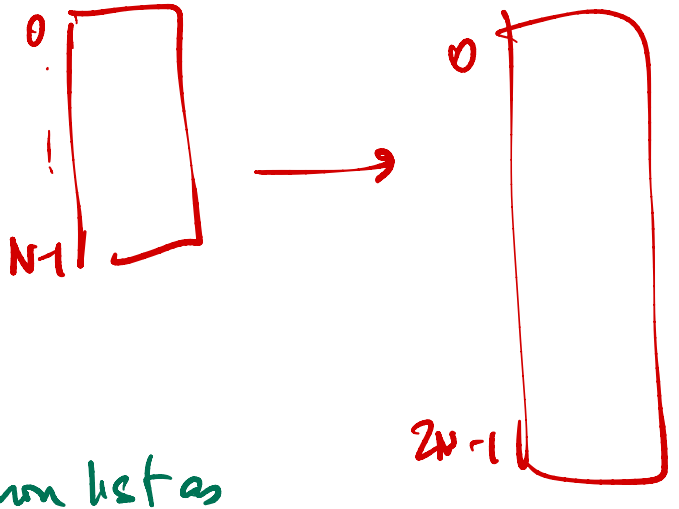
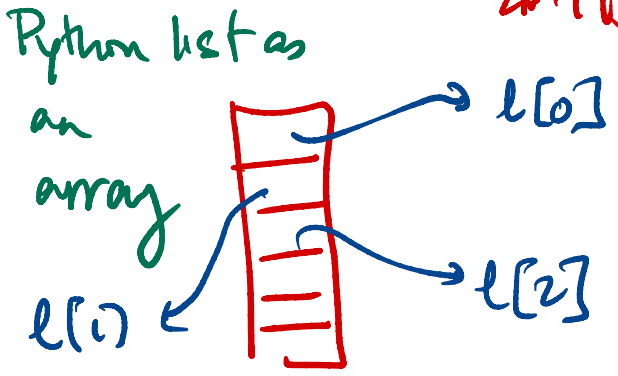
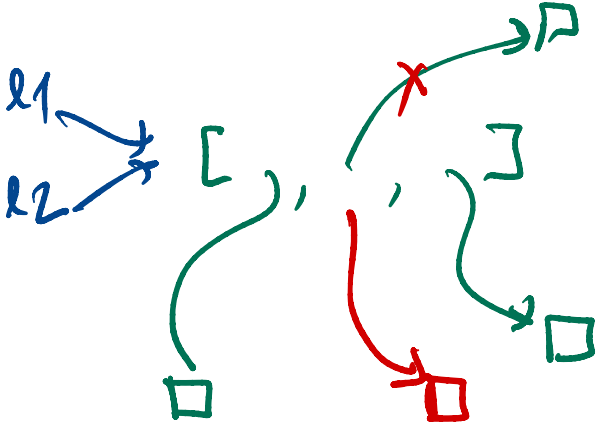
Python lists are arrays (flexible) – not uniform type

how does this work?



→ double each time you run out of space





for k in d

[for k in d.keys()]

if k in d: — compute h(k) & check

— cost independent of k, dictionary size

Given l1 & l2, check if they have a common element

```
for x in l1:          # n
    for y in l2:      # n
        if x==y:
            return(True)
return(False)
```

n^2

2 dictionaries share a key

```
for k1 in d1:         # n
    ~~for k2 in d2:~~     # n
    if k1 in d2:
        ~~if k1==k2:~~
            return(True)
return(False)
```

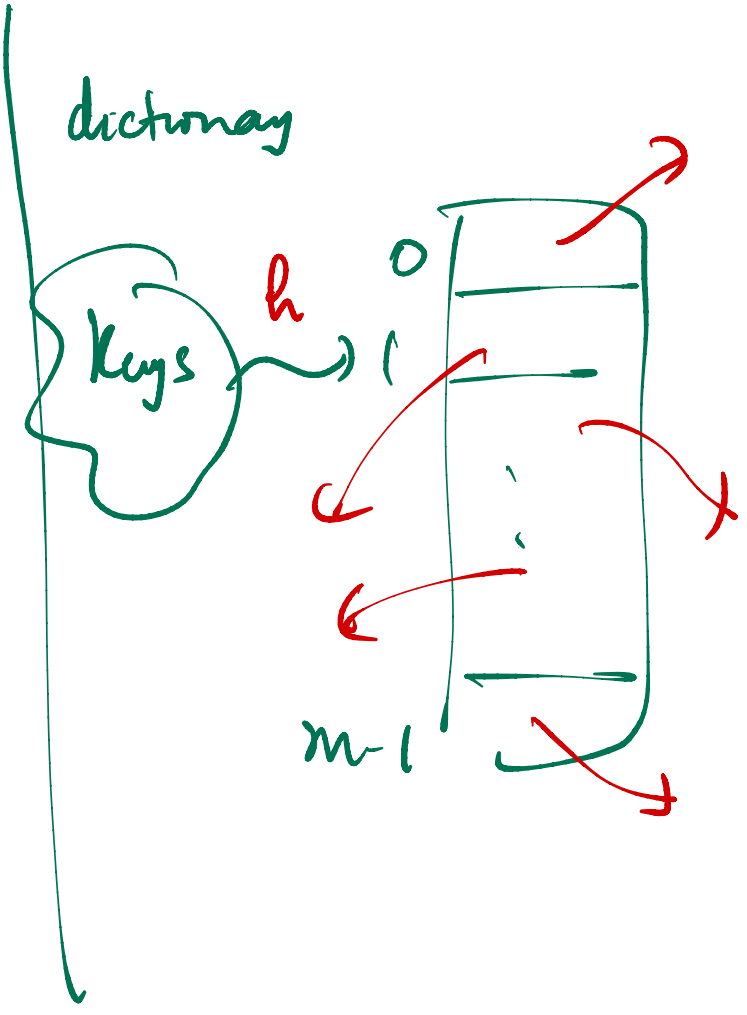
n^2 n

Therefore:

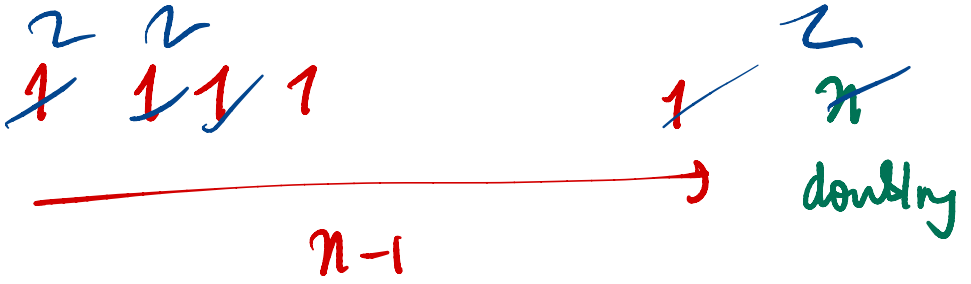
```
d1 = {}
for x in l1:
    d1[x] = True
for y in l2:
    if y in d1:
        return (True)
return (False)
```

n

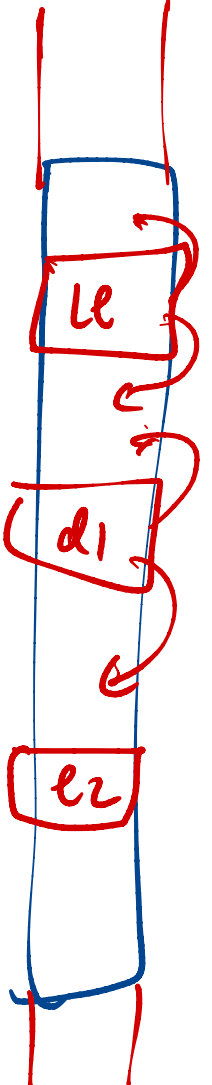
n

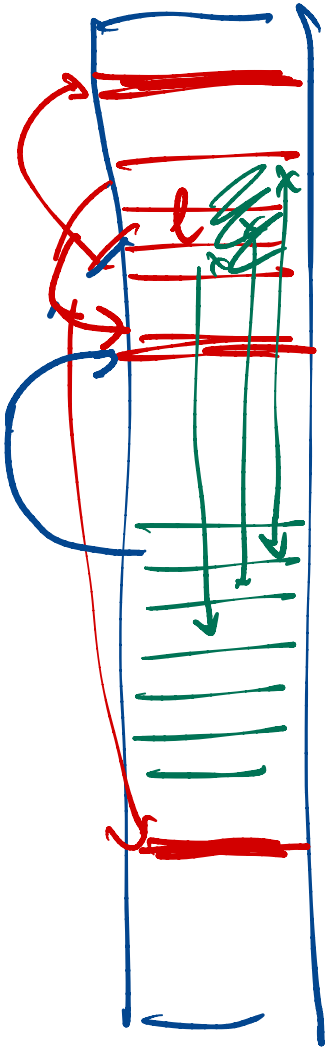


Amortised



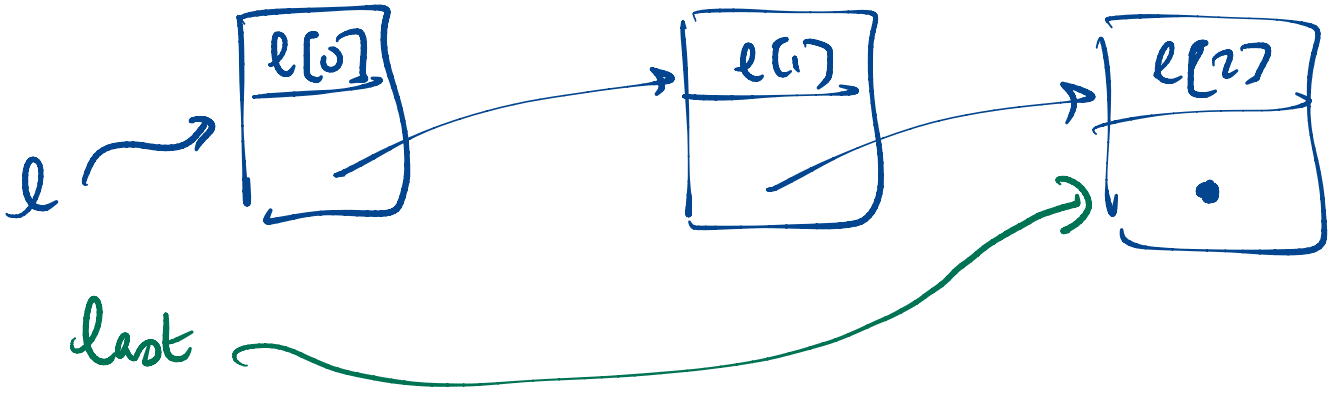
$$\frac{(n-1+n)}{2} \times 2$$

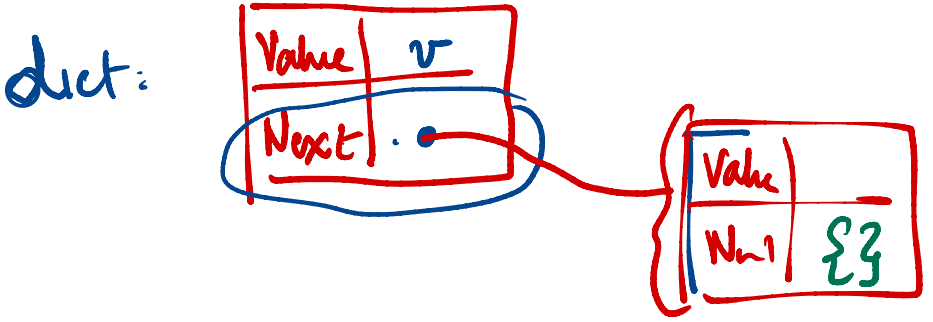




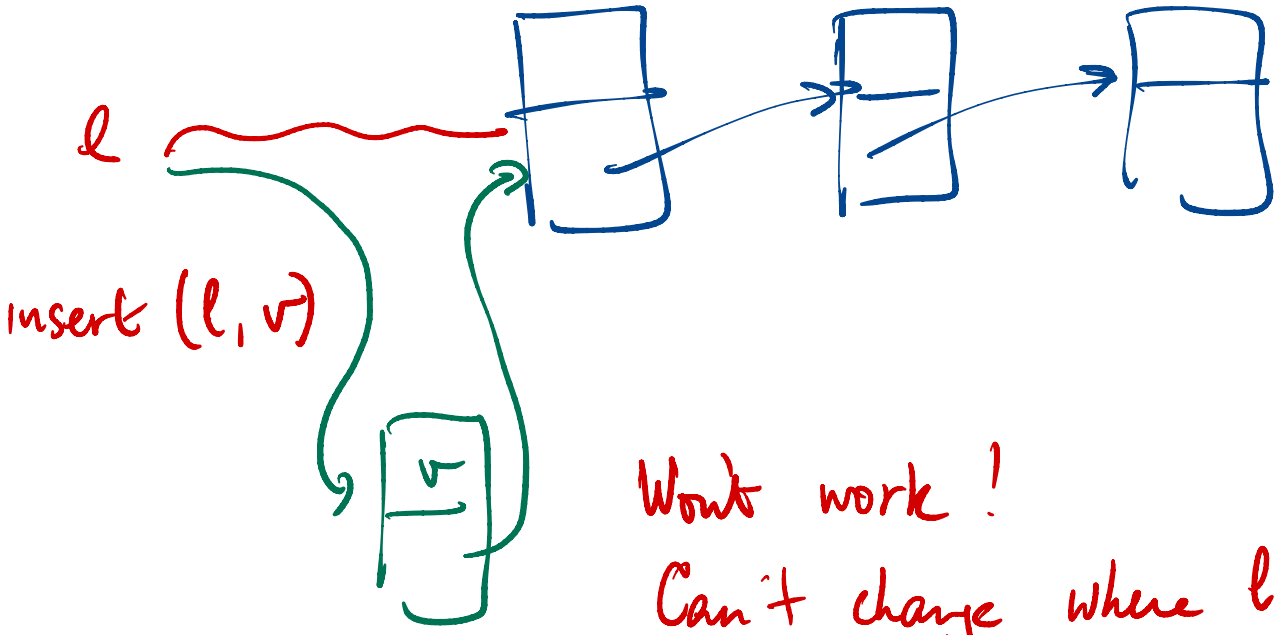
run out of space

"Real List"





Insert at beginning

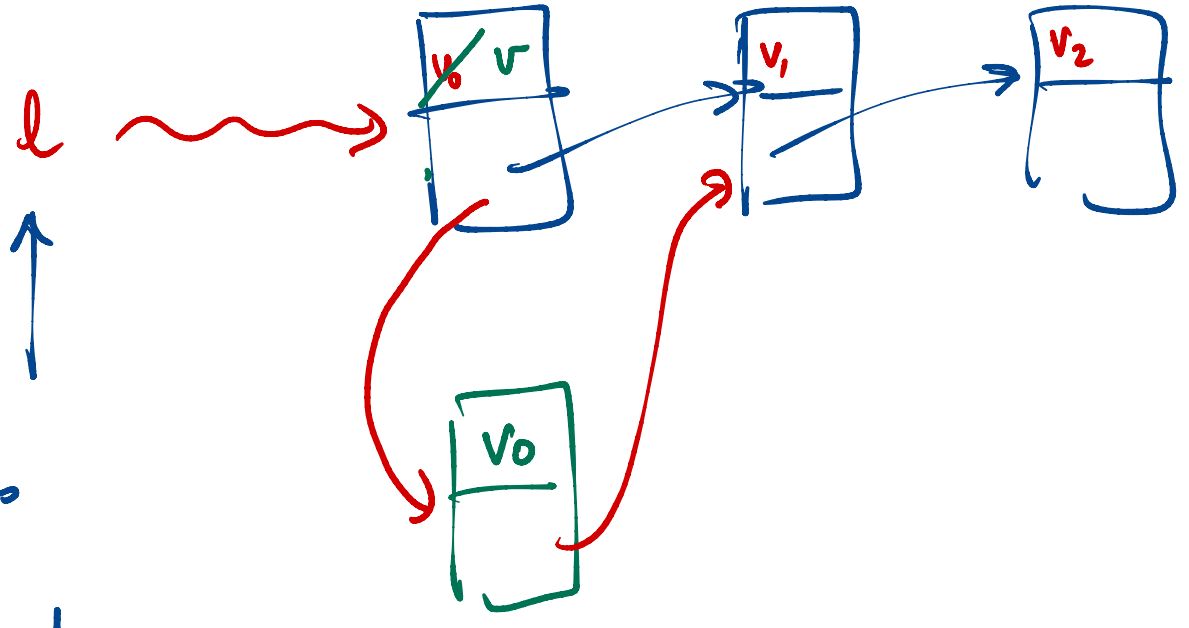


Wont work!

Can't change where l points to inside a function

Instead

insert(l, v)



l has not changed

Set Comprehension

$$S' = \{ f(x) \mid \underbrace{x \in S}_{\text{Starting Set}}, \; p(x) \}$$

$p(x)$: predicate → True/False

$$S' = \{ s \mid s \notin s \}$$

Is $S' \in S'$?

Russell's Paradox