

If true, $\boxed{\# | a | \# | \# | 1 | \#}$
 we return it by π_2^3 which is 0 ($m=0$)

If false, we increase m value by 1.

then we get the state

$\boxed{\# | a | \# | 1 | \#}$

Now we copy again, $\boxed{\# | a | \# | 1 | \# | a | \# | 1 | \#}$

true

apply $P(a, 1)$

false

$\boxed{\# | a | \# | 1 | \# | 1 | \#}$

return 1.

$\boxed{\# | a | \# | 1 | 1 | \#}$

Increase m by 1.

→ then copy & repeat the steps.

Finally we will get \lim for which $P(a, m)$ is true.

next thursday - test.

Thursday
20th Feb

Big picture : μ -recursive fn (church)
 ↓
 TM (Turing)

Proving that every μ -rec fn is Turing-computable.

$$\mu m P(\vec{a}, m) = f(\vec{a}) \quad \vec{a} = (a_1, \dots, a_k)$$

We know that, $P(\vec{a}, m)$ is Turing-computable. We want:

$\boxed{\# | \vec{a} | \# | m}$

we start with

produce a TM for computing $f(\vec{a})$

Analyze the problem :-

idea: try $m=0$; if $P(\vec{a}, 0)$ return 0, else try $m=1$

if $P(\vec{a}, 1)$ return 1.

$\therefore m=0$;

while ($\neg P(\vec{a}, m)$)

{ $m++$ }

Thus, copy the state $(\vec{a}, 0)$ once.

Correct way:

step 1:

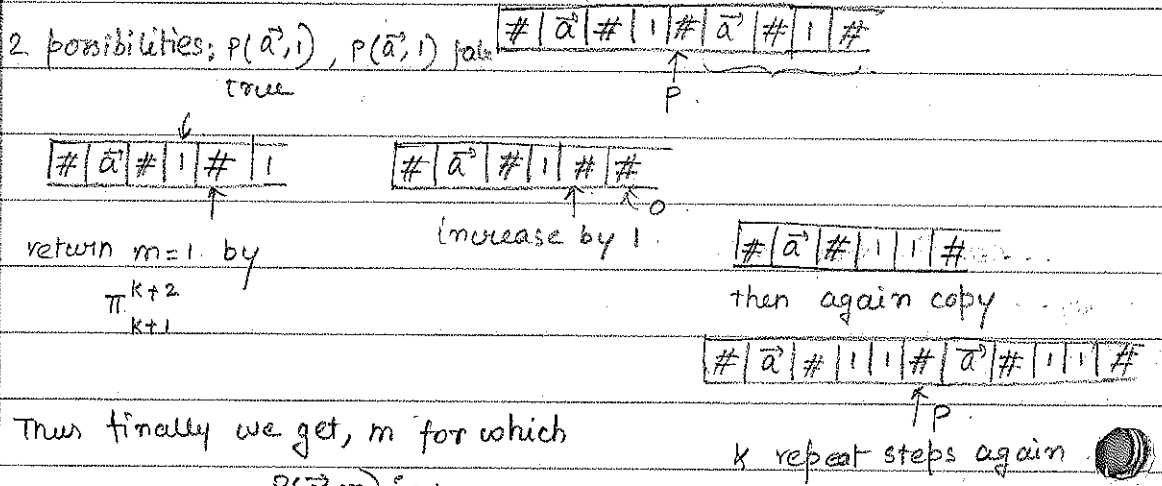
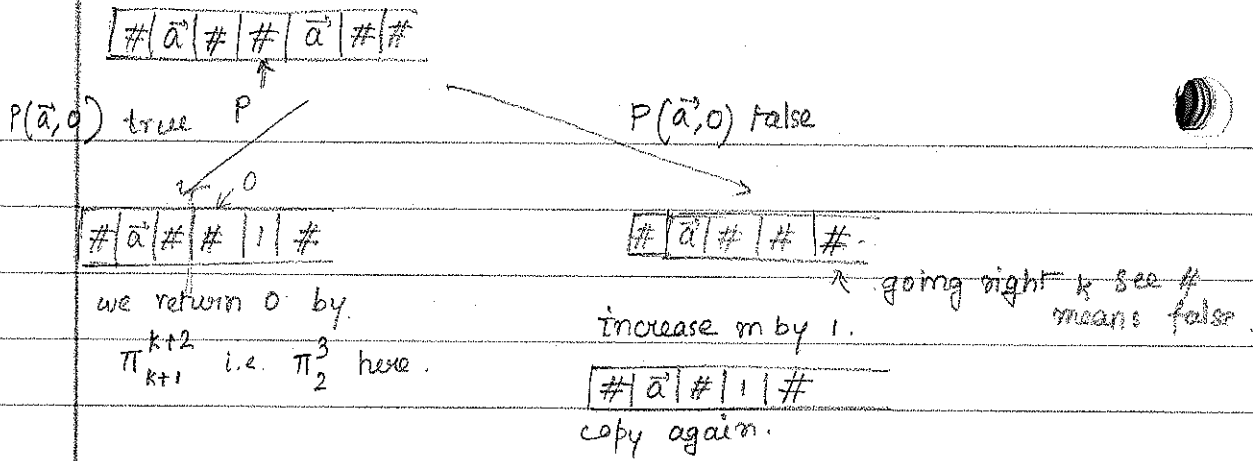
$\boxed{\# | a | \# | \#}$

copy

$\boxed{\# | \vec{a} | \# | \# | \vec{a} | \# | \#}$

If we start with $\boxed{\# | \vec{a} | \# | \#}$ if $P(\vec{a}, 0)$ false

then $\boxed{\# | \# | \#}$ so a is lost.



Thus finally we get, m for which $P(\bar{a}, m)$ is true.

\therefore le-recursion is Turing-computable.

Problem with what we had so far:-

- + : when we had a -ve result it was ok.
- : sometimes we actually produced algorithms, but many of these algorithms were unrealistically long.

Example: $\delta(n) = n+1$

If $n=1$ million, then no of operations on TM would be very huge.

for security, we use integers $\approx 100-200$ digits long. \downarrow impossible # of steps.

we need to separate "theoretical" algorithms from "practical" algorithms.

feasible

How do we define an algo?

1930s / TM
μ-recursion

How we define a feasible algorithm? No perfect definition exists.

Existing definition:

sorting $O(n \log n)$ Exhaustive search 26^n → not feasible.
linear search $O(n)$
matrix multiplication $O(n^3)$

A little bit of physics: $\approx 6 \cdot 10^{17}$ secs.

$T \approx 20$ billion years $\approx 2 \cdot 10^{10}$ years (timeline of universe)

We can make, $\frac{T}{\Delta t}$ → smallest possible time interval.
computational steps

1 year = 365 days $\approx 4 \cdot 10^2$ days $\approx 4 \cdot 10^2 \cdot 2 \cdot 10^1 \cdot 6 \cdot 10^1 \cdot 6 \cdot 10^1$

1 day = 24 hrs $\approx 2 \cdot 10^1$ hrs $\approx 300 \cdot 10^5 \approx 3 \cdot 10^7$ secs.

1 hr = 60 min $\approx 6 \cdot 10^1$ min

1 min ≈ 60 sec = $6 \cdot 10^1$ sec.

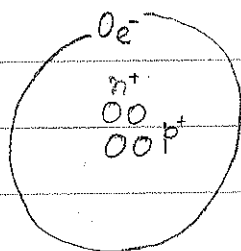
1 GHz $\approx 10^9$ computer then $6 \cdot 10^{23}$ operations.

1 THz $\approx 10^{12}$ " " $6 \cdot 10^{26}$

1 PHz $\approx 10^{15}$ " " $6 \cdot 10^{29}$

$$\Delta t = \frac{\text{dist}}{\text{velo}} = \frac{\text{dist}}{c \rightarrow \text{speed of light}}$$

time during which the light goes through the smallest elementary particle.



electron, neutron, proton have size because of Heisenberg's Uncertainty Principle.

↳ locat_n & velocity of particle cannot be measured at same time.

$$\Delta x \cdot \Delta p \geq \hbar \leftarrow \text{planck's constant.}$$

↓
mv.

So, $\Delta x \geq \frac{h}{m \cdot v}$ → velocity of particle.
 m → mass of particle

proton's size is smallest.

$$\Delta t = \frac{\text{Size of a proton}}{c \text{ (speed of light)}} \approx 10^{-23} \text{ secs.}$$

$$\Delta x \approx 10^{-13} \text{ cm, } c \approx 3 \times 10^8 \text{ km/sec.}$$

Thus, computational steps that can be done during T → $\frac{6 \cdot 10^{17}}{10^{23}} \approx 6 \cdot 10^4$

10^{80} normal particles } If we consider all the particles are
 10^{90} particles overall } computing. Then 10^{130} steps

So, $26^n = 10^{130}$. $(10^{1.5})^n \approx 10^{130}$.

$$n = \frac{130}{1.5} \approx 87.$$

[So, input size 87 for exhaustive search]