 \cdot n^2 \) steps total.
Repeat \( n \) times.

Total \( \Theta(n^3) \) steps, or \( O(n^4) \), since \( 13 < n \)

Def: Let \( M \) be a decider. The runtime of \( M \) is \( t : \mathbb{N} \rightarrow \mathbb{N} \) s.t. \( M \) takes at most \( t(n) \) steps to accept or reject on an input of length \( n \).

We typically only care about the asymptotics of the runtime.

Big-O notation: \( f(n) = O(g(n)) \) if \( \exists C, \alpha \) s.t. \( f(n) \leq C g(n) \).

\( C \) is a boot \( \mathbb{N} \rightarrow [0, \infty) \).
Ex. alg's runtime is $1,000,000 + 527n^2 + 7n^3$

- cost to initialize
- populate
- running
- data structure
- data structure
- alg.

Run time is $O(n^3)$.

**Def:** $f(n) = \Theta(g(n))$ means $c_1 g(n) \leq f(n) \leq c_2 g(n)$, $c_1, c_2$ constants or equivalently $f(n) = O(g(n))$ and $g(n) = O(f(n))$.

**Def:** If $f(n) = o(g(n))$, $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$.

**Def:** Let $\text{TIME} : \mathbb{N} \to \mathbb{R}^+$. $\text{TIME} (f(n))$ is given by

$$\text{TIME} (f(n)) = \{ L | L = L(C(n)) \text{ for some TM } M \text{ whose runtime is } O_T (n^{f(n)}) \}$$

$L_{\text{rep}} \in \text{TIME}(n^n)$

**Observation:** If $f(n) = O(g(n))$, $\text{TIME} (f(n)) \subseteq \text{TIME} (g(n))$.

What if we use a multi-tape TM for $L_{\text{rep}}$?

```
# Multi-tape TM

- Input tape
- Substring tape
- Counter

Copy 1 symbol from w to substring tape.
Initialize counter to 1.
Advance head and see if strings match.
If so, increment counter.
```
O(n) candidate substring. For each, we do n passes through input, taking k steps for each pass.
Runtime: O(n^2 k).

This doesn't tell us anything about runtime of single-tape, so we can't say \text{LONGER \ TIME(n^3)}.

Define \text{MTIME}(t(n)) = \{ L | L = \text{LCM} \text{ for some \ \text{multi-tape}} \ M \text{ with \ runtime \ } t(n) \}.

Then \text{Lrep} \in \text{MTIME}(t(n)).

Since a canonical \text{1-tape \ TM} is a special case of a \text{multi-tape \ TM}, if \text{L \in MTIME}(t(n)) then \text{L \in MTIME}(t(n)).

Recall: can simulate an \text{m-tape \ TM} with a single-tape \text{TM}.

A config of a \text{m-tape \ TM}
Therefore, if \( L \subseteq \text{TIME}(t(n)) \),
then \( L \subseteq \text{TIME}(t(n)^2) \).

Now let's use an NTM for \( L_0 \).

1. Take a nondeterministic TM \( N \), nondeterministically pick the start position for the substring (or reject).
2. Count how many times the length-\( k \) substring occurs. We said before this takes \( O(n^2/k) \).
   For multi-tape variant, this takes \( O(n/k) \).

Correctness: if \( (w, k, l, L_0) \notin L_0 \), then exists an accepting CTM \( N \) on this input.
Yes: as long as \( N \) picks the start position correctly.
No: if \( (w, k, l, L_0) \notin L_0 \), no matter what \( N \) picks we don't get an accepting CTM, and reject (there are no matches).

Def: An NTM is a dener if it halts on every branch on every input.

The runtime \( t(n) \) of an NTM is the maximum # of steps any branch of computation of \( N \) takes on any input of length \( n \).

\[ \text{NTIME}(t(n)) = \{ L \mid L = L(N) \text{ for some NTM } N \text{ with runtime } t(n) \} \]

\[ \text{TIME}(t(n)) \subseteq \text{NTIME}(t(n)) \]

Theorem: If \( t(n) \geq n \), then \( \text{NTIME}(t(n)) \subseteq \text{TIME}(2^{O(t(n))}) \).

Note: \( 2^{O(n)} = 3^{O(n)} = 4^{O(n)} = \ldots \)

Recall how to simulate on NTM on a DTM: use a multi-tape TM.
\[ W_1, W_2, \ldots, W_n \quad \text{input tape} \]

\[ W_1, W_2, \ldots, W_n \quad \text{work tape} \]

\[ \quad \text{non-det. tape} \]

\( t(n) \) symbols to guide non-det. choices - uses symbols 0, \ldots, n,  
initialize to all 0's.

To make a non-det. choice at step \( k \), look at the non-det. tape's \( k \)th cell to guide you.

If simulation accepts, accept.  
If it rejects, clear work tape, increment non-det. tape, and try again until all choices have been made, then reject.

Thus, we run \( t(n) \) \( N \) at most \( 2^{O(t(n))} \) times,  
while \( t(n) \) \( \leq O(t(n)) + \log(N) \) \( = O(N) \)

\[
t(n) = 2^{O(t(n))} = 2 \]