Turing Machines and DTIME

Exposition by William Gasarch—U of MD

Turing Machines Definition

Definition

A *Turing Machine* is a tuple $(Q, \Sigma, \delta, s, h)$ where

- Q is a finite set of states. It has the state h.
- Σ is a finite alphabet. It contains the symbol $\#$.
- $\delta : (Q - \{h\}) \times \Sigma \to Q \times \Sigma \cup \{R, L\}$
- $s \in Q$ is the start state, h is the halt state.

Note There are many variants of Turing Machines- more tapes, more heads. All equivalent.

Turing Machines Conventions

We use the following convention:

1. On input $x \in \Sigma^*$, $x = x_1 \cdots x_n$, the machine starts with tape

$$\#x_1x_2 \cdots x_n\#\#\#\# \cdots$$

   that is one way infinite.
2. The head is initially looking at the x_n.
3. $\delta(q, \sigma) = (p, \tau)$: state changes $q \to p$, σ is replaced with τ.
4. $\delta(q, \sigma) = (p, L)$: state changes $q \to p$, head moves Left. ($\delta(q, \sigma) = (p, R)$ similar).
5. TM is in state h: DONE. Left most square has a 1 (0) then M ACCEPTS (REJECTS) x.

Note We can code TMs into numbers. We say Run $M_x(y)$ which means run the TM coded by x on input y.

How Powerful are Turing Machines?

1. There is a JAVA program for function f iff there is a TM that computes f.
2. Everything computable can be done by a TM.

Other Models of Computation

There are many different models of Computation.

1. Turing Machines and variants.
2. Lambda-Calculus
3. Generalized Grammars
4. Others

Other Models of Computation

There are many different models of Computation.

1. Turing Machines and variants.
2. Lambda-Calculus
3. Generalized Grammars
4. Others

They ended up all being equivalent.

Other Models of Computation

There are many different models of Computation.

1. Turing Machines and variants.
2. Lambda-Calculus
3. Generalized Grammars
4. Others

They ended up all being equivalent.

This is what makes computability theory work! We will almost never look at the details of a Turing Machine. To show a set of function is TM-computable we just write psuedocode. DO NOT write a TM.

Decidable Sets

Definition

A set A is DECIDABLE if there is a Turing Machine M such that

$$x \in A \rightarrow M(x) = Y$$

$$x \notin A \rightarrow M(x) = N$$

Time Classes

Definition

Let $T(n)$ be a computable function (think increasing). A is in $\mathrm{DTIME}(T(n))$ if there is a TM M that decides A and also, for all x, $M(x)$ halts in time $\leq O(T(|x|))$.

Time Classes

Definition

Let $T(n)$ be a computable function (think increasing). A is in $\mathrm{DTIME}(T(n))$ if there is a TM M that decides A and also, for all x, $M(x)$ halts in time $\le O(T(|x|))$.

What do you think of this definition? Discuss.

Time Classes

Definition

Let $T(n)$ be a computable function (think increasing). A is in $\mathrm{DTIME}(T(n))$ if there is a TM M that decides A and also, for all x, $M(x)$ halts in time $\leq O(T(|x|))$.

What do you think of this definition? Discuss.

Its Terrible!

Time Classes

Definition

Let $T(n)$ be a computable function (think increasing). A is in $\mathrm{DTIME}(T(n))$ if there is a TM M that decides A and also, for all x, $M(x)$ halts in time $\leq O(T(|x|))$.

What do you think of this definition? Discuss.

Its Terrible!

The definition depends on the details of the type of Turing Machine. 1-tape? 2-tapes? This should not be what we care about.

Time Classes

Definition

Let $T(n)$ be a computable function (think increasing). A is in $\mathrm{DTIME}(T(n))$ if there is a TM M that decides A and also, for all x, $M(x)$ halts in time $\leq O(T(|x|))$.

What do you think of this definition? Discuss.

Its Terrible!

The definition depends on the details of the type of Turing Machine. 1-tape? 2-tapes? This should not be what we care about.

So what to do?

- Prove theorems about $\mathrm{DTIME}(T(n))$ where the model does not matter. (Time hierarchy theorem)).
- Define time classes that are model-independent (P, NP stuff)

Time Hierarchy Theorem

Theorem (The Time Hierarchy Theorem) For all computable increasing $T(n)$ there exists a decidable set A such that $A \notin \mathrm{DTIME}(T(n))$.

Proof Let $M_1, M_2, \ldots,$ represent all of $\mathrm{DTIME}(T(n))$ (obtain by listing out all Turing Machines and putting a time bound on them). Here is our algorithm for A. It will be a subset of 0^*.

1. Input 0^i.
2. Run $M_i(0^i)$. If the results is 1 then output 0. If the results is 0 then output 1.

For all i, M_i and A DIFFER on 0^i. Hence A is not decided by any M_i. So $A \notin \mathrm{DTIME}(T(n))$.

End of Proof

Full Time Hierarchy Theorem (I don't care!)

The Time Hierarchy Theorem is usually stated as follows:
Theorem (The Time Hierarchy Theorem) For all computable increasing $T(n)$ there exists a decidable set A such that $A \in \mathrm{DTIME}(T(n)\log(T(n)) - \notin \mathrm{DTIME}(T(n))$.

Full Time Hierarchy Theorem (I don't care!)

The Time Hierarchy Theorem is usually stated as follows:

Theorem (The Time Hierarchy Theorem) For all computable increasing $T(n)$ there exists a decidable set A such that $A \in \mathrm{DTIME}(T(n)\log(T(n)) - \notin \mathrm{DTIME}(T(n))$.

The proof I did of our Time Hierarchy Theorem can be done more carefully and you will see that $A \in \mathrm{DTIME}(T(n)\log(T(n))$. But this involves specifying the model more carefully.

I DO NOT CARE!

Full Time Hierarchy Theorem (I don't care!)

The Time Hierarchy Theorem is usually stated as follows:

Theorem (The Time Hierarchy Theorem) For all computable increasing $T(n)$ there exists a decidable set A such that $A \in \mathrm{DTIME}(T(n)\log(T(n)) - \notin \mathrm{DTIME}(T(n))$.

The proof I did of our Time Hierarchy Theorem can be done more carefully and you will see that $A \in \mathrm{DTIME}(T(n)\log(T(n))$. But this involves specifying the model more carefully.

I DO NOT CARE!

But good to know so that you know SOME seperations: $\mathrm{P} \subset \mathrm{EXP}$.

P and EXP

Definition

1. $\mathrm{P} = \mathrm{DTIME}(n^{O(1)})$.
2. $\mathrm{EXP} = \mathrm{DTIME}(2^{n^{O(1)}})$.