Lecture 13

- Big picture
- Time Complexity
- The class P
- The class NP

We are transitioning into complexity theory, talking about the complexity needed for a computation. For TMs, we care about time and space complexity.

**Definition:** The running time (also known as runtime or time complexity) of a TM M is a function $f: \mathbb{N} \rightarrow \mathbb{N}$ such that the maximum number of steps M takes on any input $x$ of length $n$.

We will concern ourselves with "big-Oh" notation:

$f(n) = O(g(n))$ if $\exists C, n_0$ s.t. $\forall n \geq n_0$, $f(n) \leq Cg(n)$

Recall the language $L = \{a^n b^n c^n | n \geq 0\}$ for which we designed a TM M earlier. This M has runtime $O(n^3)$, as for each $a$, it has to scan over the tape once. The tape is $O(n)$ long, so we also $O(n)$ work total.

Let us construct a multitape TM for $L$:

3 tape

1. Copy all the $a$'s onto tape 2
2. Copy all the $b$'s onto tape 3
3. Scan all three tapes, matching all $a$'s, $b$'s, $c$'s

The runtime of this is $O(n)$, as our tapeheads only move in one direction.

**Note:** The optimal runtime of recognition of this language $L$ is $O(n)$ as we would not be able to read the whole string in less time, and thus cannot know if the string is in the language.
Q: Can you do better than $O(n^2)$ by simulating the MTM on a canonical TM?
A: No (we will come back to this!)

Q: Can you decide $L$ on a TM w/ runtime better than $O(n^3)$?
A: Yes! We can do $O(n \log n \cdot \log(n \log(n)))$, where $\log$ is a polynomial in $\log$.  This can be done by somehow carrying a counter, and then directly counting the a's, b's and c's.

Consider the language $L_w = \{ w \mid w \text{ is a palindrome} \}$. We can decide it in $O(n)$ time, where $|w| = n$ on a TM with two tapes, by copying the string $w$ over on the second tape, and reading it simultaneously forwards and backwards. However, we cannot do better than $O(n^3)$ on a canonical TM.

Let us return to the question of going from an MTM to a canonical TM in general.  We simulate a MTM with some number of tapes as follows:

\[
\begin{align*}
\text{contents of tape 1} & \quad \text{contents of tape 2} & \cdots \\
& \quad & \\
& & \\
& & \\
\end{align*}
\]

and the arrows are the heads of the MTM.

Each move of this 1-tape TM takes $O(S^2)$ steps, where $S$ is the amount of space used by MTM, with $O(t(n))$.

Thm: For an MTM $M$ w/ runtime $t(n) \geq n$, if a regular TM $M'$ s.t. $L(M) = L(M')$ and the runtime of $M'$ is $O(t^2(n))$.

The proof can be done by preparing the entire simulation of an MTM by a reg. TM.
Thm: For any $t(n)$-time $N$ TM $N$ st $L(N)=L(M)$ and $M$'s runtime is $2^{O(t(n))}$.

Proof Idea: Recall how to simulate $N$ on a deterministic $M$. To simulate $i$ steps of computation of $N$, need to explore $c^i$ possible non-det choices that $N$ can make.

Definition: The runtime of an NTM $N$ is $f: \mathbb{N} \to \mathbb{N}$ st $\forall n$, $\forall x$ of length $n$, a computation branch of $N$ on input $x$, $N$ halts within $f(n)$ steps.

$\text{TIME}(t(n)) = \{ L | \exists \text{ a TM } M \text{ that runs in } t(n) \text{ time st } L(M) = L \}$

for example, $\{a^n b^n c^n \}$ $\in$ TIME ($n$ polylog $n$), and Palindrome $\in$ TIME ($n^2$)

Note that $\text{TIME}(t(n))$ is a set of languages!

$\text{NTIME}(t(n)) = \{ L | \exists \text{ an NTM } N \text{ that runs in } O(t(n)) \text{ time st } L(N) = L \}$

since a TM is trivially a NTM, if we get the following relationship:

$\text{TIME}(t(n)) \subseteq \text{NTIME}(t(n)) \subseteq \text{TIME}(2^{O(t(n))})$

We are now ready to define P, NP:

$P = \bigcup_{k=1}^{\infty} \text{TIME}(n^k)$

$NP = \bigcup_{k=1}^{\infty} \text{NTIME}(n^k)$

with the following relationship:

$P \subseteq NP \subseteq \text{TIME}(2^{O(n^{100})})$.

while the last class is actually called EXP.

The class P captures all languages that can be solved in polynomial time.

However, some polynomial runtimes are really slow! eg $O(n^{100})$

The reason why we still care so much about P is because we care more about things being really hard (eg not in P) than things that are doable but somewhat inefficient.
We see that if a language is polynomial in time on a multitape TM, we know that it is in \( P \) because of the first TM in this lecture.

Here comes the biggest question of them all:

\[ P = \text{NP?} \quad \text{we don't know.} \]

We conjecture that \( P \neq \text{NP} \) (proper subset), but it is not proven. We do know, however, that there are some languages \( \in \text{NP} \) that are very hard. These are called \( \text{NP-complete languages} \). If we can find a polytime deterministic solver for an \( \text{NP-complete language} \), we have solved \( P = \text{NP} \).

\[ \text{PATH} = \{ (G, u, v) \mid G \text{ is an undirected graph, } u, v \in V(G) \text{ and there is a path from } u \text{ to } v \} \]

\[ \text{PATH} \in \text{P}. \]

\[ \text{HAMPATH} = \{ (G, u, v) \mid G \text{ is an undirected graph, } u, v \in V(G) \text{ and there is a path from } u \text{ to } v \text{ for each vertex of } G \text{ only once} \} \]

\[ \text{HAMPATH} \in \text{NP}. \]

To see that \( \text{PATH} \in \text{P} \), we need to:

1. Give a TM \( M \) that decides \( \text{PATH} \)
2. Analyze its correctness
3. Analyze its runtime
1. Idea: make all neighbours of things we have already seen (marked),
continue until you have marked every thing possible. Then check if
V is marked.

"On input (G, u, v), do:
mark u
while there are unmarked neighbours of the marked vertices
mark all the unmarked neighbors
if V is marked, accept
else reject v

2. Suppose (G, u, v) \in \text{PATH}. Then v is eventually marked, so M accepts.
Suppose (G, u, v) \notin \text{PATH}. Then v will never be marked, so M rejects.

3. Assume the graph G is given as a list of indexes (in order) of vertices,
a list of tuples representing edges,
we mark vertices by checking, for every already marked vertex,
the edges to see what vertices connect, and mark them. Hence
we if we have n vertices and m edges, we are doing O(nm)
work, which is only nominal in the input (G, u, v).

It follows that since \text{PATH} has a (correct) polytime decider, \text{PATH} \in P.
Similarly, for seeing that \text{HAMPATH} \in \text{NP}, we need to:
1. Give an NTM M that decides HAMPATH.
2. Analyze its correctness
3. Analyze its runtime.
1. No: "on input $(G, u, v)$, 
   Non-deterministically pick an order $V_i, \ldots, V_n$ 
   in which to visit every vertex starting at $u$, ending in $v$. 
   Verify that this is a path in $G$, i.e. $V_i \in S$, $(V_i, V_{i+1}) \in E(G)$ 
   If successfully verified, accept. 
   Else reject.

2. Suppose $(G, u, v) \in \text{HAMPATH}$. Then $\exists V_i, \ldots, V_n$ that is a ham. path from $u$ 
   to $v$. Then, on some branch of its computation, $N$ will pick this ordering, and 
   accept. 
   Suppose $(G, u, v) \notin \text{HAMPATH}$ 
   Suppose $N$ accepts: Then $\exists V_i, \ldots, V_n$ that is verified, and that was a 
   ham. path.

3. Runtime: Generating an ordering takes some polynomial time because we pick an 
   vertex, check that it has not been picked, add it to the sequence etc. 
   Verify by $O(V|E|)$. Hence $N$ is non-deterministic poly time. 

$\implies \text{HAMPATH} \in \text{NP}$.