Lecture 7, 28 August 2025

Madhavan Mukund

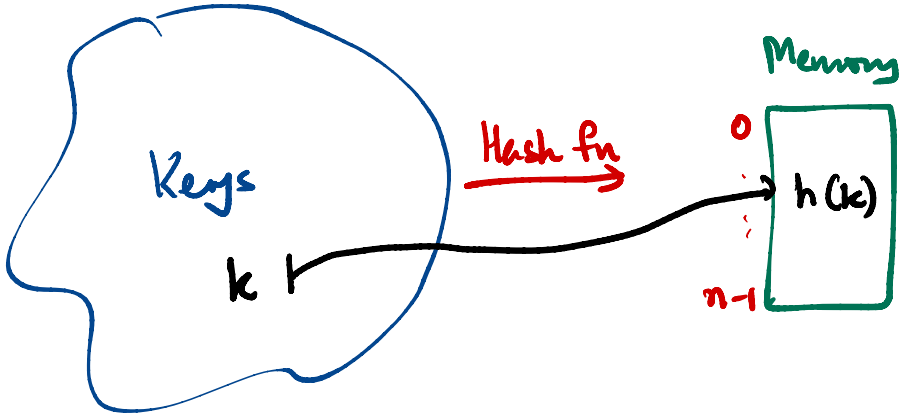
https://www.cmi.ac.in/~madhavan

Programming and Data Structures with Python

Dictionaries

$\{k_1 : v_1, k_2 : v_2, \ldots, k_n : v_n\}$

$d[k] = v$



Ignore the issue of collisions, how storage space is calculated ...

Ideally $h(k_1) = h(k_2) \Rightarrow k_1 = k_2$

Assume Computing $h(k)$ is constant time for any k

assign a value to $d[k]$

check if $d[k]$ current exists

Lists

x in l

↓

prop to len(l)

Dictionary

k in d (k in d.keys())

1

Const. time, by assumption

Duplicates

Keep only one copy of every value in a list

```
uniqlist = []
for v in l:
    if not(v in uniqlist):
        uniqlist.append(v)
```

Iteration in uniqlist

|| Implicitly a nested loop

No duplicates

$\Rightarrow (len(l))^2$

Slightly better

Sort l

5, 1, 2, 1, 3, 1, 2, 4 ⟶ 1, 1, 1, 2, 2, 3, 4, 5

check prev with current → single traversal

Cost

Sorting l = len(l) x log(len(l))

+ length l

1, 2, 3, 4, 5

Dictionary

Freq count → Extract keys

Insert each v into d → Extract keys

↓

Assume in size(d)

Prop to len(l)

List intersection

```
newL = []
for x in L1:
    for y in L2:
        if x == y and not (x in newL):
            newL.append(x)
```

Approx n^3

↑ Third iteration

Dictionary

```
list1d = {}
for x in l1:
    list1d[x] = 1
intersectd = {}
for y in l2:
    if y in list1d:
        intersectd[y] = 1
```

$\| n$

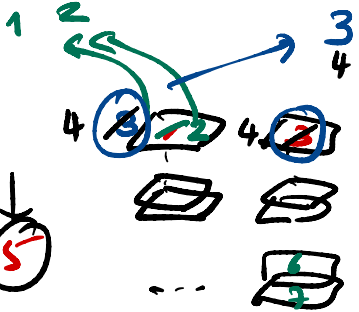
$\| n$

$\Longrightarrow$ prop. to n

Extract keys of intersectd

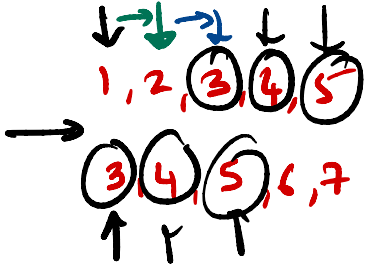
Classical soln

Sort l1
Sort l2
] Then what?



5, 3, 1, 2, 4

7, 5, 4, 3, 6



Time: Sort + sort + merge

Sort = $n \log n$

Merge? Input n, n Output n

$$2(n \log n) + 2n$$