$$\text{in}A(n) = \begin{cases} 1 : & \text{if } n \in A \\ 0 , & \text{otherwise} \end{cases}$$ || decidable.

Semidecidable :

$$\text{in } A(n) = \begin{cases} 1 : & \text{if } n \in A \\ \emptyset & \text{if } n \notin A \end{cases}$$

(runs indefinitely)

+ If A is r.e → then A is semi-decidable
  - Have an algorithm that prints all elements in A.

inA(n) waits and checks every hour whether n was printed.

halts

→ true (1) → $n \in A$

→ runs definitely → $\text{In } A(n)$

+ If A is semi-decidable then A is r.e.

b 26, 09 Review : μ Recursive implementation on TM

Big Pic : → μ. recursive fn

→ TM.

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

0, σ, π by (o) (PR) μ. recursion

$\mu m \; x \; P(\vec{a}, m) = f(\vec{a})$

// while → no condition to stop

We know $P(\vec{a}, m)$ is Turing computable

#	$\vec{a}$	#	m	→ 1 :	#	1	#
				→ 0 :	#	#	#

We wanna produce a TM for computing,

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;

```
m = 0
while ( ! P(a→, m))
{
    m++ ;
}
```

#	$\vec{a}$	#	=

P

#	1	# ....

#	#	1

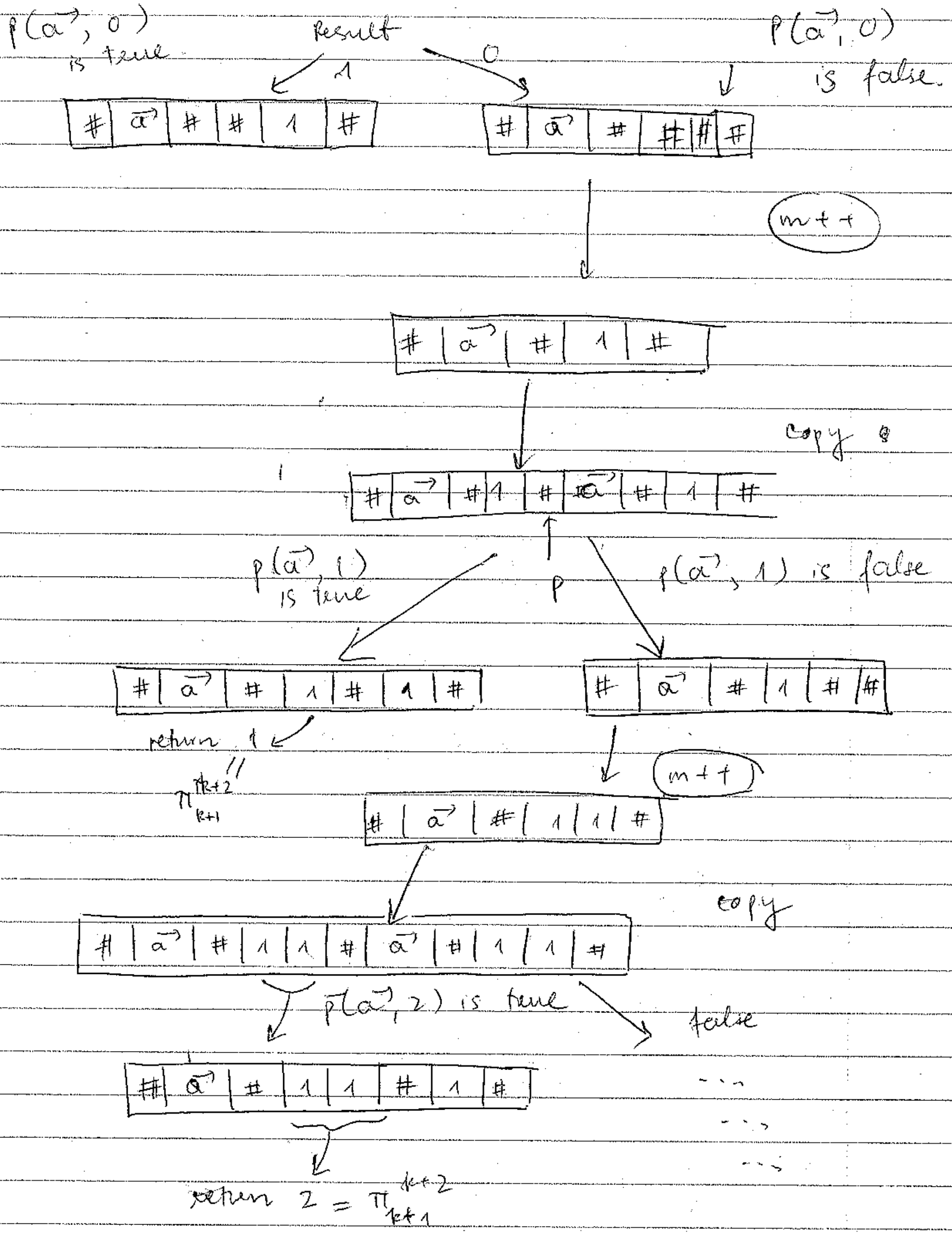
* Correct way :

#	$\vec{a}$	#	#

copy $\vec{a}$

#	$\vec{a}$	#	#	$\vec{a}$	#

↑ start here

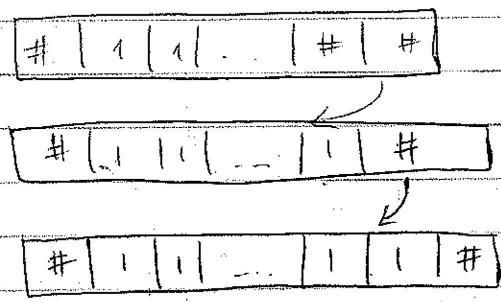

p(a⃗, 0) is true
Result
1
0
p(a⃗, 0) is false
| a⃗ | # | # | 1 |
| a⃗ | # | # | # |
m++
| a⃗ | # | 1 |
copy
| a⃗ | # | 1 | # | a⃗ | # | 1 |
p(a⃗, 1) is true
p
p(a⃗, 1) is false
| a⃗ | # | 1 | # | 1 |
| a⃗ | # | 1 | # |
return 1
m++
| a⃗ | # | 1 | 1 |
copy
| a⃗ | # | 1 | 1 | # | a⃗ | # | 1 | 1 |
p(a⃗, 2) is true
false
| a⃗ | # | 1 | 1 | # | 1 |
return 2 = π k+2 k+1

Big problem with what we had so far.

+ when we had a negative result, it was OK.

+ sometimes, actually produced algorithms, but many of these algorithms were unrealistically long

Example : $\sigma(n) = n+1$.



For security, we have integers : 100 - 200 digit long.

10^{100} impossible # of steps.

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

feasible.

+ How do you define an algorithm ?

1930's ⟷ TM, μ rec

+ How do you define a 'feasible algorithm ?

No perfect definition exists.

Existing

sorting O(n log n)

exhaustive search.

A little bit of physics

1 year = 365 days = 3×10^7 seconds/year

$T \approx 20$ billion years
$\approx 2 \cdot 10^{10}$ years
$= 6 \times 10^{17}$ seconds

$$\Delta t = \frac{\text{dist}}{\text{velo.}}$$

$\frac{T}{\Delta t}$ ← smallest possible time interval

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

Heisenberg's uncertainty principle.

$\Delta x \, \Delta p \geq \hbar$ — planck's constant

Q Next Thursday : Quiz / Test.