Lists Arrays

Sequences

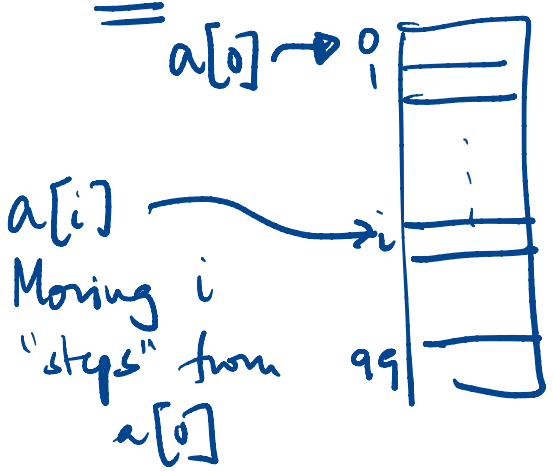
Dictionaries

Key-Value store

Array: Block of storage - Uniform type

Predefined size a

100



Accessing a[i] takes same amount of time for any i

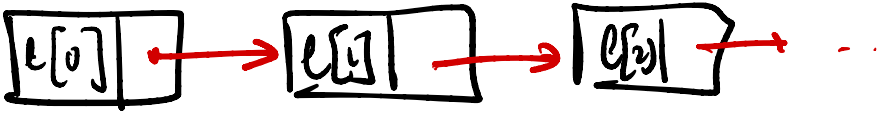
"Random Access"

Swap a[i] & a[j]

x, y = y, x

a[i], a[j] = a[j], a[i] — should not depend on i, j

List Sequence of "cells" that are linked together, like a train



Each cell would be far away from its neighbours

Accessing l[i] is proportional to i

l[i], l[j] = l[j], l[i] — cost depends on i & j

Why use a list?

Flexibility

Adding & deleting elements easily

Array a[0..99] Currently using a[0..49]

Insert a new value after a[30]

a[0] .. a[29] a[30] a[31] .. a[49]

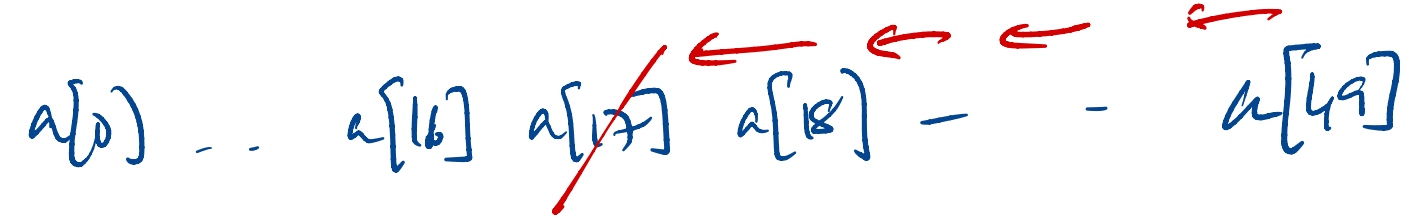
v

Need to make space

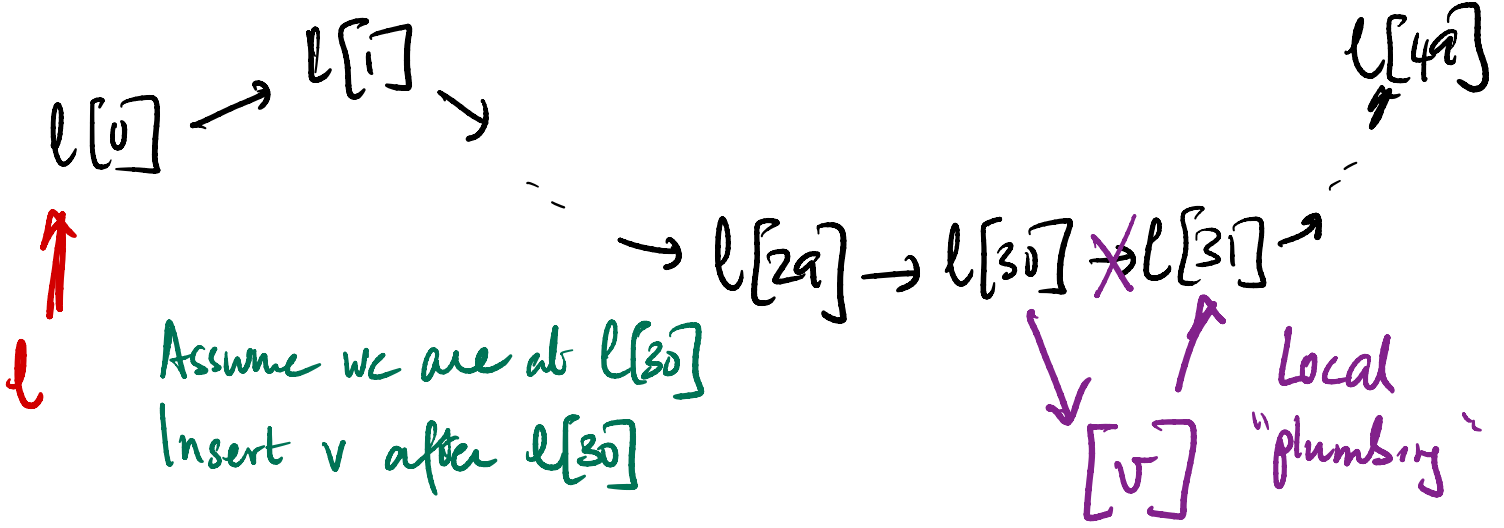
Shift a[49] → a[50]
a[48] → a[49]
⋮
a[31] → a[32]
a[31] = v

Shifting values depending on i

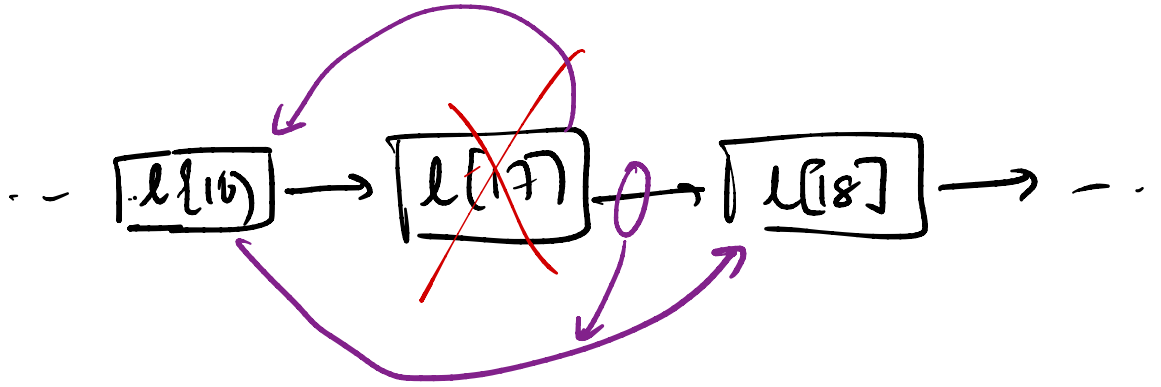
Delete a[17]



Need to shift to ensure contiguous storage, no gaps



Analogous for delete



Are Python lists arrays or lists ?

Dictionaries?

$k_1 \rightarrow v_1$

$k_2 \rightarrow v_2$

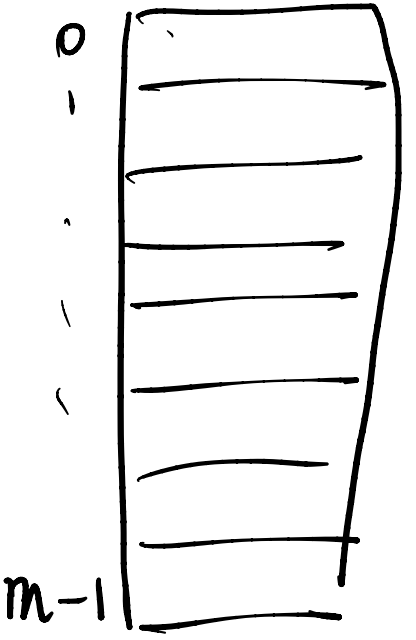
⋮

$k_n \rightarrow v_n$

$v_1 \ldots v_n$ are stored in locations $0 \ldots m-1$, but not in a fixed order

Storage is a fixed block, like array

Function: $k_i \rightarrow [0 \ldots m-1]$



Example key k → write in binary → number n_k

$n_k \bmod m \rightarrow [0..m-1]$

Space of keys is large, the range of locations is small

There will always be collisions $f(k_1) = f(k_2)$

Strategies to deal with this

- Have a second function
- Move to next free location

These functions are called hash functions

h: large set $\rightarrow$ Very small set

Download a software package $\rightarrow$ get a "fingerprint" or a "hash certificate"

SHA-256 — output is 256 bytes
= 256 × 8 bits

Dropbox

Python list?

Expanding array

l = []

Run out of space

Double the size

l.append(x) - fast

l.insert(0,x) - slow