Hashing - 4

Tuesday, 23 July 2024 8:44 PM

Agenda

→ Minimum Window Substring

→ LRU Cache

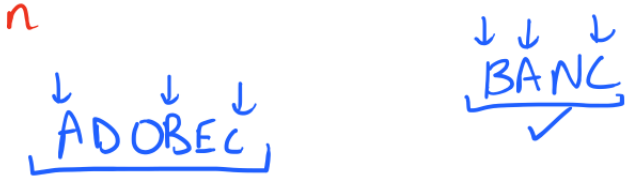
→ +1

---

1) Minimum Window Substring

s → ADOBE CODEBANC m

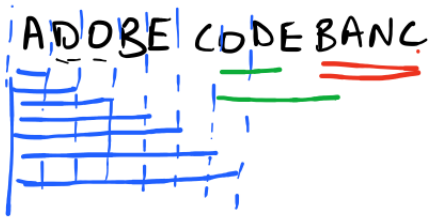
t → ABC n



1) Brute force

↳ generate all the possible substrings

of s' and compare with t.
↳ n



TC: O(m² × n)
↑ Substr ↑ comparison

SC: O(1)

2) DOBECODEBA
↳ exclude this.

↳ Be careful while choosing what to include

str s = "dbe acdbccab"
str t = "abbcdc"

✓ O(m)

f maps
d → 1 2 1 ∅ 0
b → 1 2 1 2
e → 1 0
a → 1 1
c → 1 2

✓ O(n)

f maps t
a → 1 ✓
b → 2 ✓
c → 2 ✓
d → 1

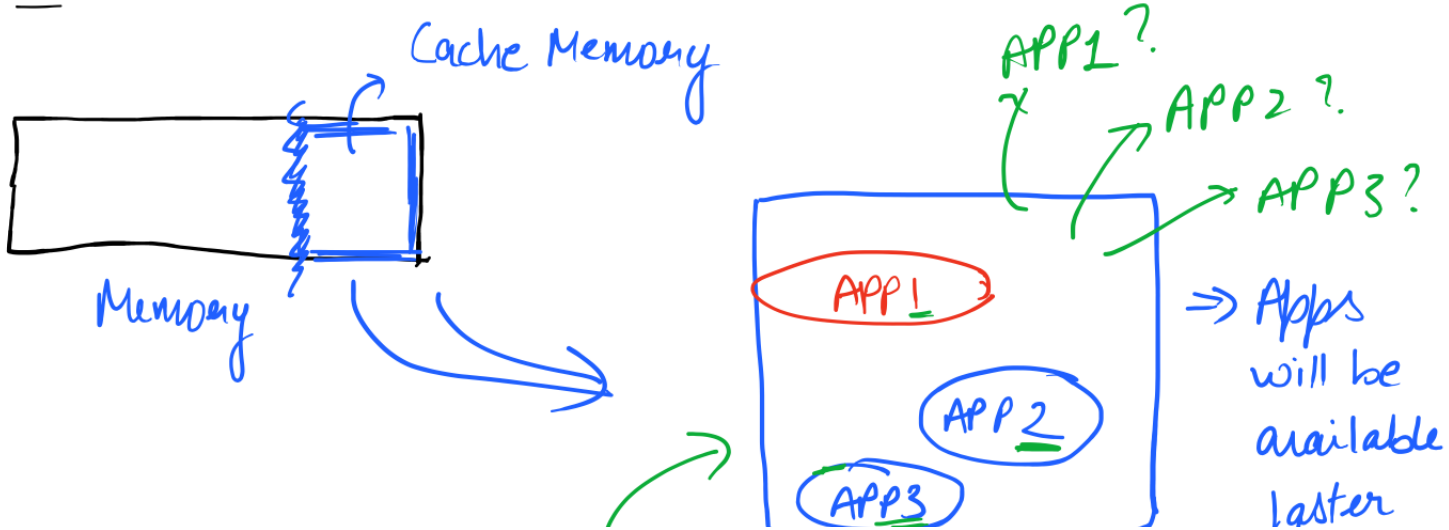
→ include all the alphabets until all is found then exclude the remaining

→ comparison → O(1)

TC: O(m + n)

SC: O(1)

LRU Cache



Algorithms for Cache Removal

APP1 : 5
APP2 : 15
APP3 : 4

LRU
Least Recently Used

LFU
Least Frequently Used

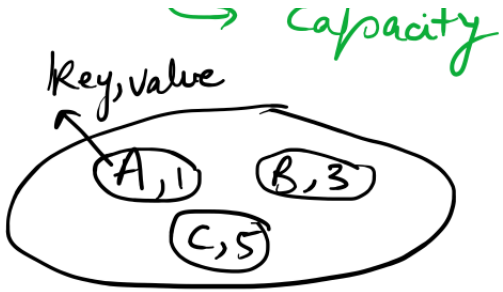
LRU cache

App 1
App 2
App 3
App 1
App 4

{3 App}

App 3
App 5 ✓

→ capacity



Recent Old

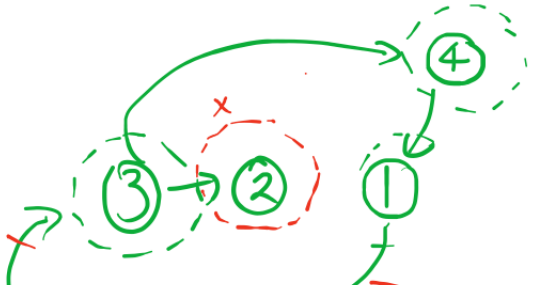

```
class LRUCache {
  // your code here
  public LRUCache(int capacity) {
    // your code here
  }

  public int get(int key) {
    // your code here
  }

  public void set(int key, int value) {
    // your code here
  }

}
```

→ capacity

fetch a particular app and make it most recently used

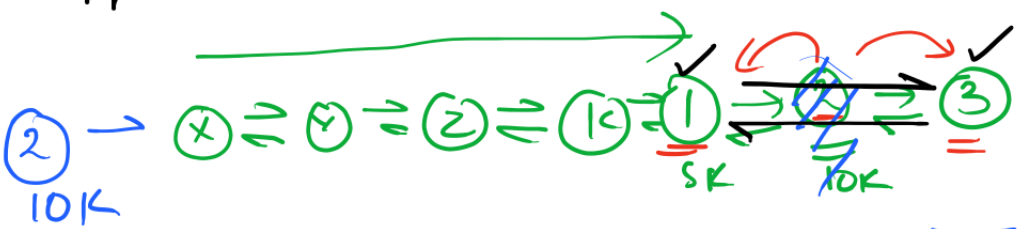
puts an app in the cache and makes it most recently used

App 1 ✓
App 2
App 3



App 1
App 4
App 3
App 5

Linked List ??



2	2 10K
1	1 5K

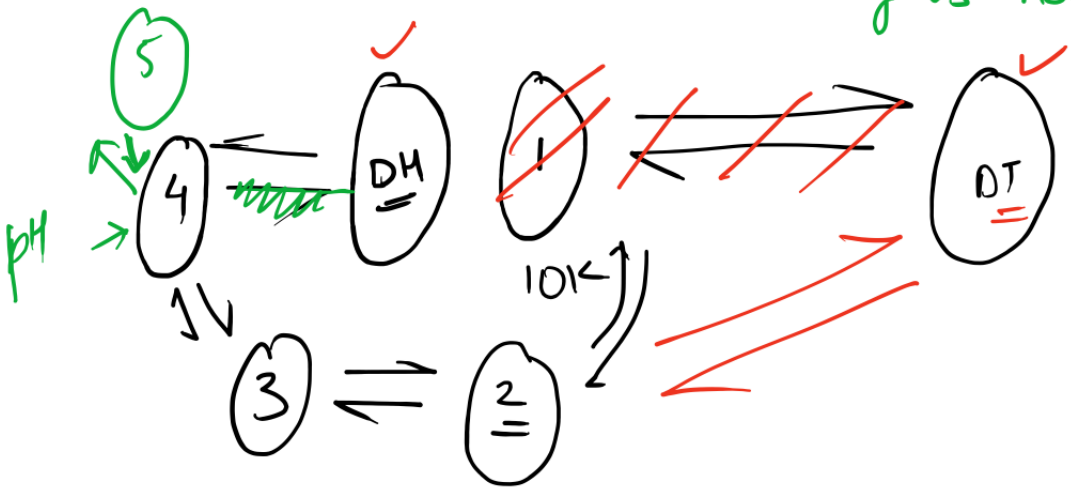
O(1)

Doubly Linked List + HashMap → To avoid the traversal

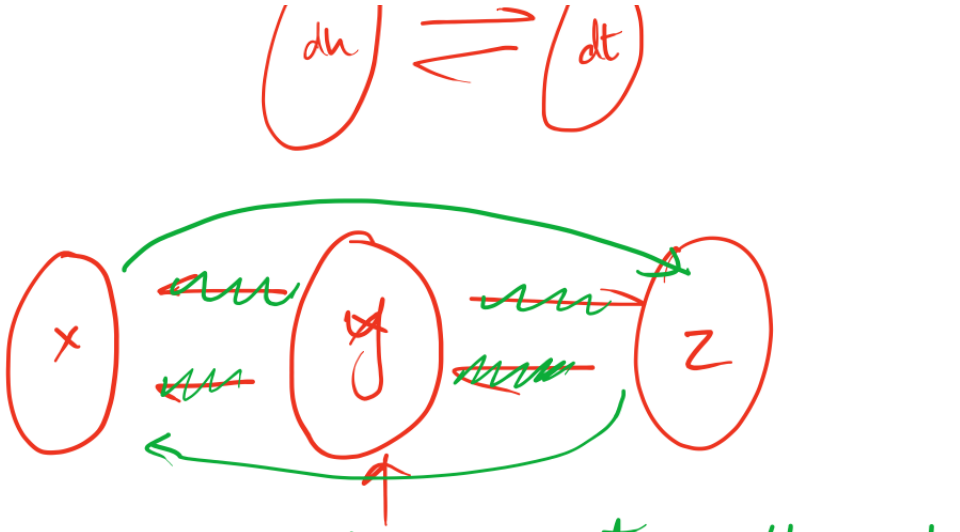
O(1)

Ease the Updation

Key Vs Address of DLLNode



~~1~~	~~10K~~
2	20K
3	30K



$y \cdot prev \cdot next = y \cdot next$

$y \cdot next \cdot prev = y \cdot prev$