

Indian Institute of Technology Madras
ZANZIBAR CAMPUS

Z5007 · PROGRAMMING AND DATA STRUCTURES

Doubly Linked Lists, Sentinels, Circular Lists and an LRU Cache

Lecture 14 · Friday 30 October 2026 · Canning, Broder & Lafore Ch 5

Dr. Innocent Nyalala | MTech Semester 1 | IIT Madras Zanzibar

Today's Plan

IIT Madras Zanzibar

1. (0–4 min) What a singly linked list cannot do cheaply
2. (4–15 min) Doubly linked lists: prev and next; remove and insert, every link drawn
3. (15–22 min) Sentinels: two dummy nodes that remove the special cases
4. (22–27 min) Circular lists: round-robin turns, and when to stop walking
5. (27–40 min) Application: an LRU cache with O(1) get and put
6. (40–50 min) Live demo, practice questions, your questions

Learning Outcomes

IIT Madras Zanzibar

By the end of this session, you will be able to:

Explain	what a prev reference makes possible, and what it costs
Order	the two link changes for remove and the four for insert so that no link is lost
Use	head and tail sentinels so that insert and remove need no special cases
Walk	a circular list correctly, and stop when you return to the start
Design	an LRU cache from a dictionary and a doubly linked list, with O(1) get and put

Key Terms for Today

IIT Madras Zanzibar

Term	Meaning	Example
Doubly linked list	each node has next AND prev references	A ⇄ B ⇄ C
prev	reference to the node before; None at the front (without sentinels)	B.prev is A
Sentinel (dummy node)	an extra node at an end that holds no data and is never removed	H ⇄ … ⇄ T
Invariant	a fact that is true after every operation	x.next.prev is x
Circular list	the last node points back to the first; there is no None	A → B → C → A
Cache	a small, fast store of recently used items	browser cache
Hit / miss	the item is / is not already in the cache	hit: no download
LRU (least recently used)	when full, remove the item unused for the longest time	evict the oldest

Doubly Linked Lists and an LRU Cache

IIT Madras Zanzibar

→ **What Singly Linked Lists Cannot Do**

Doubly Linked Lists

Sentinels: No More Special Cases

Circular Lists

Application: an LRU Cache

Demo and Practice

Three Things a Singly Linked List Cannot Do Cheaply

IIT Madras Zanzibar

1 **Remove the last node**

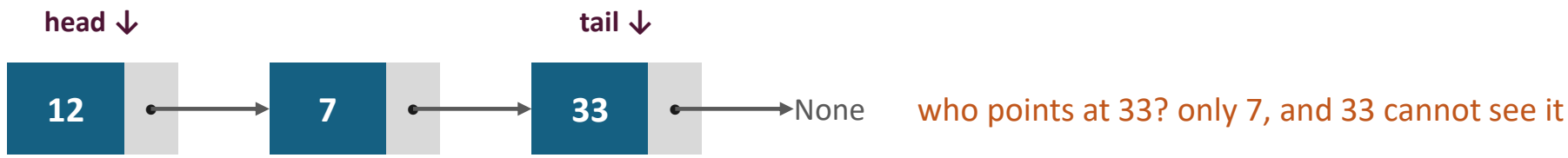
the new last node is the one BEFORE tail: walk from head, O(n)

2 **Go backwards**

'back' in a browser or 'previous track' needs a walk from the start

3 **Remove a node you hold**

the bypass needs its predecessor: walk from head, O(n)



who points at 33? only 7, and 33 cannot see it

? **Think:** what single change to the node fixes all three problems?

Doubly Linked Lists and an LRU Cache

IIT Madras Zanzibar

What Singly Linked Lists Cannot Do

→ **Doubly Linked Lists**

Sentinels: No More Special Cases

Circular Lists

Application: an LRU Cache

Demo and Practice

The Doubly Linked Node: prev, val, next

```
class DNode:
    def __init__(self, val):
        self.val = val
        self.prev = self.next = None
```



A chain: top arrows follow next, bottom arrows follow prev



A.prev and C.next are None (for now)

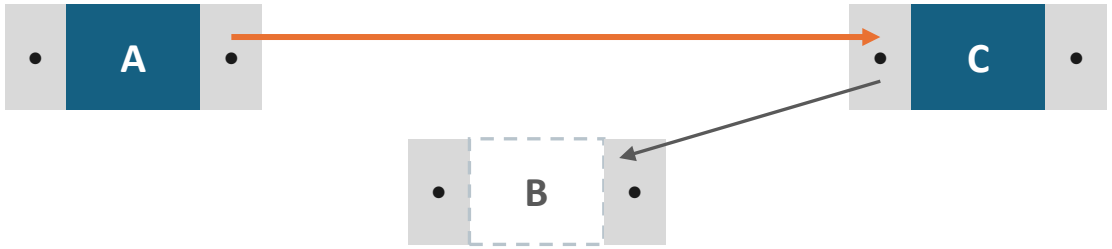
Definition: **Invariant:** for every node x with a next node, x.next.prev is x (and x.prev.next is x). Every operation must keep it true.

? **Think:** in A ⇄ B ⇄ C, what is B.next.prev? And A.next.next.prev?

Remove a Node You Hold: Two Link Changes, O(1)



1. B.prev.next = B.next (A now skips forward to C)



2. B.next.prev = B.prev (C now points back to A)



1. Step 1: A → C.
2. Step 2: A ← C.
3. No walk: B knows its neighbours.
4. Cost: O(1).

? **Think:** what goes wrong if B is the first node of the list?

Insert X Between P and Q: Four Link Changes, in a Safe Order



1. X.prev = P (new node first)
2. X.next = Q (new node first)
3. Q.prev = X (then the neighbours)
4. P.next = X (then the neighbours)

1. Fill in the new node first.
2. Then redirect the neighbours.
3. Hold P and Q in variables.
4. Cost: O(1).



P ⇄ X ⇄ Q

? **Think:** P.next = X first, then X.next = P.next. What is X.next?

The Problem: Special Cases Everywhere

IIT Madras Zanzibar

Situation	insert	remove
empty list	set head AND tail	—
at the front	new.prev = None; move head	move head; new head.prev = None
at the back	new.next = None; move tail	move tail; new tail.next = None
only node	—	head = tail = None
in the middle	the 4 link changes	the 2 link changes

Idea: if every real node always had a node before it and a node after it, every case would be the middle case.

? **Think:** how many if branches does the middle case need?

Doubly Linked Lists and an LRU Cache

IIT Madras Zanzibar

What Singly Linked Lists Cannot Do

Doubly Linked Lists

→ **Sentinels: No More Special Cases**

Circular Lists

Application: an LRU Cache

Demo and Practice

Sentinels: a Dummy Node at Each End

IIT Madras Zanzibar

empty



push_back(10)



push_back(30)



Definition: Sentinel: a dummy node that holds no data and is never removed. With H and T, every real node has a neighbour on both sides.

? **Think:** in the empty list, what are H.next and T.prev?

Every Operation Becomes One of Two

IIT Madras Zanzibar

Operation	With sentinels it is…	Cost
push_front(x)	insert x between H and H.next	O(1)
push_back(x)	insert x between T.prev and T	O(1)
pop_front()	remove H.next (if it is not T)	O(1)
pop_back()	remove T.prev (if it is not H)	O(1)
remove(node)	the two-step bypass	O(1)

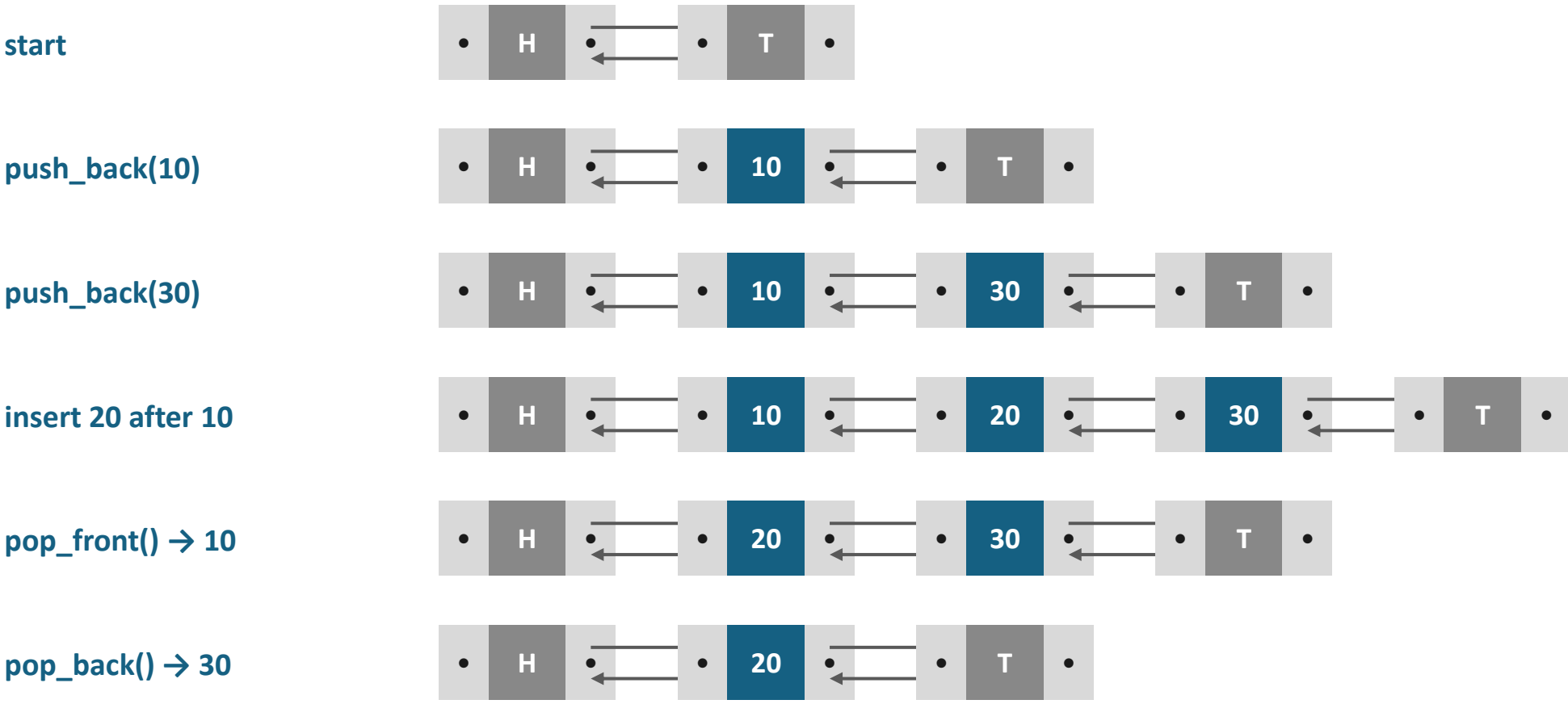
```
def insert_between(p, q, x):
    node = DNode(x)
    node.prev, node.next = p, q
    q.prev = node
    p.next = node

def remove(node):
    node.prev.next = node.next
    node.next.prev = node.prev
```

Result: pop_back is O(1): the weakness of the singly linked list is gone, and there are no end cases.

? **Think:** how do you test whether a sentinel list is empty?

Trace: Five Operations on a Sentinel List



? **Think:** what does one more pop_front() return, and what is left?

Testing a Linked Structure: Check the Invariant

```
def check(self):
    cur = self.head                # sentinel H
    while cur is not self.tail:
        assert cur.next.prev is cur
        cur = cur.next
    # also: walk back from T, same count
```

1. Forward printing can look right while prev links are wrong.
2. Assert x.next.prev is x at every node.
3. Run it after every operation on random tests.

Bug: insert X forgot Q.prev = X



forward: P → X → Q looks fine; but Q.prev is still P

? **Think:** which test catches the bug: printing forward, or the invariant check?

Cost Table: Array, Singly, Doubly

IIT Madras Zanzibar

Operation	Array (Python list)	Singly (head + tail)	Doubly + sentinels
get i-th item	O(1)	O(n)	O(n)
insert / remove at front	O(n)	O(1)	O(1)
insert at back	O(1) amortized	O(1)	O(1)
remove at back	O(1)	O(n)	O(1)
remove a node you hold	O(n)	O(n): needs prev	O(1)
memory per item	1 slot	+1 reference	+2 references

? **Think:** a 'recent files' menu: opening a file moves it to the top; keep 10 entries. Which structure, and why?

Doubly Linked Lists and an LRU Cache

IIT Madras Zanzibar

What Singly Linked Lists Cannot Do

Doubly Linked Lists

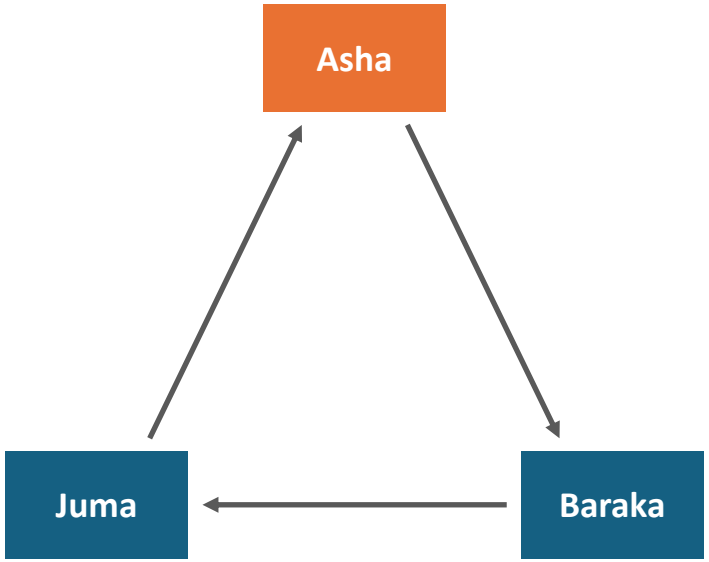
Sentinels: No More Special Cases

→ **Circular Lists**

Application: an LRU Cache

Demo and Practice

Circular Lists: Round-Robin Turns



Juma.next is Asha: no None

1 Turns in a game

players in a circle; next turn is cur = cur.next

2 Round-robin scheduling

the operating system gives each program a time slice, in a circle

3 Load balancing

requests go to servers 1, 2, 3, 1, 2, 3, …

? **Think:** Asha has turn 1. Whose turn is turn 7? What is the rule for turn t with 3 players?

Walking a Circle: When Do You Stop?

```
cur = start
while True:
    print(cur.val)
    cur = cur.next
    if cur is start:      # back where we began
        break
```

1. There is no None in a circle.
2. Stop when you return to the start node.
3. Empty circle: check start is None first.

Remove Baraka: prev.next = cur.next



Juma.next is still Asha: a circle of two

Think: what does while cur is not None do on a circular list?

Doubly Linked Lists and an LRU Cache

IIT Madras Zanzibar

What Singly Linked Lists Cannot Do

Doubly Linked Lists

Sentinels: No More Special Cases

Circular Lists

→ **Application: an LRU Cache**

Demo and Practice

What Is a Cache, and What Is LRU?

IIT Madras Zanzibar

1 Browser cache

recent images and pages, so a revisit is fast

2 CPU cache

recent memory, a few nanoseconds away

3 Database buffer pool

recent disk pages kept in memory

4 Phone app switcher

recent apps first; the oldest are closed

Definition: LRU (least recently used): when the cache is full, remove the item whose last use is the oldest.

1. Hit: the item is in the cache: fast.
2. Miss: fetch it (slow), store it, maybe evict one.

Think: in the phone app switcher, which app is the least recently used?

An LRU Trace: Capacity 3, Requests A B C A D B E

request	result	least recent → most recent
A	MISS	A
B	MISS	A B
C	MISS	A B C
A	HIT	B C A
D	MISS, evict B	C A D
B	MISS, evict C	A D B
E	MISS, evict A	D B E

What the cache must do

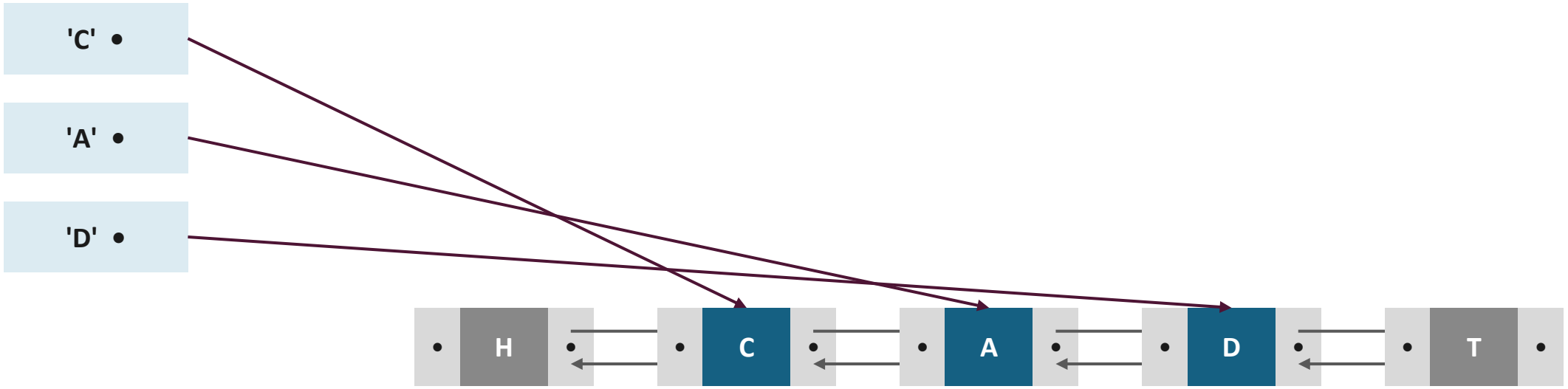
1. Every request: FIND the item.
2. Then MOVE it to the most-recent end.
3. When full: REMOVE the least recent.
4. All three must be O(1).

? **Think:** on the D row, why is B evicted and not A?

The Design: a Dictionary plus a Doubly Linked List

IIT Madras Zanzibar

dictionary: key → node **doubly linked list: least recent (next to H) → most recent (next to T)**



1. Find: dictionary, O(1). Move to the recent end: remove + insert before T, O(1). Evict: H.next, O(1).

? **Think:** why must the list be DOUBLY linked?

get(key) and put(key, value), Step by Step

IIT Madras Zanzibar

1 **get(key)**

1. node = map.get(key); if None: miss, return None
2. remove(node); insert it before T (now most recent)
3. return node.val

2 **put(key, value)**

1. key present: update value, move node before T, done
2. full: old = H.next; remove(old); del map[old.key]
3. node = new node; insert before T; map[key] = node

get('A') on H ⇄ C ⇄ A ⇄ D ⇄ T:



A moved next to T: it is now the most recent

? **Think:** why does each node store its KEY as well as its value?

In Practice: Python's Built-ins and How Much a Cache Helps

IIT Madras Zanzibar

```
from functools import lru_cache

@lru_cache(maxsize=128)
def exchange_rate(day):
    ...                 # slow web request
# second call with the same day: instant
```

Capacity	Hit rate (demo)
10	15.0%
100	43.4%
1,000	74.8%

1. OrderedDict.move_to_end(key) and popitem(last=False) are the same two linked-list moves.
2. Hit rate grows with capacity, but each extra slot buys less: diminishing returns.

? **Think:** why does the first slot of the cache help the most?

Doubly Linked Lists and an LRU Cache

IIT Madras Zanzibar

What Singly Linked Lists Cannot Do

Doubly Linked Lists

Sentinels: No More Special Cases

Circular Lists

Application: an LRU Cache

→ **Demo and Practice**

Live Demo

IIT Madras Zanzibar

```
cd ~/PDS/Class_Demos/Week_04
python3 W04_Fri_Oct30_doubly_circular_list.py
python3 W04_Fri_Oct30_lru_cache.py                  # capacity 3: A B C A D B E
python3 W04_Fri_Oct30_lru_cache.py 2 1 2 1 3 2 1    # your own capacity and requests
python3 W04_Fri_Oct30_lru_cache.py hitrate
```

What to watch

1. Compare the four printed link updates with slide 10.
2. Compare the LRU order after each request with slide 23.

In-Class Exercise (6 minutes, in pairs)

IIT Madras Zanzibar

IN-CLASS TASK: draw first, then answer

1. Sentinel list H ⇄ 10 ⇄ 30 ⇄ T. Insert 20 between 10 and 30: write the four link changes in a safe order.

2. In the same list, remove the node 30 you hold. Which two links change? What are T.prev and 20.next afterwards?

3. An insert set X.prev, X.next and P.next but forgot Q.prev = X. Printing forward looks right. What does walking BACKWARD from T print?

4. Circular list Asha → Baraka → Juma → Asha. Asha has turn 1. Baraka leaves after turn 4. Whose turn is turn 8?

5. LRU cache, capacity 2, requests 1 2 1 3 2 1. Mark each HIT or MISS, name every eviction, and give the final order.

Solution

IIT Madras Zanzibar

1

1: X.prev = 10; X.next = 30; 30.prev = X; 10.next = X

new node first, then the neighbours: H ⇄ 10 ⇄ 20 ⇄ 30 ⇄ T.

2

2: 20.next = T and T.prev = 20

the two-step bypass; no special case because T is a sentinel.

3

3: backward prints Q, P

Q.prev was never updated, so X is invisible going backward: the invariant check catches it.

4

4: Asha

turns: Asha, Baraka, Juma, Asha; then the circle is Asha, Juma: Juma 5, Asha 6, Juma 7, Asha 8.

5

5: M M H M(evict 2) M(evict 1) M(evict 3)

one hit, five misses; final order, least to most recent: 2, 1.

Common Mistakes

IIT Madras Zanzibar

1 **Neighbours first**

set the new node's prev and next first, then the neighbours

2 **A missing link**

forward print looks fine; check x.next.prev is x

3 **Waiting for None in a circle**

stop when you come back to the start node

4 **LRU: dict not updated**

on eviction also del map[old.key]; the node stores its key for this

Summary and Exam Checkpoints

IIT Madras Zanzibar

Doubly linked

prev and next: remove a node you hold and pop_back are O(1)

Sentinels

H and T are never removed; every operation is insert-between or remove

Circular

no None; stop when you return to the start; round-robin turns

LRU cache

dict key → node + doubly linked list by recency: get and put O(1)

Next

Mon 2 Nov: Thinking recursively: base cases, the call stack, Towers of Hanoi

Wed 4 Nov: Merge sort: divide and conquer

Demo code: Class_Demos/Week_04/W04_Fri_Oct30_doubly_circular_list.py and ..._lru_cache.py

References and Further Reading

[1] Canning, J., Broder, A. & Lafore, R. (2022). Data Structures & Algorithms in Python. Addison-Wesley. Chapter 5 (doubly linked and circular lists).

[2] Cormen, T. H., Leiserson, C. E., Rivest, R. L. & Stein, C. (2022). Introduction to Algorithms, 4th ed. MIT Press. Section 10.2 (doubly linked lists with sentinels).

[3] Silberschatz, A., Galvin, P. B. & Gagne, G. (2018). Operating System Concepts, 10th ed. Wiley. Section 5.3 (round-robin scheduling) and 10.4 (LRU page replacement).

[4] Python documentation: functools.lru_cache; collections.OrderedDict.