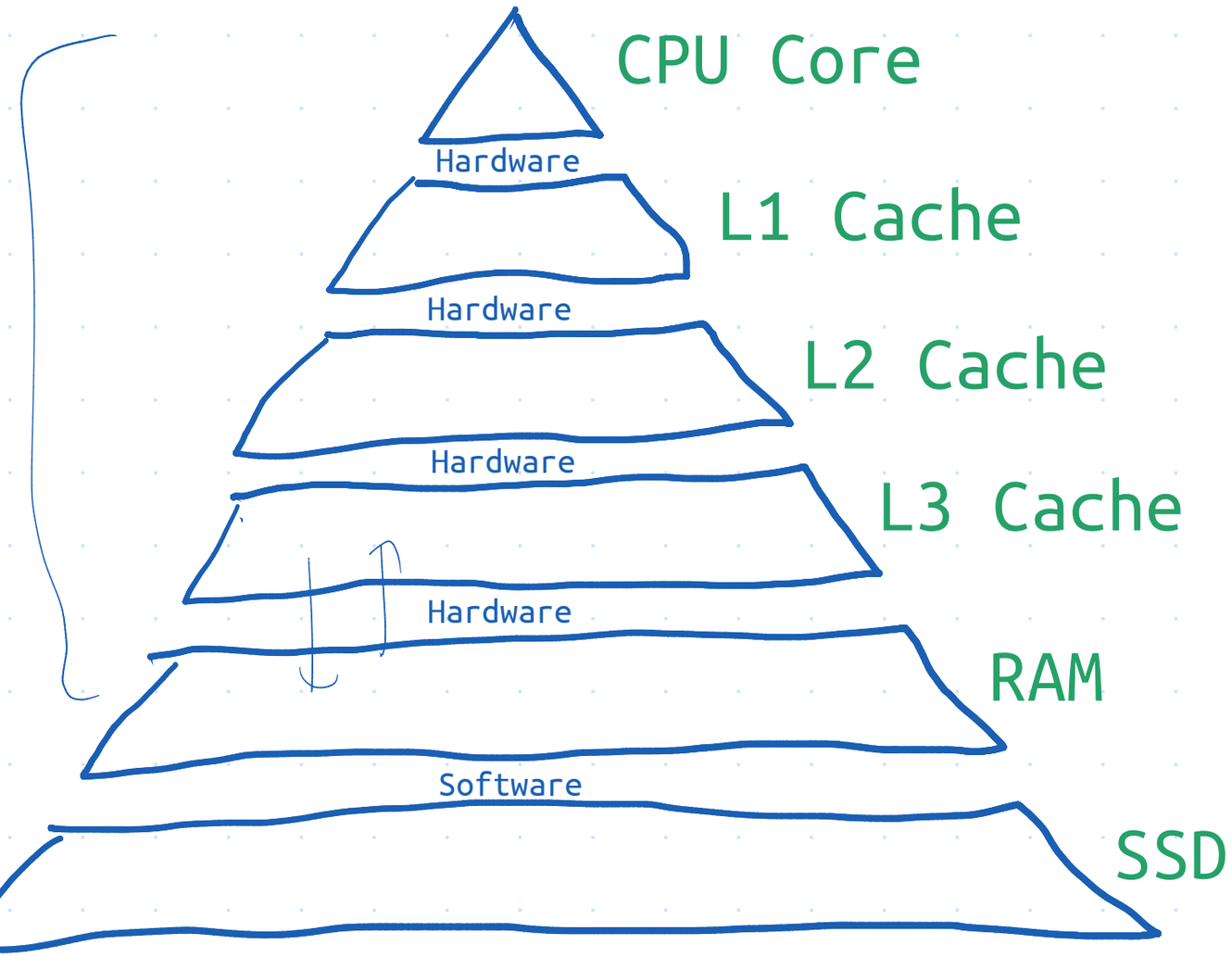
CSE 350

Advanced Data Structures

Topic 6: The Buffer Manager □□ & Virtual Memory □

~ fast



Slow
Limited BW
persistent

Buffer Manager Responsibilities

Get(address) -> data (Access to data)

Pin/Unpin (Tracking Usage)

↳ Pin(address) -> frame

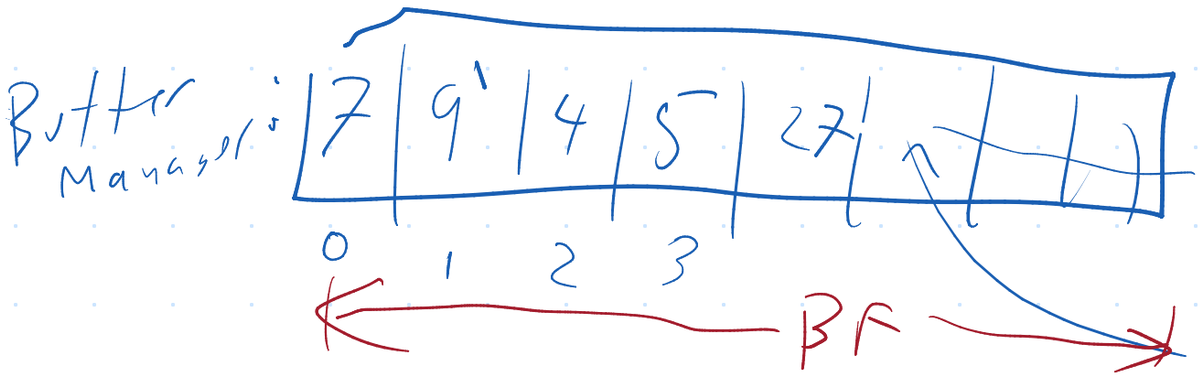
Unpin(frame)

Flush (Guaranteeing Persistance)

flush(frame)

flushAll()

Cache Replacement (Freeing up space)



Get (Pseudocode)

```
get(page)
  if page not in Buffer:
    if no free frame:
      F=replace_cache_policy()
      if F has edits: "frame is dirty"
        write F back to disk (flush)
    else:
      F=next free frame()
    read p into F
  Pin(F)
  return F
```

Major Costs

Data Structure

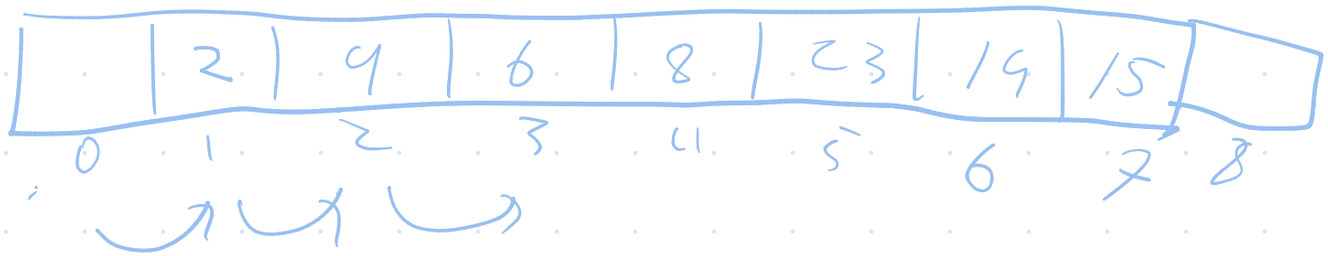
checking each frame

O(BF)

cost?

one write
one read
) 70!!

Fixed cost



Free: Linked List

Existing: Map (Hash, Tree) from page to index

Cache Replacement Policies

Goals

- Minimize cache miss rate
  - historical data about usage
  - spatial/temporal locality
- Minimize resources
  - CPU
  - Memory

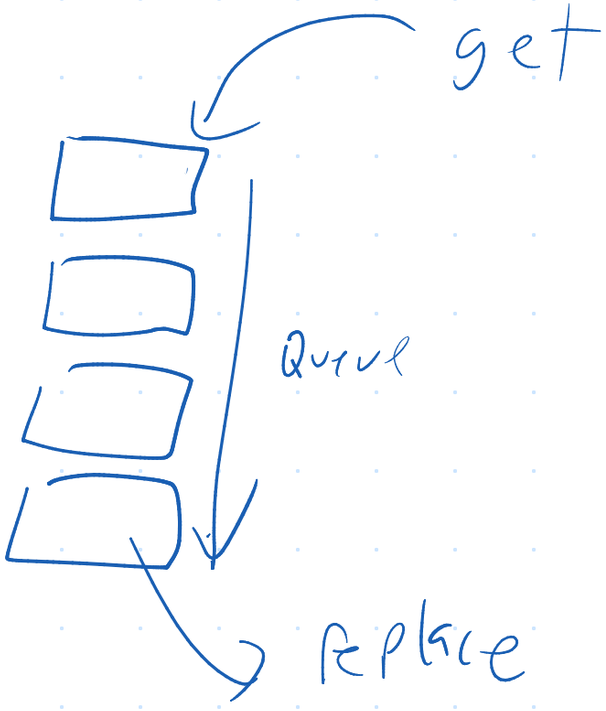
Techniques

- Random

"high" Miss rate
minimal compute
no state

- FIFO
- Least Recently Used
- Second Chance

FIFO

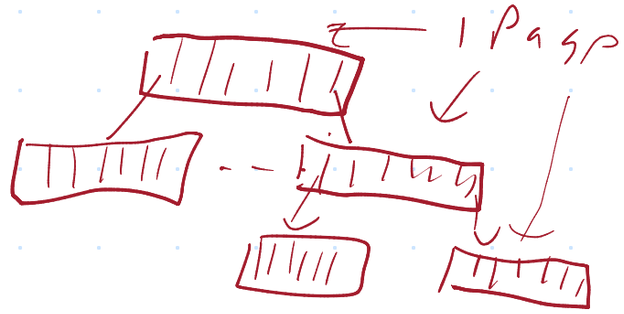


oldest frame
gets replaced first

Pro
O(1) compute
Better than random
↳ some temporal locality

Con
O(BF) memory

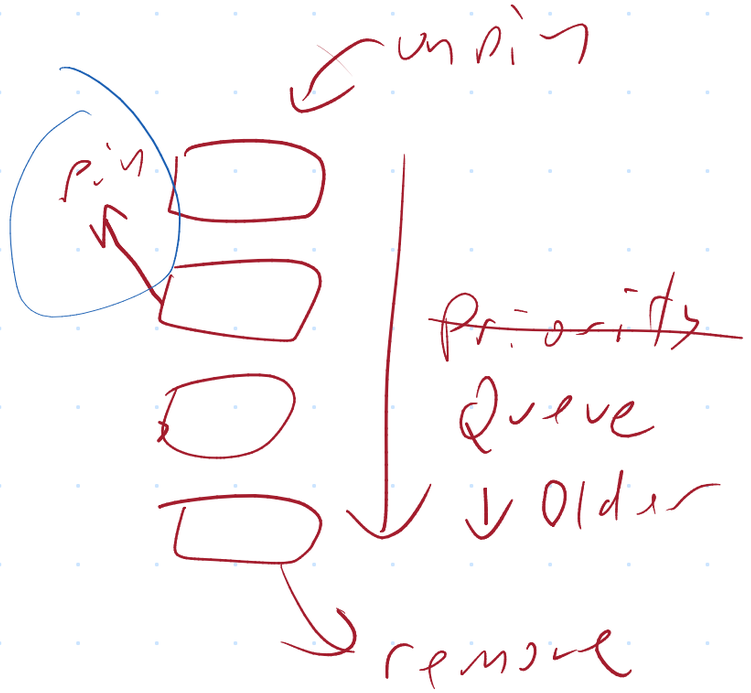
Example
B+ Tree:



get(root)
get(p1)
get(p2)

Case 1: BF < size of tree
Case 2: BF > size of tree

LRU



pro

O(1) compute (if O(1) remove)

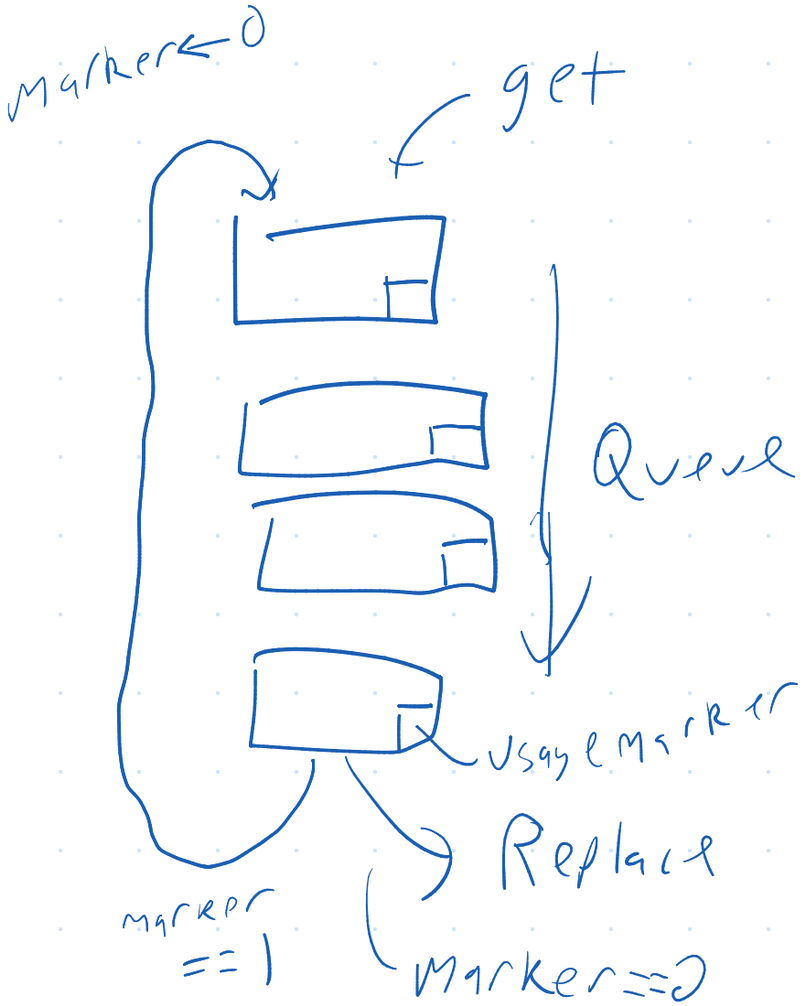
Respects temporal locality within BF calls to get

con

O(BF) Mem

More complex than fifo
↳ more compute
↳ more mem

Second Chance



when replacing

while queue.peek().marker=1

queue.push(

queue.pop with marker=0

)

return queue.pop()

unpin sets marker = 1

Pro

simpler

captures locality for freq pages

less mem than LRU

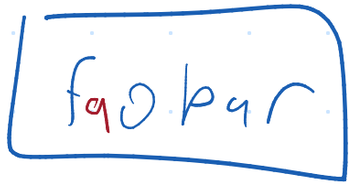
Con

slower (slightly)

more mem than FIFO

Writeback Challenges

- Can't write a page being modified (a pinned page)
- Need to guarantee that the page is written fully, or not at all



Virtual Memory

Virtual Addr	Physical Addr
68	87
17	3
19	2
93	42
86	7

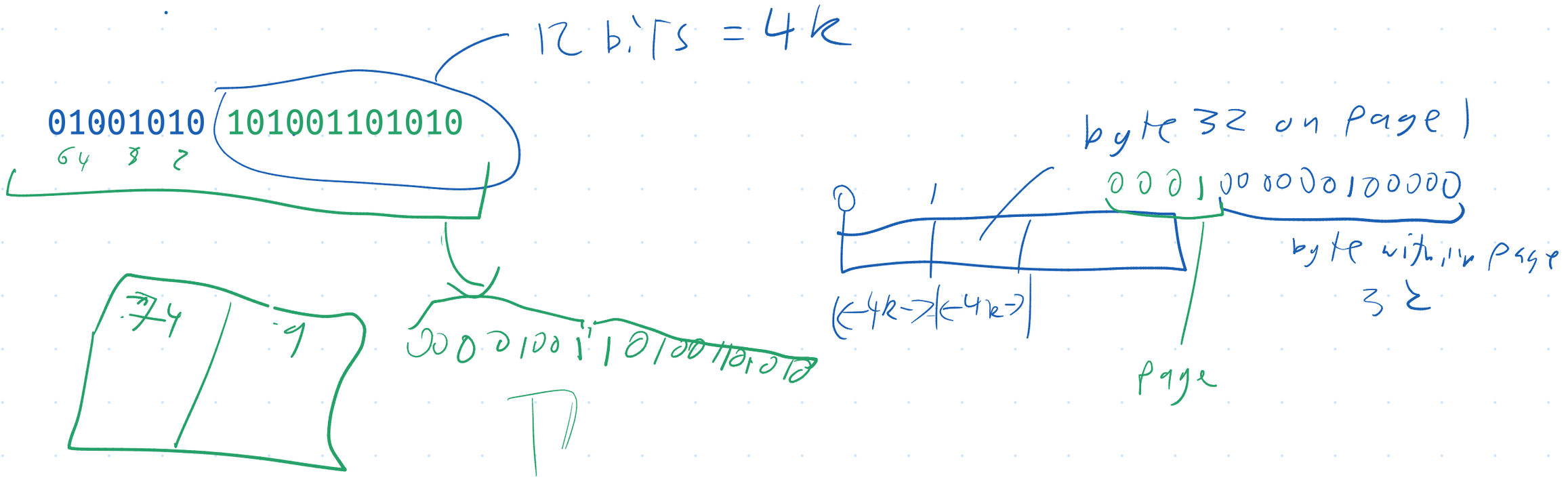
0101 0110

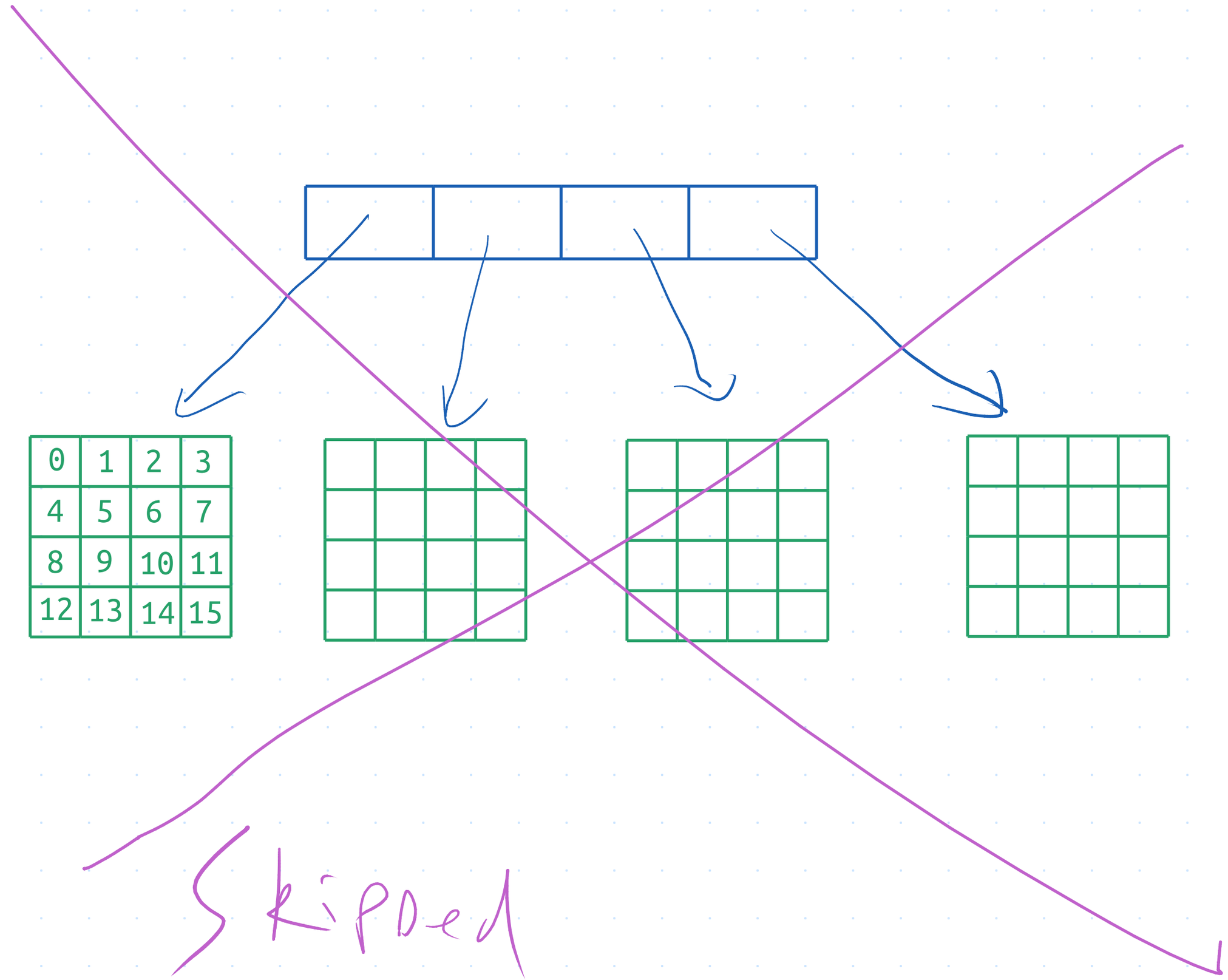
data on page 86

↓

data in frame 7

Prefix Maps

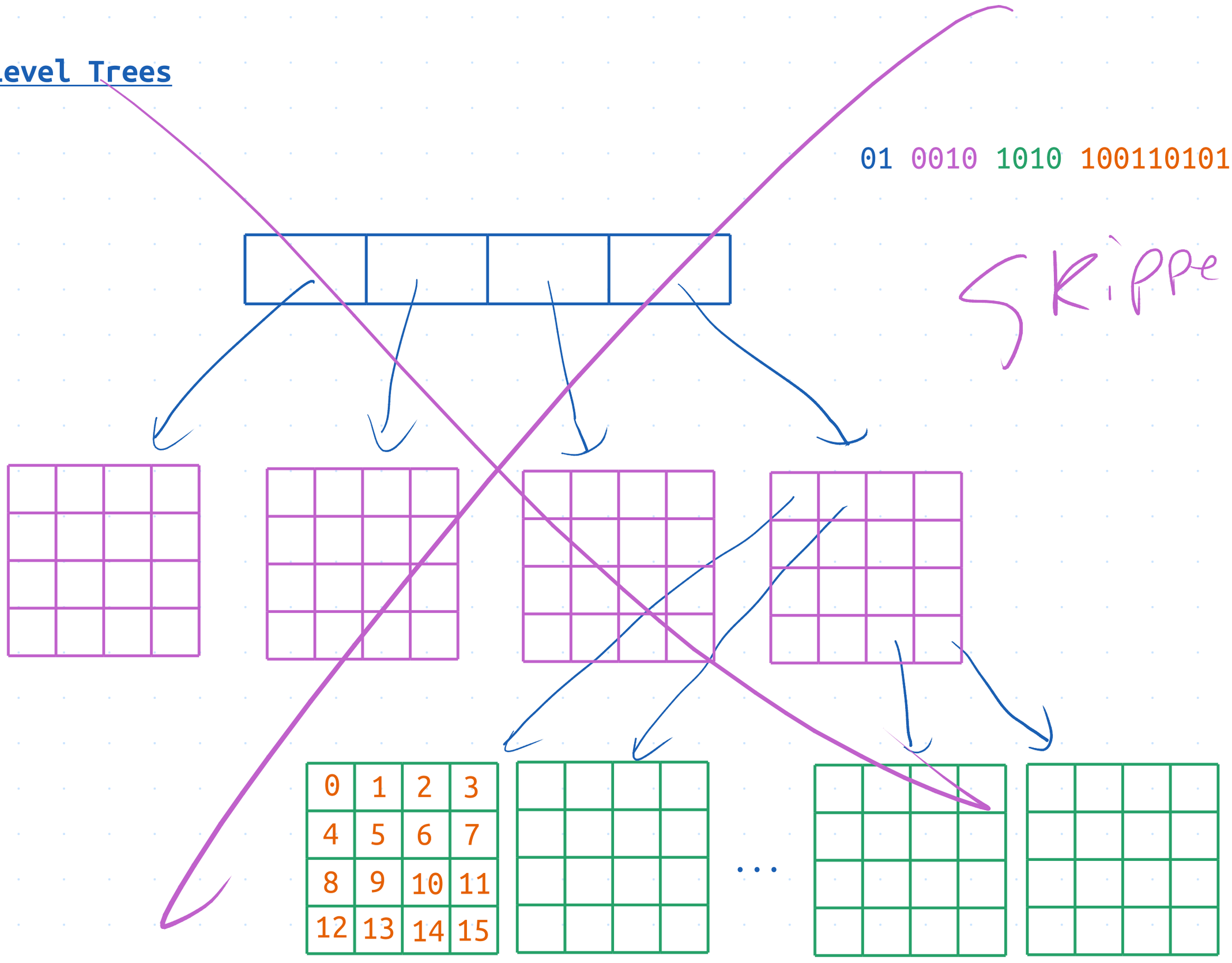

12 bits = 4k
01001010 101001101010
64 8 2
byte 32 on page 1
0001000000100000
byte within page
32
page


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Skipped

Multi-Level Trees

01 0010 1010 100110101001

skipped



Prefix Addressing