basic HASHING

- SEARCH
- INSERT
- DELETE

$\left.\begin{array}{l}\bullet\ \text{SEARCH}\\ \bullet\ \text{INSERT}\\ \bullet\ \text{DELETE}\end{array}\right\}$ $O(1)$ expected with assumptions

basic HASHING

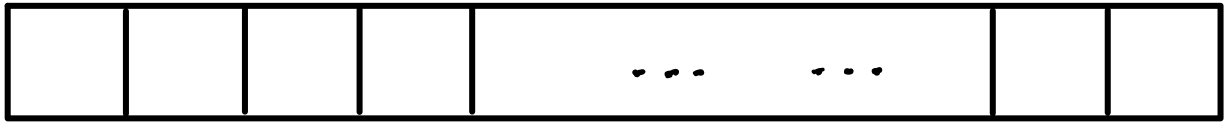
- SEARCH
- INSERT
- DELETE

$\left.\right\}$ $O(1)$ <u>expected</u> with assumptions

not "expected worst-case",
it's just average time.

For some hashing methods, some operations can be $O(1)$ worst-case.

The simplest form of hashing → Direct access table

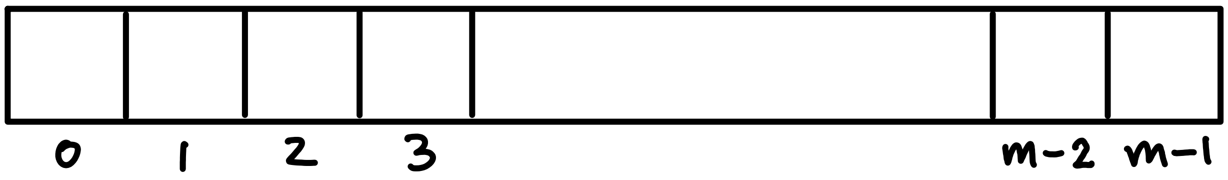


The simplest form of hashing → Direct access table

Assume keys are distinct

and come from a small $\underbrace{\text{set of possible values}}_{\text{Universe}}$, $\mathcal{U}$

e.g., $\mathcal{U} = \{0, 1, 2, \ldots, m-1\}$



The simplest form of hashing → Direct access table

Assume keys are distinct

and come from a small set of possible values, U

Universe

e.g., $U = \{0, 1, 2, \ldots, m-1\}$

$m = 74$

T

0	1	2	3		72	73
0	1	2	3		m−2	m−1

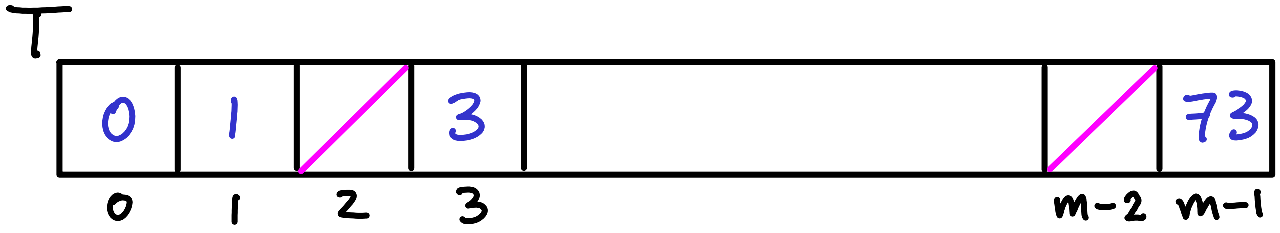
insert$(T, k) \rightarrow T[k] = k$

The simplest form of hashing → Direct access table

Assume keys are distinct and come from a small $\underbrace{\text{set of possible values}}_{\text{Universe}}$, $\mathcal{U}$

e.g., $\mathcal{U} = \{0, 1, 2, \dots, m-1\}$

$m = 74$



insert$(T, k) \rightarrow T[k] = k$ // delete$(T, k) \rightarrow T[k] = \varnothing$

The simplest form of hashing → Direct access table

Assume keys are distinct
and come from a small $\underbrace{\text{set of possible values}}_{\text{Universe}}$, U

e.g., $U = \{0, 1, 2, \ldots, m-1\}$

$m = 74$

T

0	1		3			73
0	1	2	3		m−2	m−1

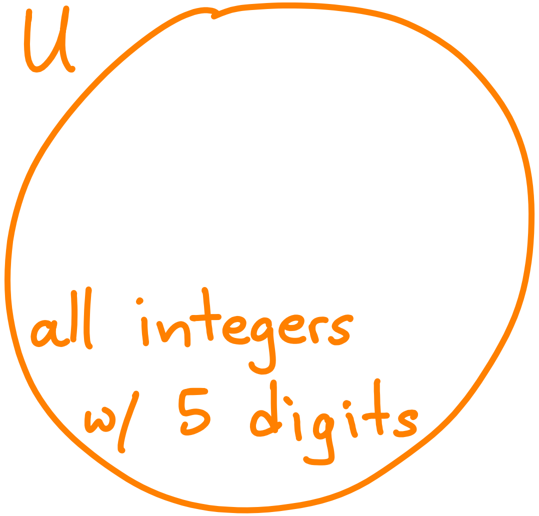
search(T, 3) = 3 // insert(T, k) → T[k] = k // delete(T, k) → T[k] = $\emptyset$

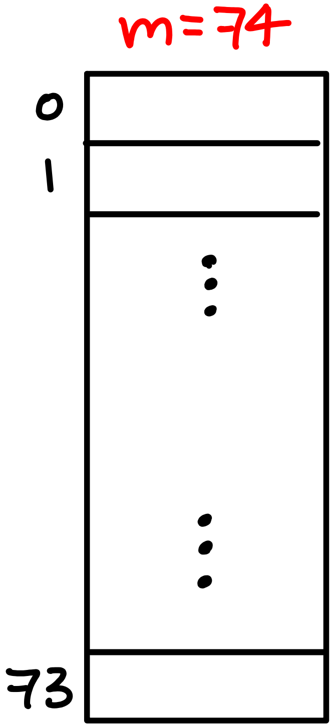
Often U is larger than the available space, m

but we only need to deal with a subset S of U, where $|S| \leq m$

Often U is larger than the available space, m

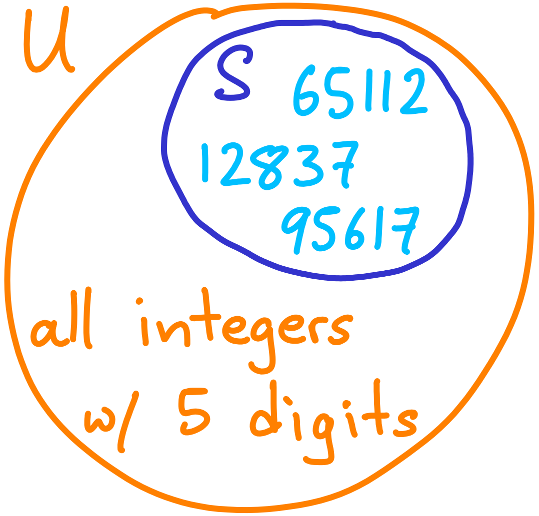
but we only need to deal with a subset S of U, where $|S| \leq m$

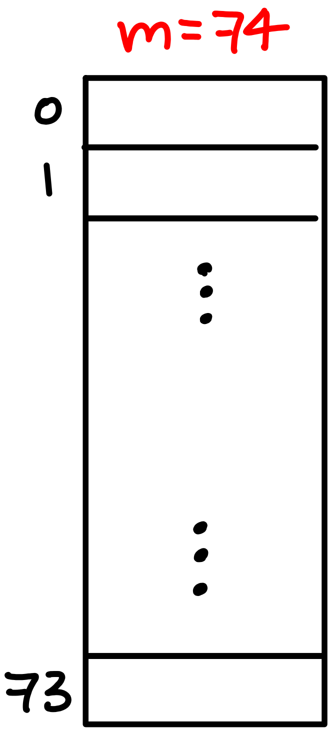




Often U is larger than the available space, m

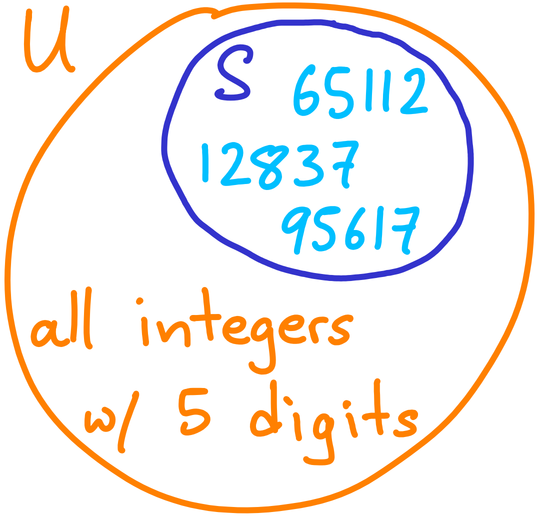
but we only need to deal with a subset S of U, where $|S| \le m$

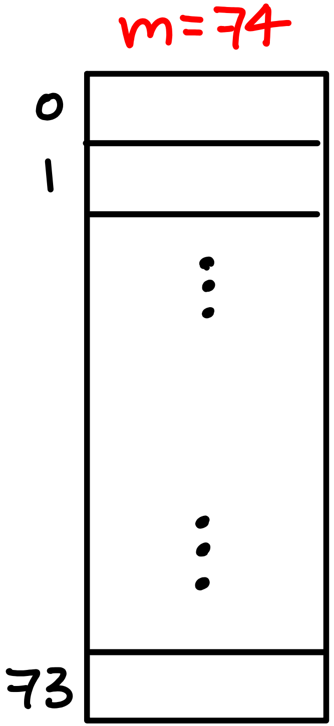




Often U is larger than the available space, m

but we only need to deal with a subset S of U, where $|S| \leq m$



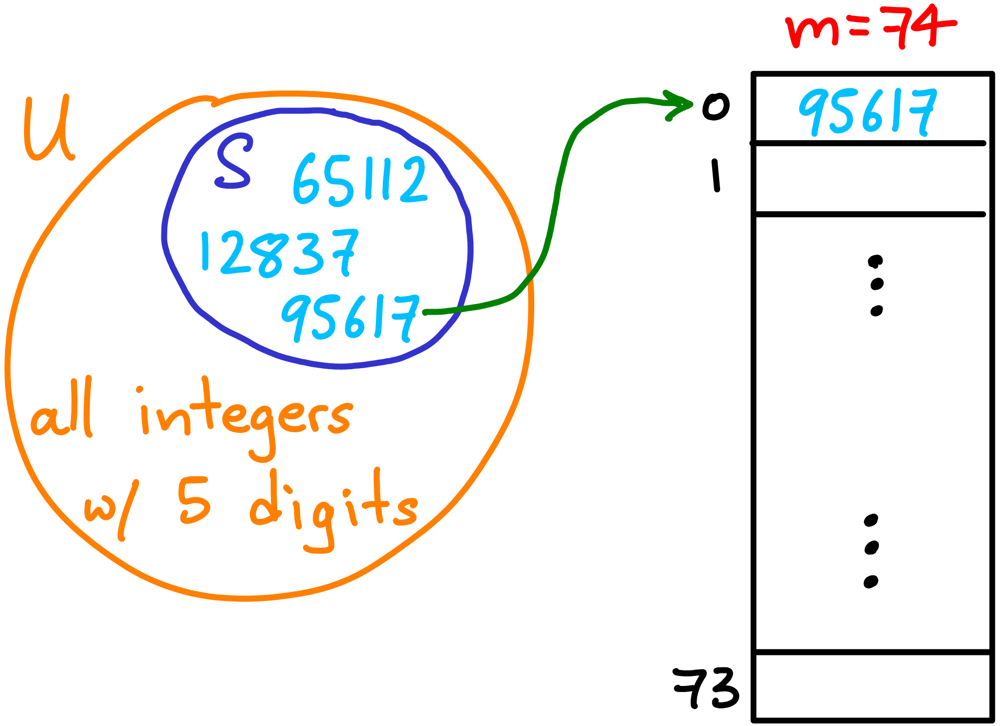


h: hash function

maps keys to T

Often U is larger than the available space, m

but we only need to deal with a subset S of U, where $|S| \le m$



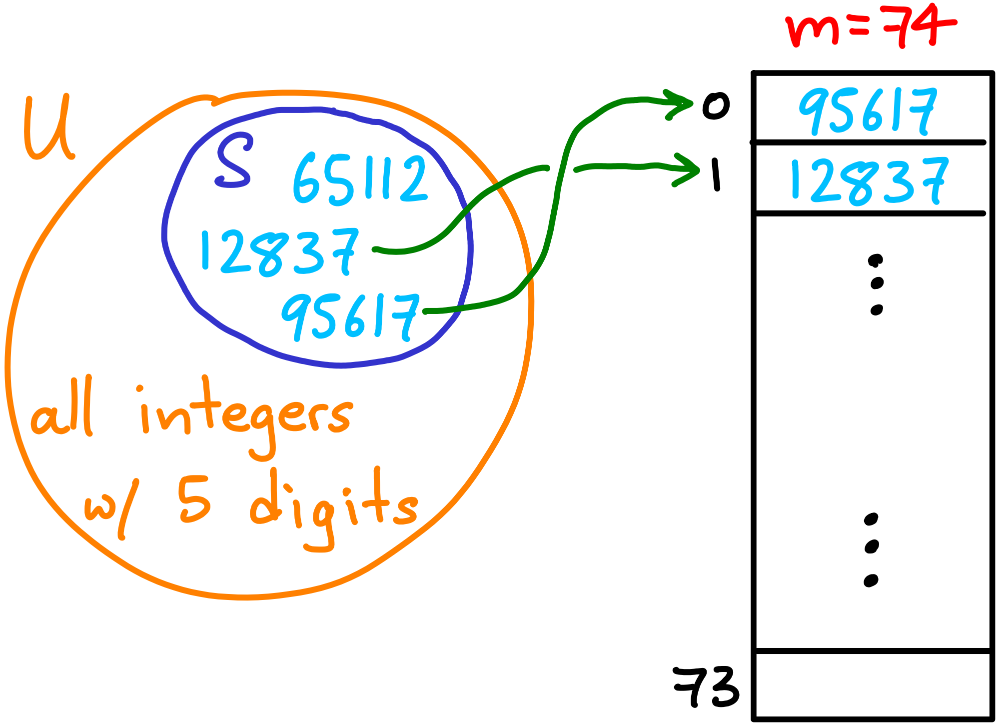
h: hash function

maps keys to T

$h(95617) = 0$

Often U is larger than the available space, m

but we only need to deal with a subset S of U, where $|S| \leq m$



h: hash function

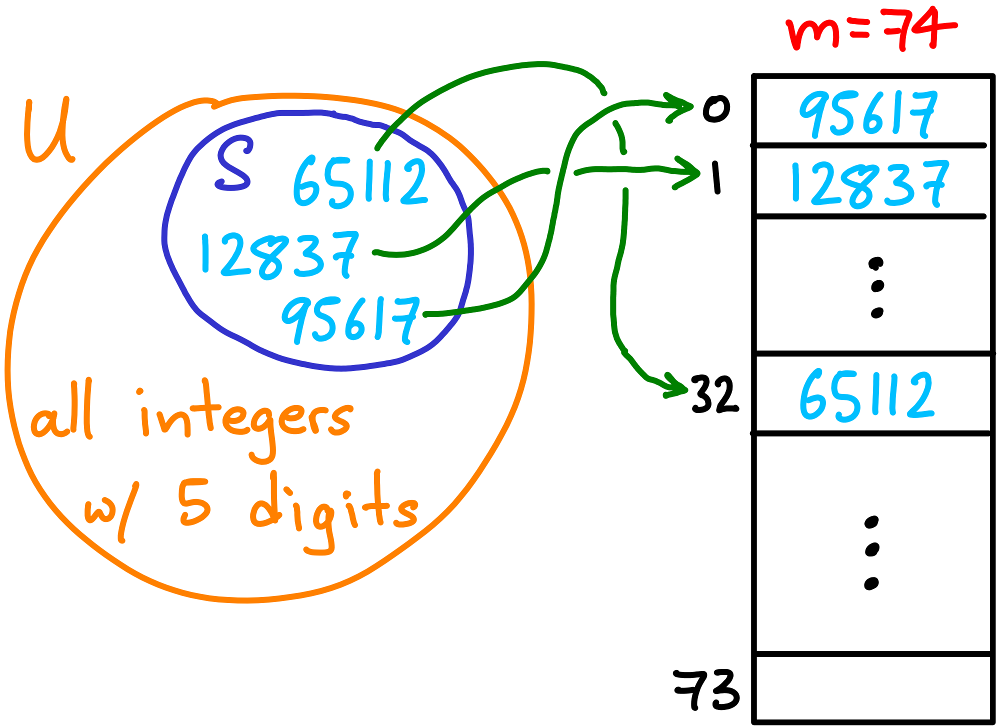
maps keys to T

$h(95617) = 0$

$h(12837) = 1$

Often U is larger than the available space, m

but we only need to deal with a subset S of U, where $|S| \le m$



h: hash function

maps keys to T

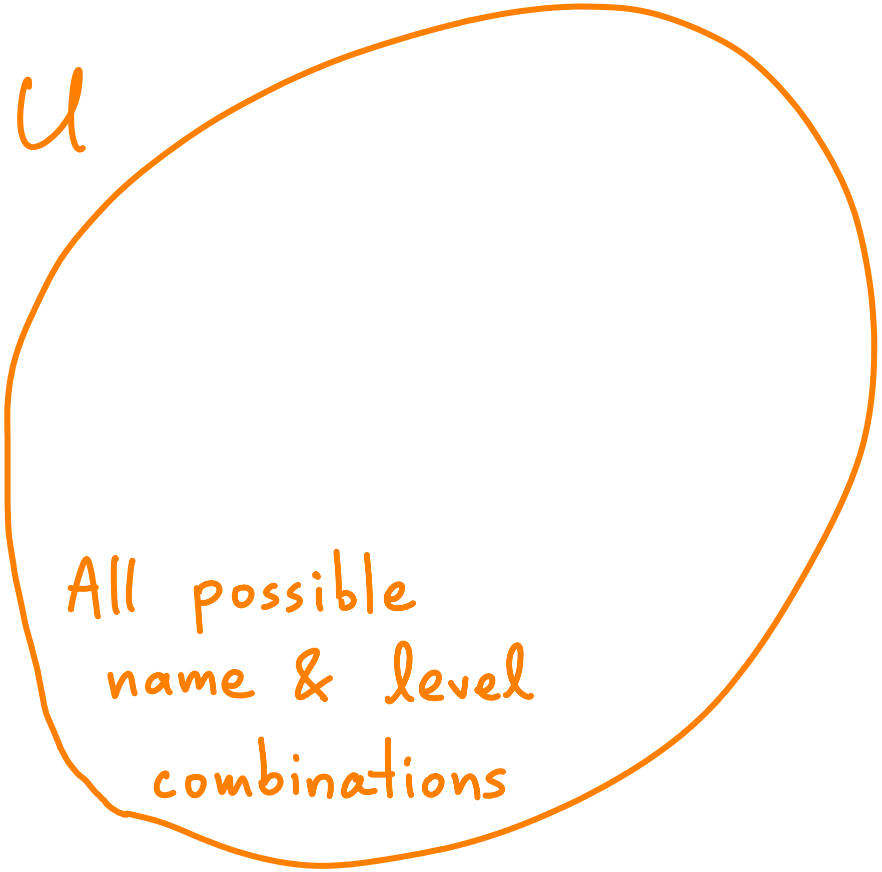
$h(95617) = 0$

$h(12837) = 1$

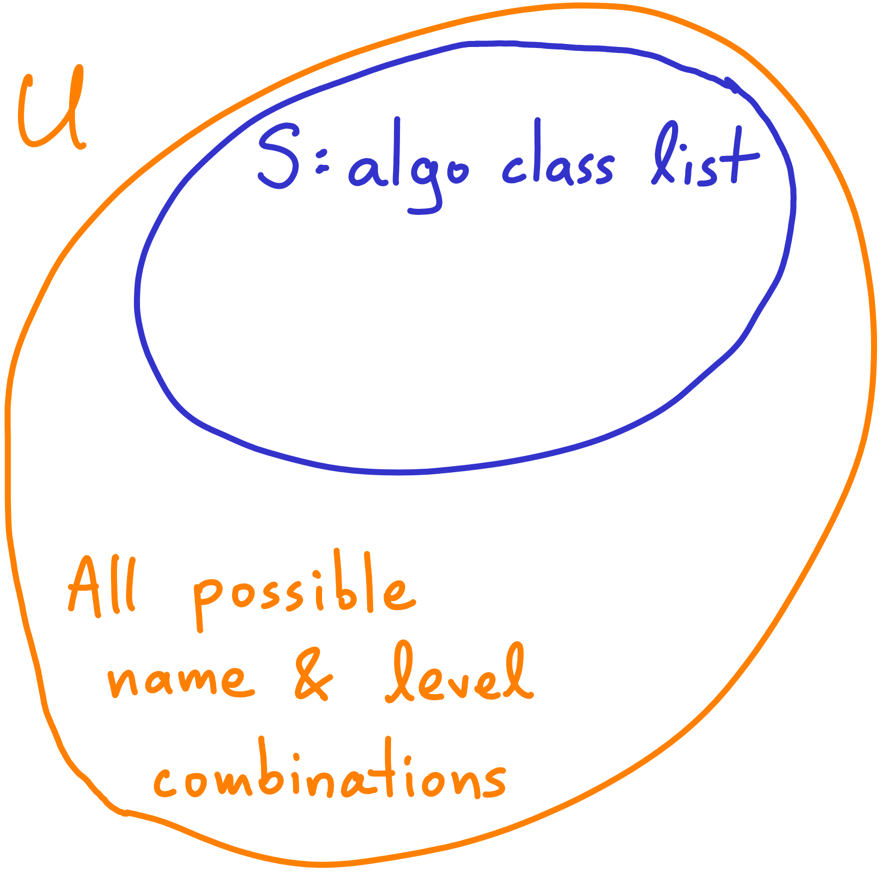
$h(65112) = 32$

Example: look up this semester's ALGO students using only their first name & academic level

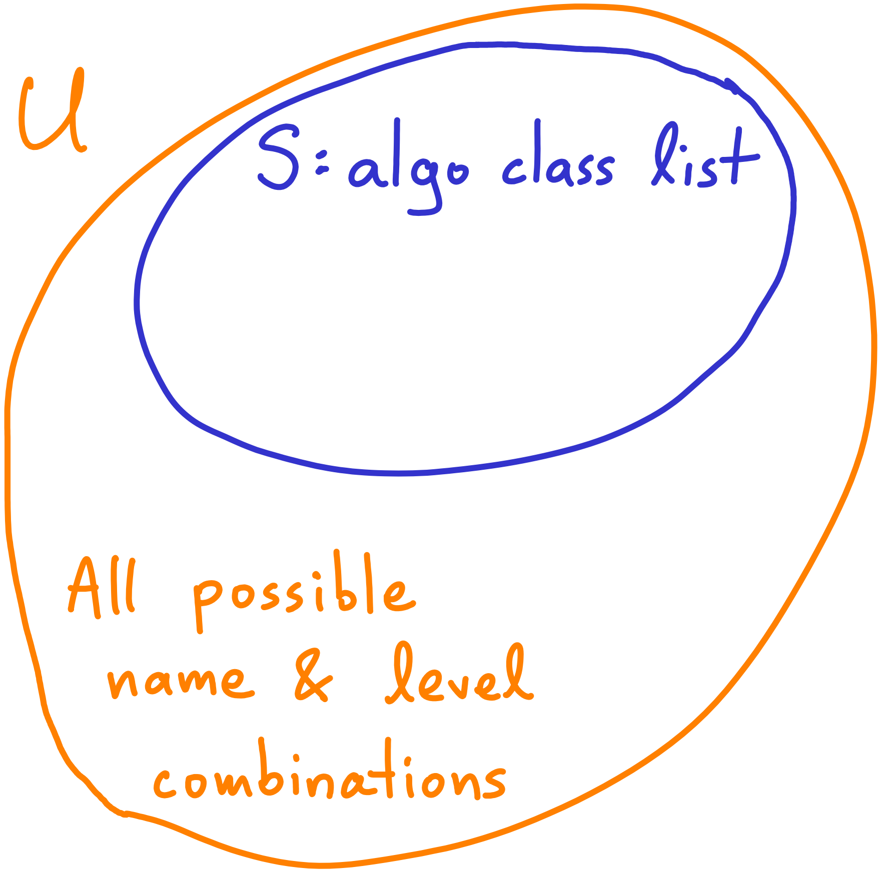
Example: look up this semester's ALGO students using only their first name & academic level



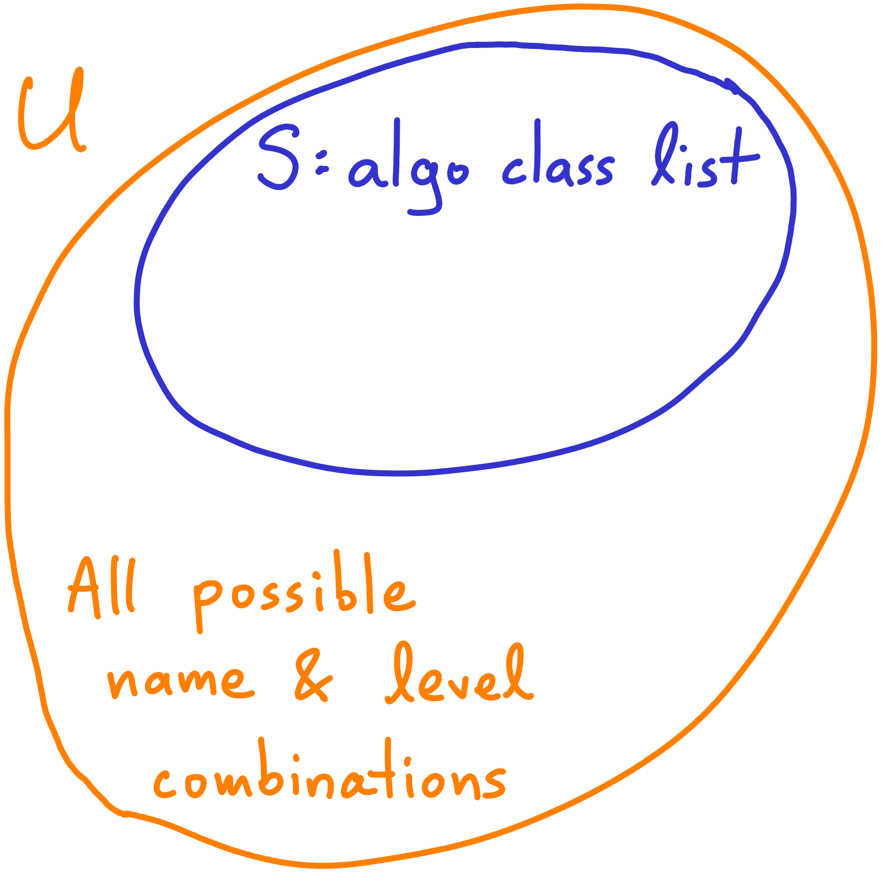
Example: look up this semester's ALGO students using only their first name & academic level



- take 1st letter of name, map to number → $N = \{1 \ldots 26\}$



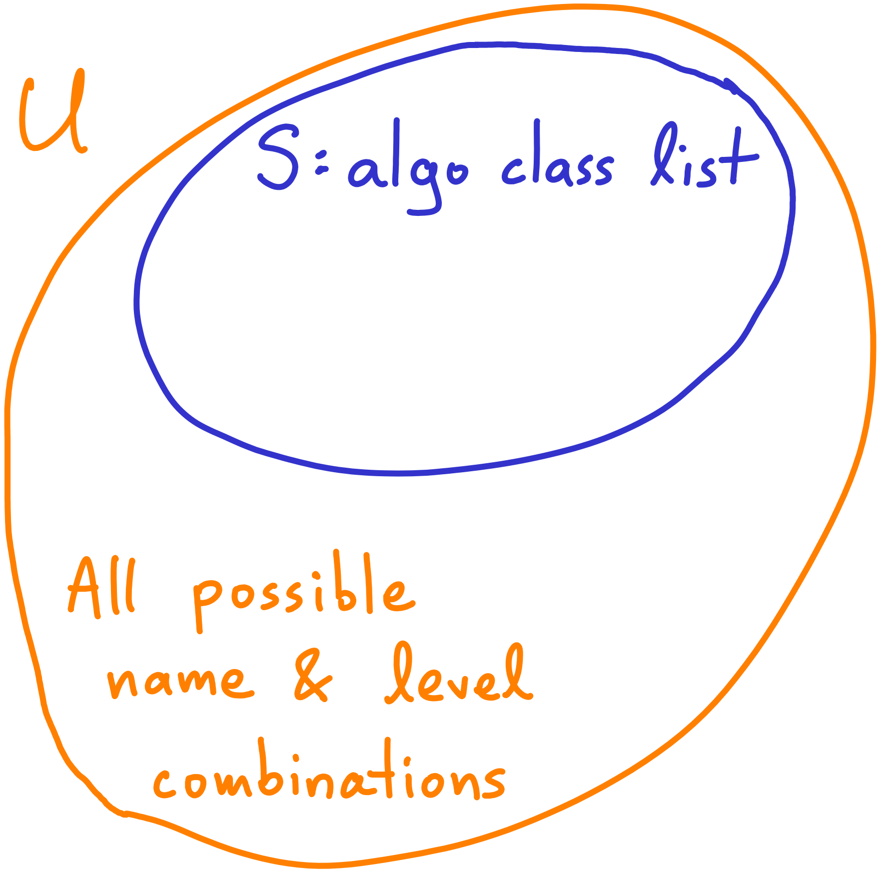
- take 1st letter of name, map to number $\longrightarrow N = \{1 \dots 26\}$
- map level similarly: junior = 3, PhD = 7, etc $\longrightarrow L = \{0 \dots 9\}$



- take 1st letter of name, map to number $\longrightarrow N = \{1 \ldots 26\}$
- map level similarly: junior=3, PhD=7, etc $\longrightarrow L = \{0 \ldots 9\}$

$$h(\text{student}) = h(L,N) = 10 \cdot N + L$$

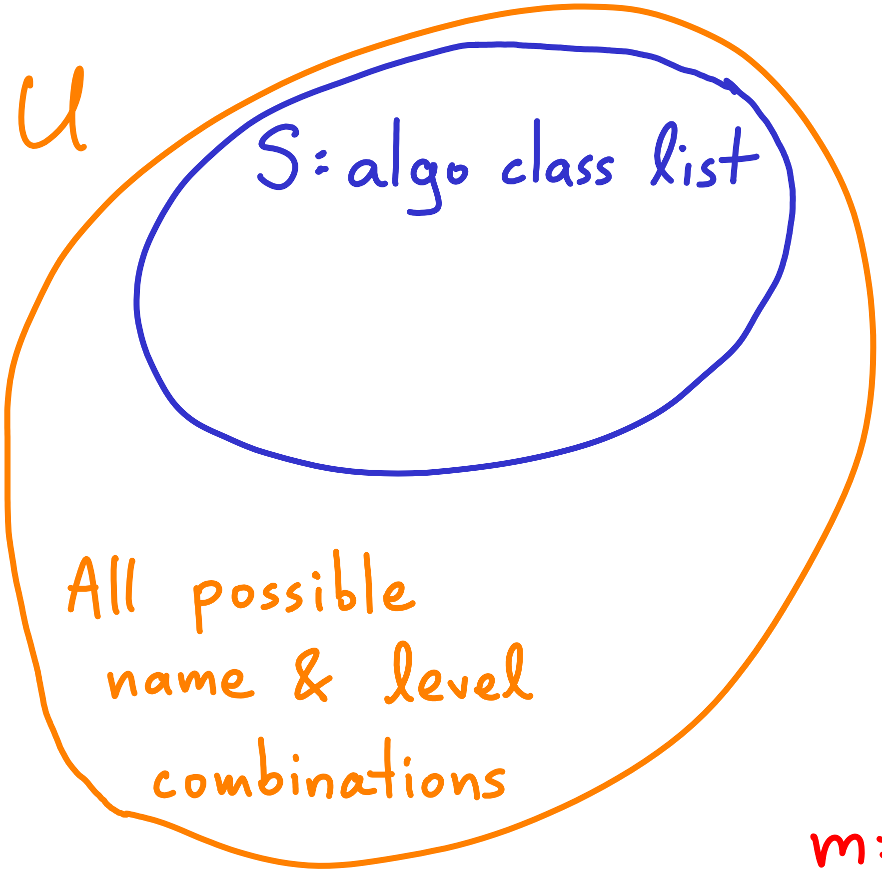
unique per pair N,L



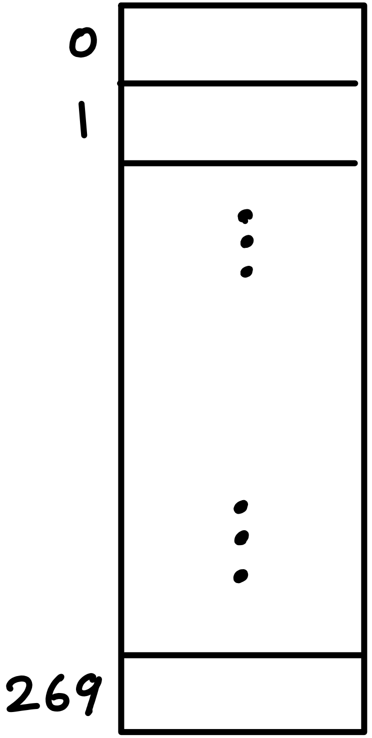
- take 1st letter of name, map to number $\rightarrow N = \{1 \ldots 26\}$
- map level similarly: junior = 3, PhD = 7, etc $\rightarrow L = \{0 \ldots 9\}$

$$h(\text{student}) = h(L, N) = 10 \cdot N + L$$

└ unique per pair N, L



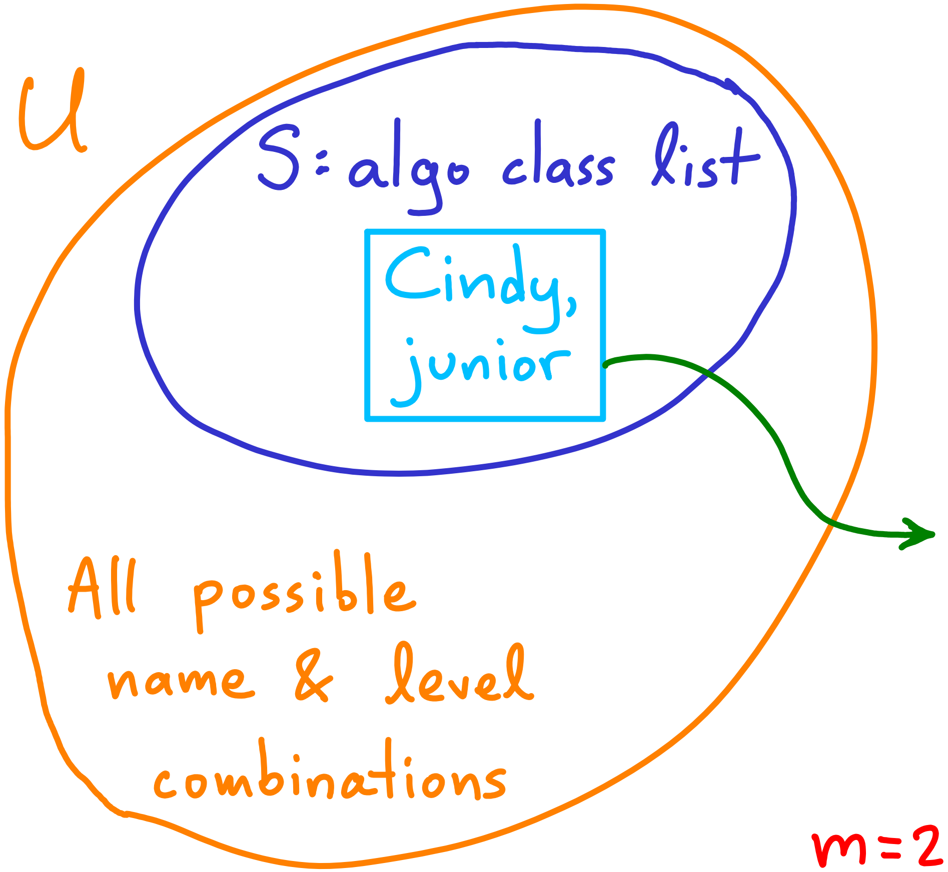
m = 270



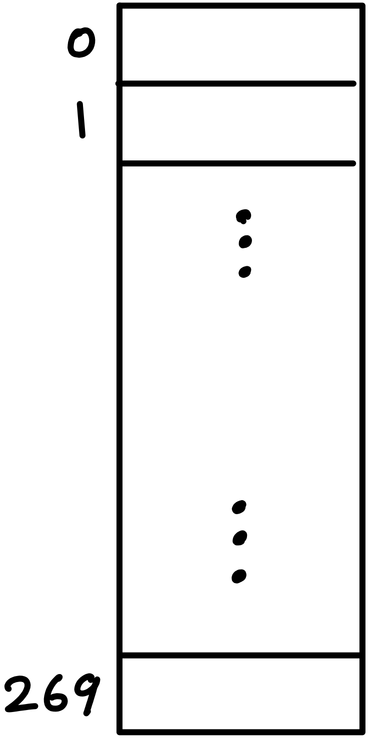
- take 1st letter of name, map to number → $N = \{1 \ldots 26\}$
- map level similarly: junior = 3, PhD = 7, etc → $L = \{0 \ldots 9\}$

$$h(\text{student}) = h(L, N) = 10 \cdot N + L$$

Unique per pair N, L



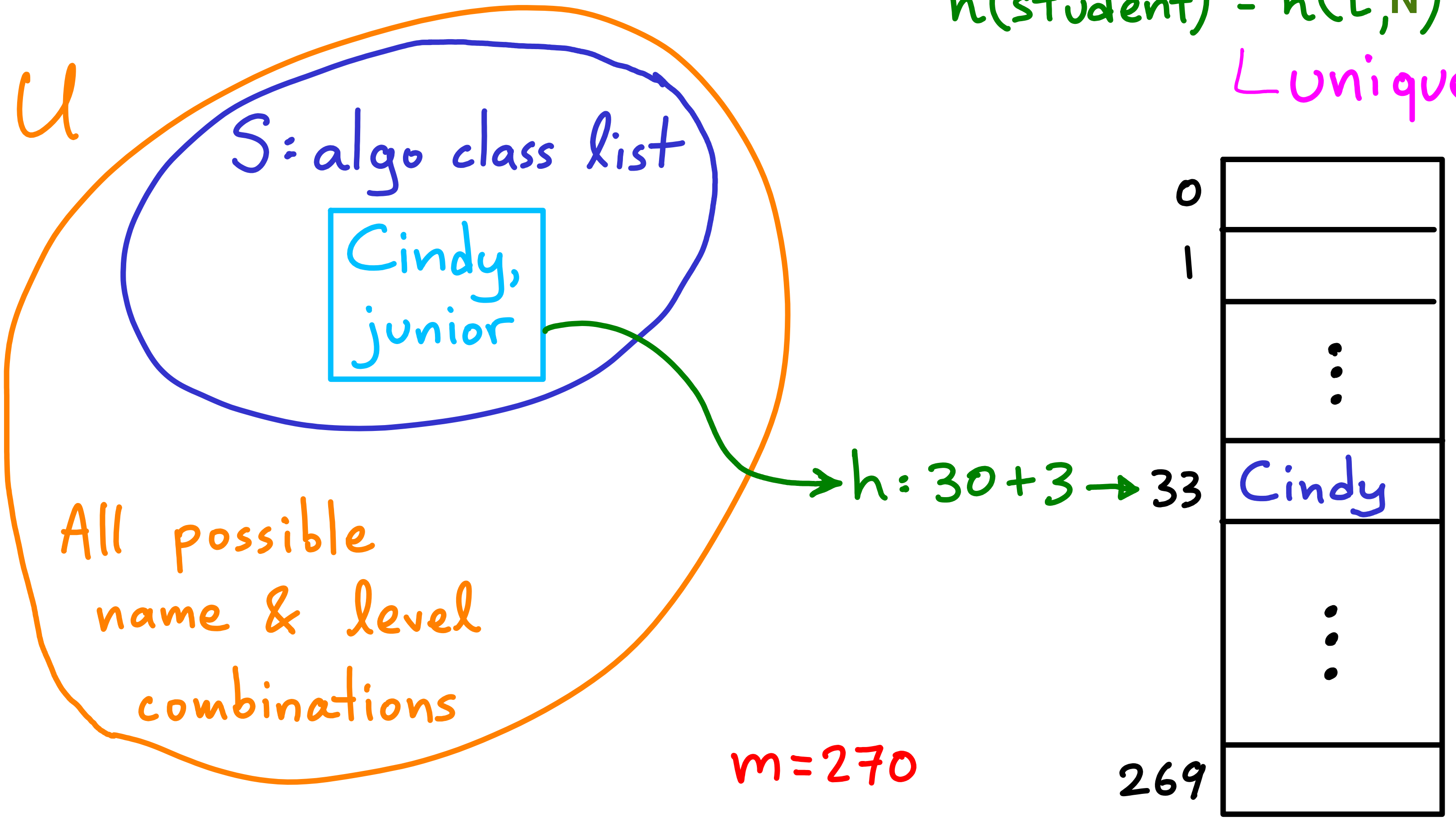
m = 270



- take 1st letter of name, map to number $\rightarrow N = \{1 \ldots 26\}$
- map level similarly: junior = 3, PhD = 7, etc $\rightarrow L = \{0 \ldots 9\}$

$$h(\text{student}) = h(L, N) = 10 \cdot N + L$$

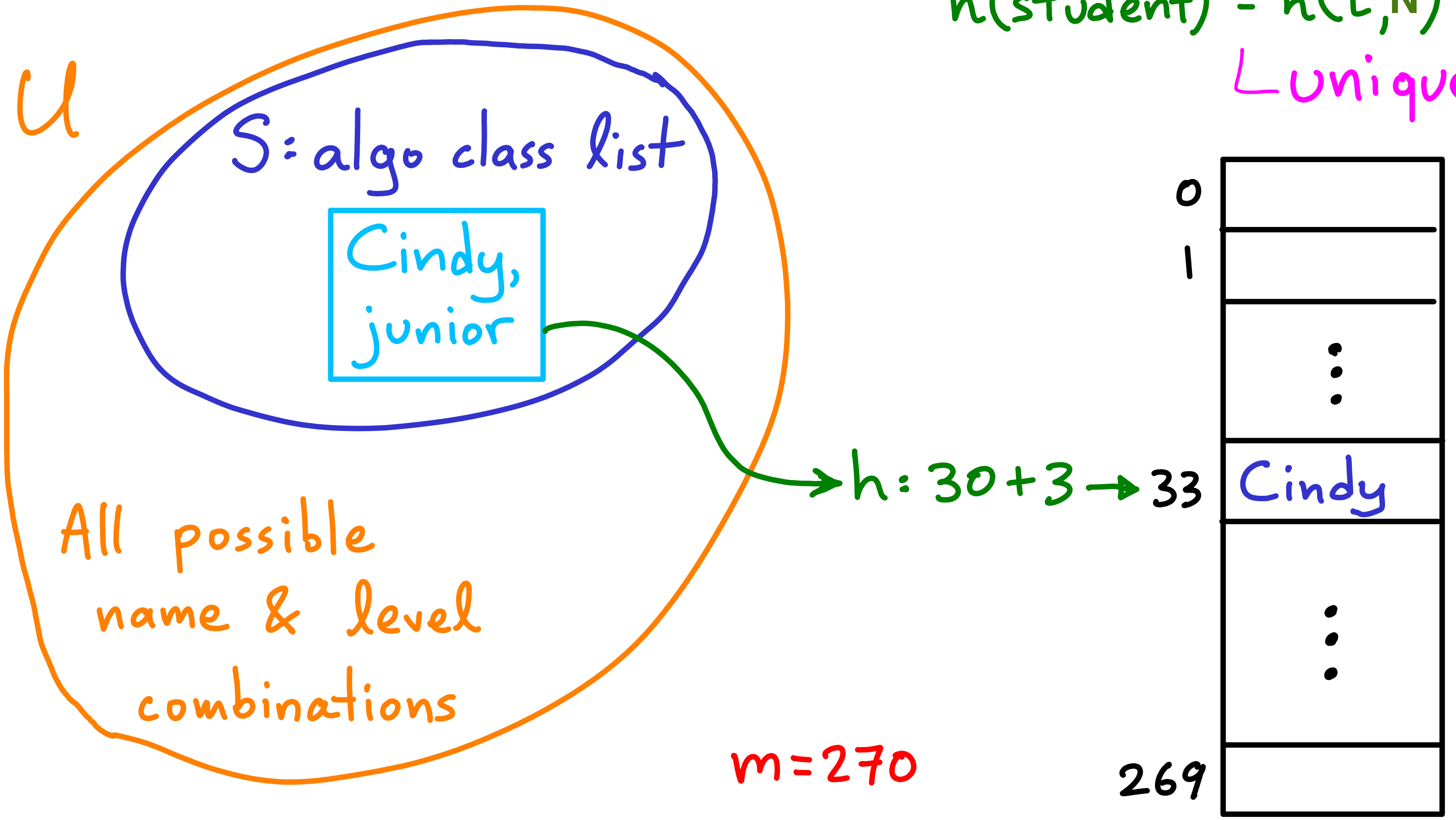
unique per pair N, L



- take 1st letter of name, map to number $\rightarrow N = \{1 \ldots 26\}$
- map level similarly: junior = 3, PhD = 7, etc $\rightarrow L = \{0 \ldots 9\}$

$$h(\text{student}) = h(L, N) = 10 \cdot N + L$$

unique per pair N, L

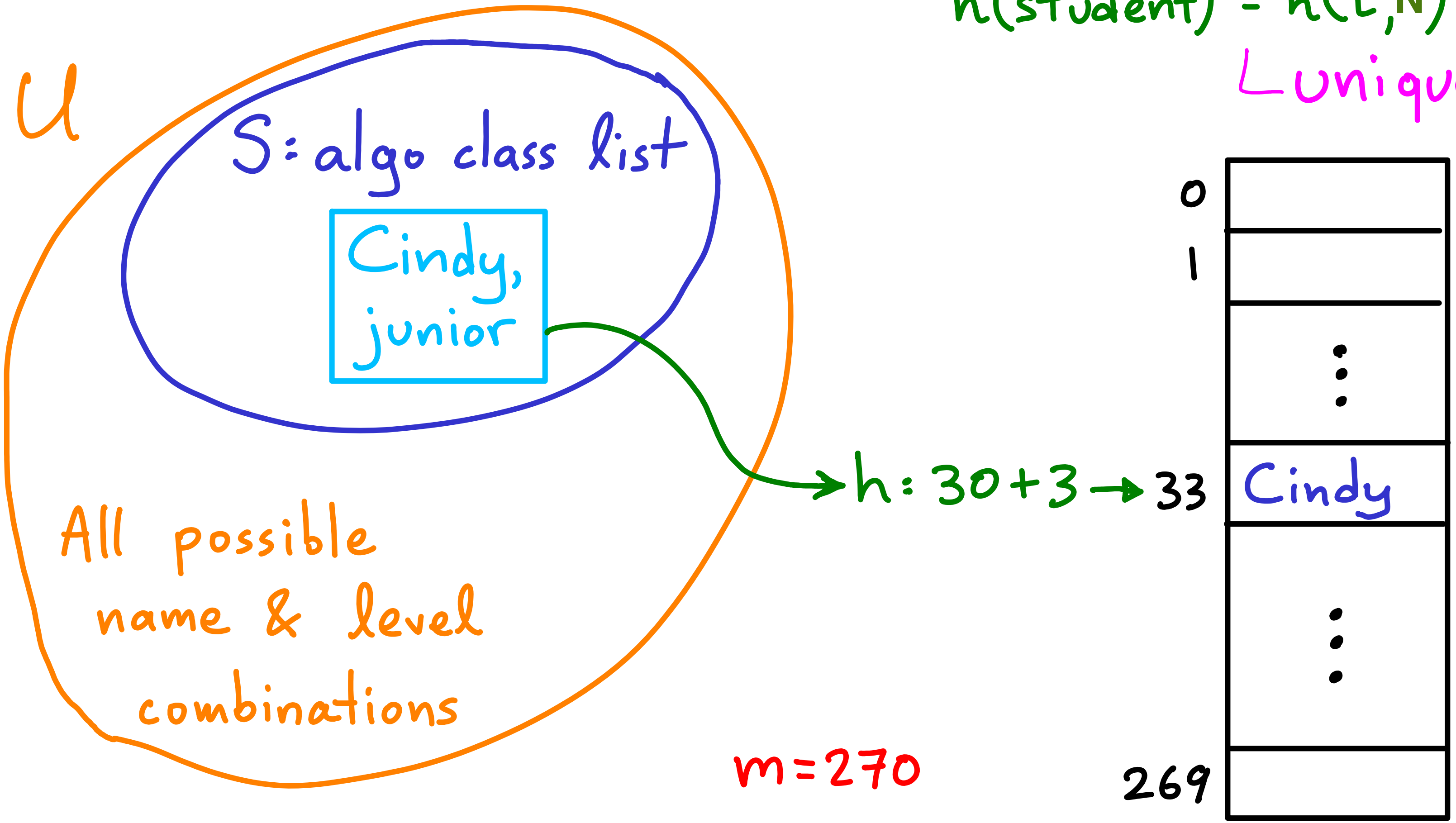


PROBLEMS?

- take 1st letter of name, map to number → $N = \{1 \ldots 26\}$
- map level similarly: junior = 3, PhD = 7, etc → $L = \{0 \ldots 9\}$

$$h(\text{student}) = h(L, N) = 10 \cdot N + L$$

unique per pair N, L



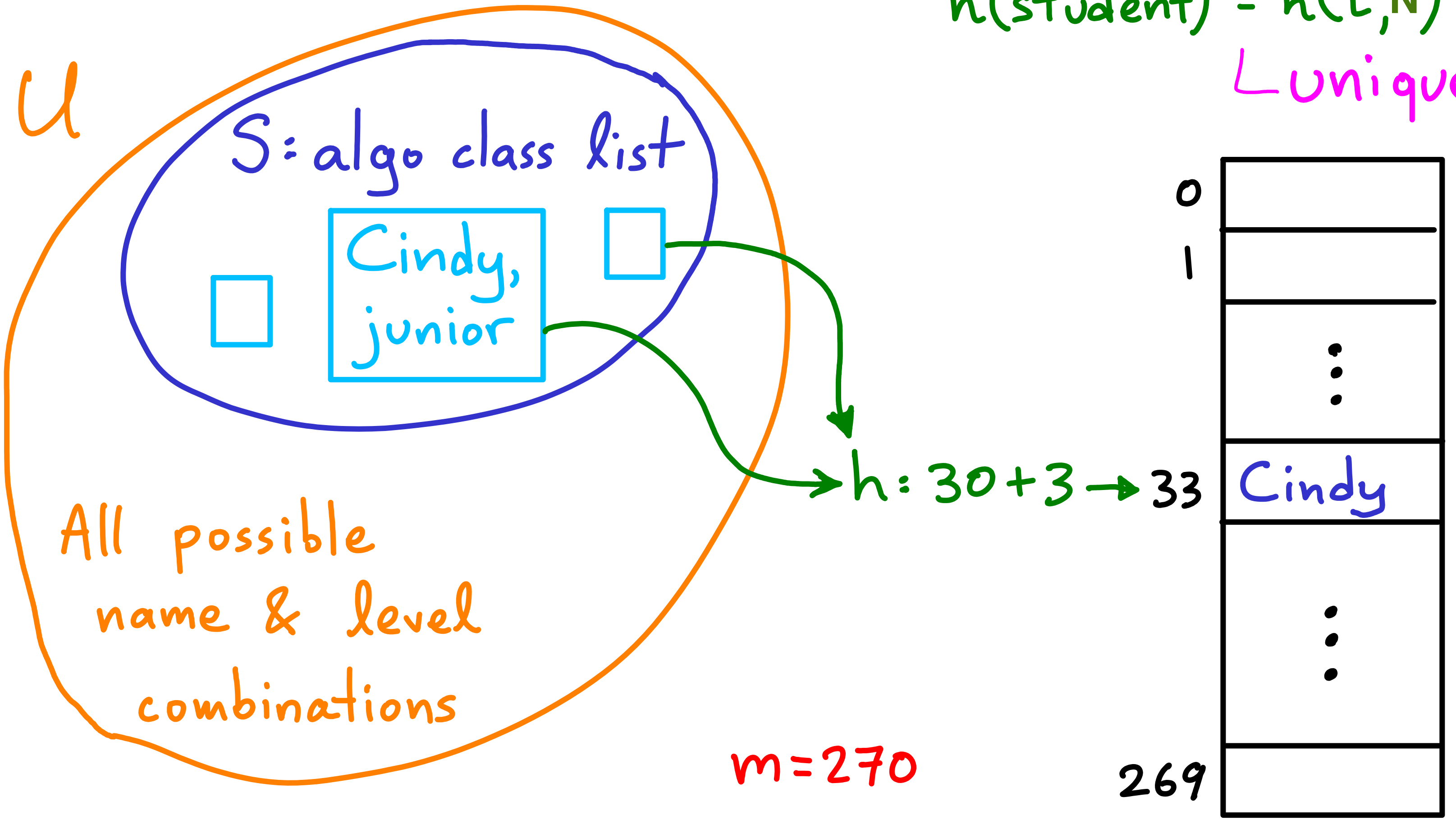
PROBLEMS?

(1) some permanently empty slots

- take 1st letter of name, map to number $\rightarrow N = \{1 \ldots 26\}$
- map level similarly: junior = 3, PhD = 7, etc $\rightarrow L = \{0 \ldots 9\}$

$$h(\text{student}) = h(L, N) = 10 \cdot N + L$$

└ unique per pair N, L



PROBLEMS?

(1) some permanently empty slots

(2) collisions

Could use more of the given info to design a more complicated $h(\ )$

↪ might reduce collisions

Could use more of the given info to design a more complicated h( )

↳ might reduce collisions

But that involves costly processing

Could use more of the given info to design a more complicated h( )

↳ might reduce collisions

But that involves costly processing

and will need to be repeated if S changes (next semester)

Could use more of the given info to design a more complicated h( )

↳ might reduce collisions

But that involves costly processing
and will need to be repeated if S changes (next semester)

We want to use a simple h( ) and deal with collisions