A_{TM} undecidability proof continued...

Theorem 4.11 A_{TM} is undecidable.

Proof: We prove this theorem by contradiction.

Assume A_{TM} is decidable.

Let TM H is a decider for the A_{TM} language.

$$H(\langle M, w \rangle) = \begin{cases} \text{accept}, & \text{If } M \text{ accepts } w \\ \text{reject}, & \text{If } M \text{ does not accept } w \end{cases}$$

Then I am going to construct TM D, where it will take <K>; K is a TM, as the input.

D = "on <K>, where K is a TM

1. Run H on <K, <K>>
2. output the opposite of what H outputs on <K, <K>>

Now, I am going to feed <D> to D.

$$D(\langle D \rangle) = \begin{cases} \text{accept, if D does not accept } \langle D \rangle \\ \text{reject, if D accepts } \langle D \rangle \end{cases}$$

This is a contradiction.

$\therefore A_{TM}$ is not decidable.

Let's look into this proof bit more carefully.

- This proof implicitly involves diagonalization.
  - H accepts $\langle M, w \rangle$ when M accepts w.
  - D rejects $\langle K \rangle$ when K accepts $\langle K \rangle$
  - D rejects $\langle D \rangle$ when D accepts $\langle D \rangle$

Let us create a table that describes behaviorrs of TM when we input TM as inputs to them.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$ ----------
M_1	accept		accept	
M_2		accept		accept
M_3				
M_4	accept	accept		
⋮				

each entry is accept if the machine accepts the input and blank if it rejects or loop on that input.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$ ----------
M_1	accept		accept	
M_2		accept		accept
M_3				
M_4	accept	accept		
⋮				

Note that entry i,j is the value of H on the input $\langle M_i, \langle M_j \rangle \rangle$

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	.....	D	.. ..
M_1	accept		accept			⋮	
M_2		accept		accept		⋮	
M_3						⋮	
M_4	accept	accept				⋮	
⋮						⋮	
D	reject	reject	accept		. . . .	?	

I cannot put a value here.

This proof looks very hard.

Seems like it's very hard to prove something is undecidable using these type of argument.

* There is a easier way.

Reduction

$$A \propto B \quad (A \leq B)$$

A, B are problems

If $A \propto B$, then we can use an algorithm that solves problem B to solve problem A.

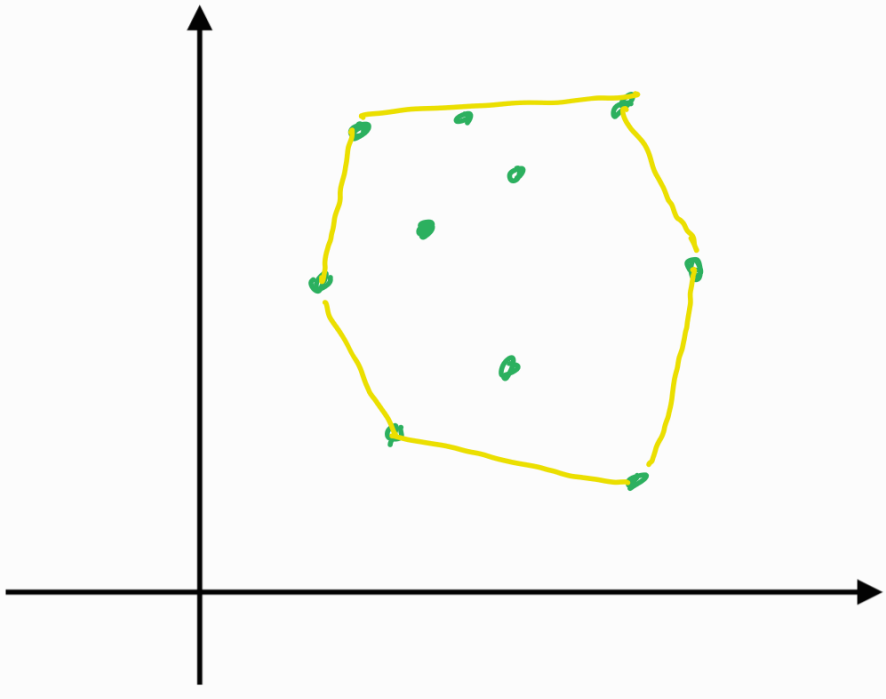
* If you already know problem A is very hard to solve, and If A is reducible to B, then you can use a hypothetical algo that solves B to solve problem A.

Then Problem B must be at least as hard as problem A.

Example of a reduction.

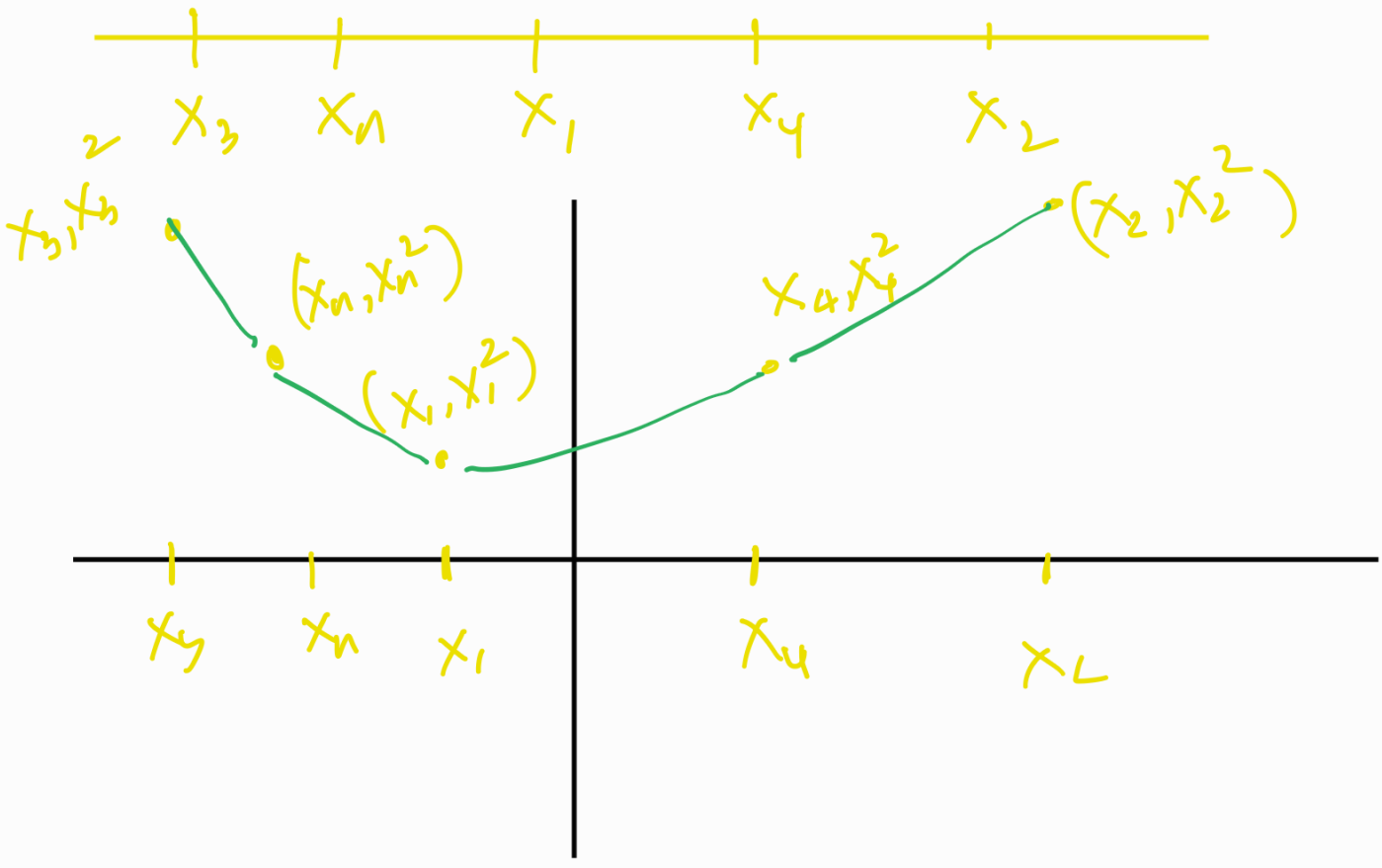
Sorting n real numbers $x_1, x_2, x_3, \ldots, x_n$ $\leq$ 2D convex hull problem.

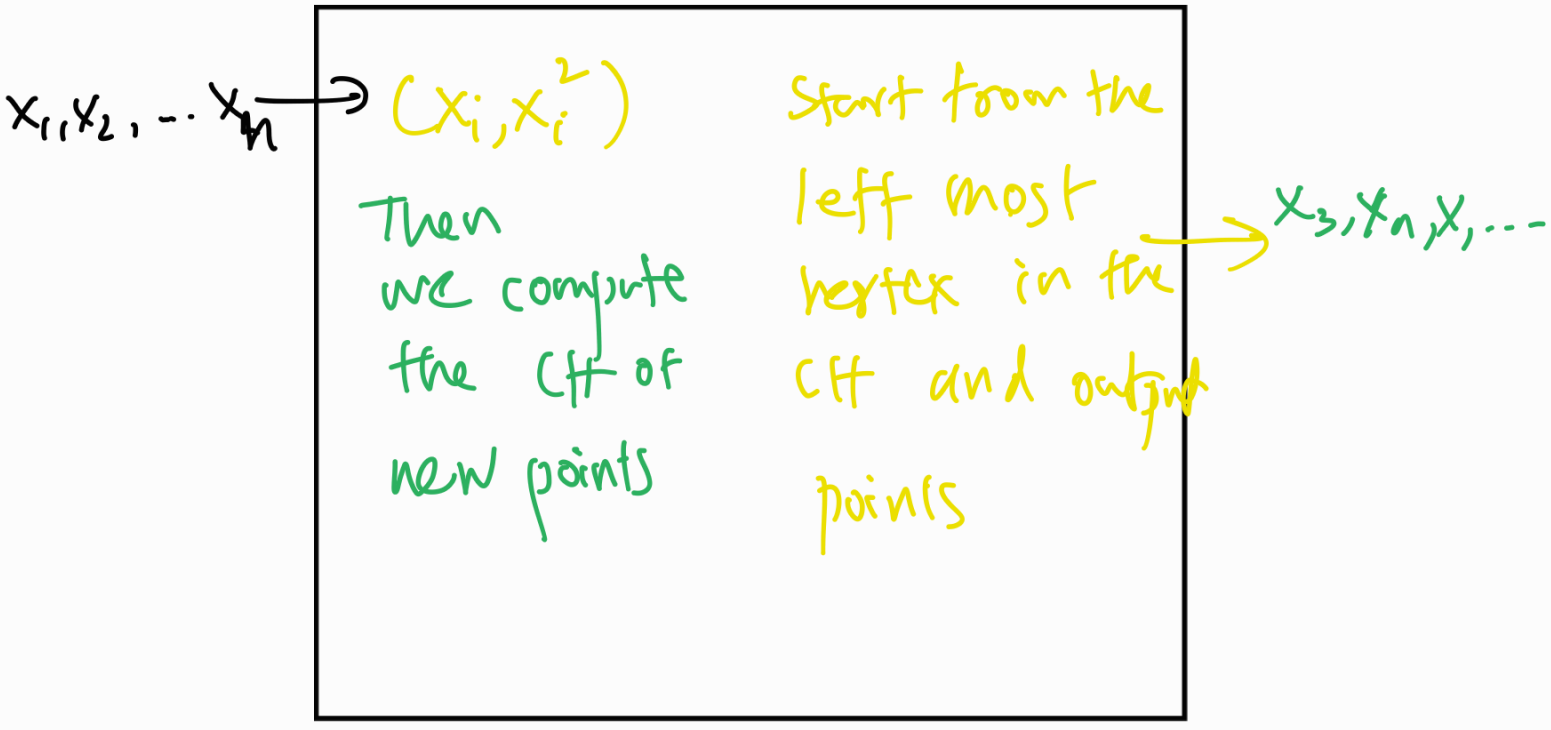
Given set of n points in 2D plane, find the convex hull.



Given $x_1, x_2, x_3, \ldots x_n$, convert these points to set of 2D points, where

$x_i \rightarrow (x_i, x_i^2)$





Sorting is $\Omega(n \log n)$, therefore CH algorithm must have $\Omega(n \log n)$

Reducibility

- Reduction: convert problem A to B s.t the solution to B can be used to solve A.

$(A \leq B, A \propto B)$

How can we use this to show the undecidability?

Right now we proved A_{TM} problem is undecidable.

We are going to prove $HALT_{TM}$ is undecidable.

$$HALT_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on } w \}$$

$HALT_{TM}$ and A_{TM} are two different languages.

We are going to prove $HALT_{TM}$ is undecidable.

We will show that

$$A_{TM} \leq HALT_{TM}$$

(Note that order of the reduction matter)

Assume $HALT_{TM}$ is decidable, let TM R be the decider for $HALT_{TM}$.

A deider for A_{TM}

clearly this is an algorithm that decides A_{TM}. But this cannot happen as A_{TM} is undecidable.

∴ Our initial assumption is incorrect.

$HALT_{TM}$ is undecidable.