Lecture 6, 27 August 2026

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Programming and Data Structures with Python

Dictionaries

Keys → Value

hash function h : LargeSet → SmallSet

"infinite"

Collisions

Hashing

Collision: $x \neq y$, but $h(x) = h(y)$

Theoretically unpredictable behaviour

Collision management

Python dictionaries work "perfectly"

Looking up $d[k]$ — only cost is computing $h(k)$

"constant" time

"Adversarial" argument

'Unorganized list':

7 in l must look at every value in l

↓

False, and some value is skipped

Unique values in list

Given l, compute the list of unique values in l

If use a list uniqlist to keep track — $\approx n^2$

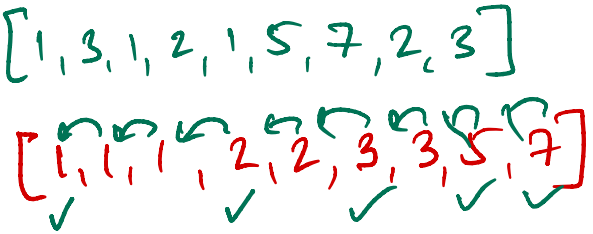
Dictionary

- Each value v in l is a key

for v in l : $\approx n$ (assuming dictionary works)

d[v] = 1 ⟶ output d.keys()

Sorting



$\approx n$

$2^{10} \simeq 1000$

$2^{20} \simeq 1000000$

Sorting is $n \log_2 n$

Intersection of 2 lists

l1 = [1, 7, 3, 5, 6]

l2 = [8, 2, 1, 3, 7]

⇝ [1, 7, 3]

for x in l1

check if x in l2

intersect = []

for x in l1:

if x in l2:

intersect.append(x)

for y in l2:

if x == y

Using dictionaries ?

Elements common to l1 & l2

d1 → keys are values in l1 $\approx n$

for x in l2

check if x is in d1.keys() $\approx n$

append to output

x in l

↳ $\approx n$ scan

x in d.keys()

↳ constant time

Unique vs Intersection

Unique value

Naive list — $\approx n^2$

Dictionary $\approx n$

Sort + blocks $\times n \log n$ + n close to n

List intersection

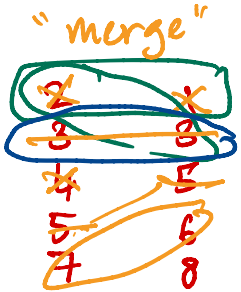
Naive list $\approx n^2$

Dictionary $\times n$

Sorting ?? $2 \times n \log n$ + $2 \times n$

2, 7, 5, 3, 4 ⇝ 2, 3, 4, 5, 7

8, 5, 6, 3, 1 ⇝ 1, 3, 5, 6, 8



$\leq n+n$ comparisons

[3, 5]