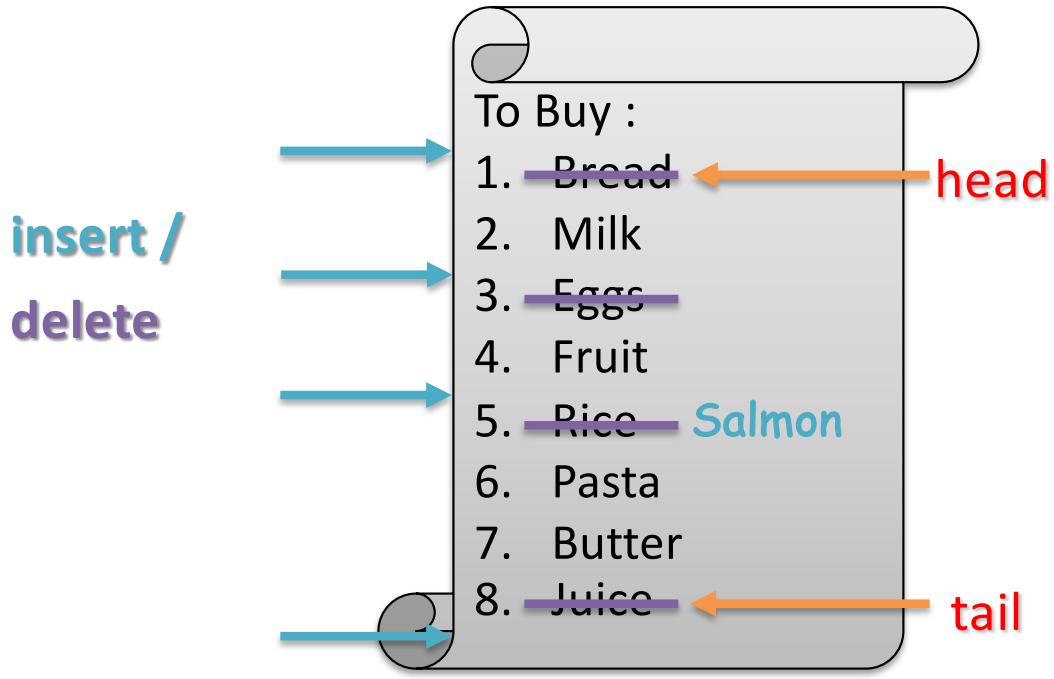
Linked List

List

List

http://www.gograph.com/vector-clip-art/queue.html

List



Already bought
Eggs, Rice, Bread, Juice

Oh I forgot
Salmon

Ordered List – Unordered List

To Buy :

1. Bread
2. Milk
3. Eggs
4. Fruit
5. Rice
6. Pasta
7. Butter
8. Juice

Unordered List

Scores :

1.	Bruce	2
2.	Tom	3
3.	Ben	5
4.	Max	7
5.	Tim	7
6.	Marry	8
7.	Ron	9
8.	Harry	10

Ordered List
Ascending Order

$98n^5 - 4n^4 + n^3 - 8n^2 + 5n + 7$

Ordered List
Decending Order

Logical Abstract Data Type & Implementation

Logical ADT : ขึ้นกับ application

1. Data : ของมีลำดับ มีปลาย หัว head และ/หรือ ท้าย tail

2. Methods :ขึ้นกับ และ list เป็น ordered list หรือไม่

Data Implementation ?

Python List

Unordered List / Ordered List

•List()	สร้าง empty list	
•isEmpty()	returns boolean ว่า empty list หรือไม่	
•size()	returns จำนวนของใน list	
•search(item)	returns ว่ามี item ใน list หรือไม่	
•index(item)	returns index ของ item	กำหนดให้ item อยู่ใน list

unordered list

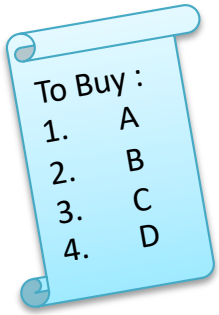
•append(item)	adds item ท้าย list ไม่ Returns	คิดว่า item ไม่มีอยู่ก่อนใน list
•insert(pos,item)	adds item ที่ index pos ไม่ Returns	คิดว่า item ไม่มีอยู่ก่อนใน list

ordered list

add(item)	adds item เข้า list ตามลำดับ ไม่ Returns	

•remove(item)	removes & return item	คิดว่า item มีอยู่ใน list
•pop()	removes & return item ตัวสุดท้าย	list _size >= 1
•pop(pos)	removes & return item ตัวที่ index = pos	list _size >= 1

List Implementation : C Sequential (Implicit) Array, Python List



A B C D

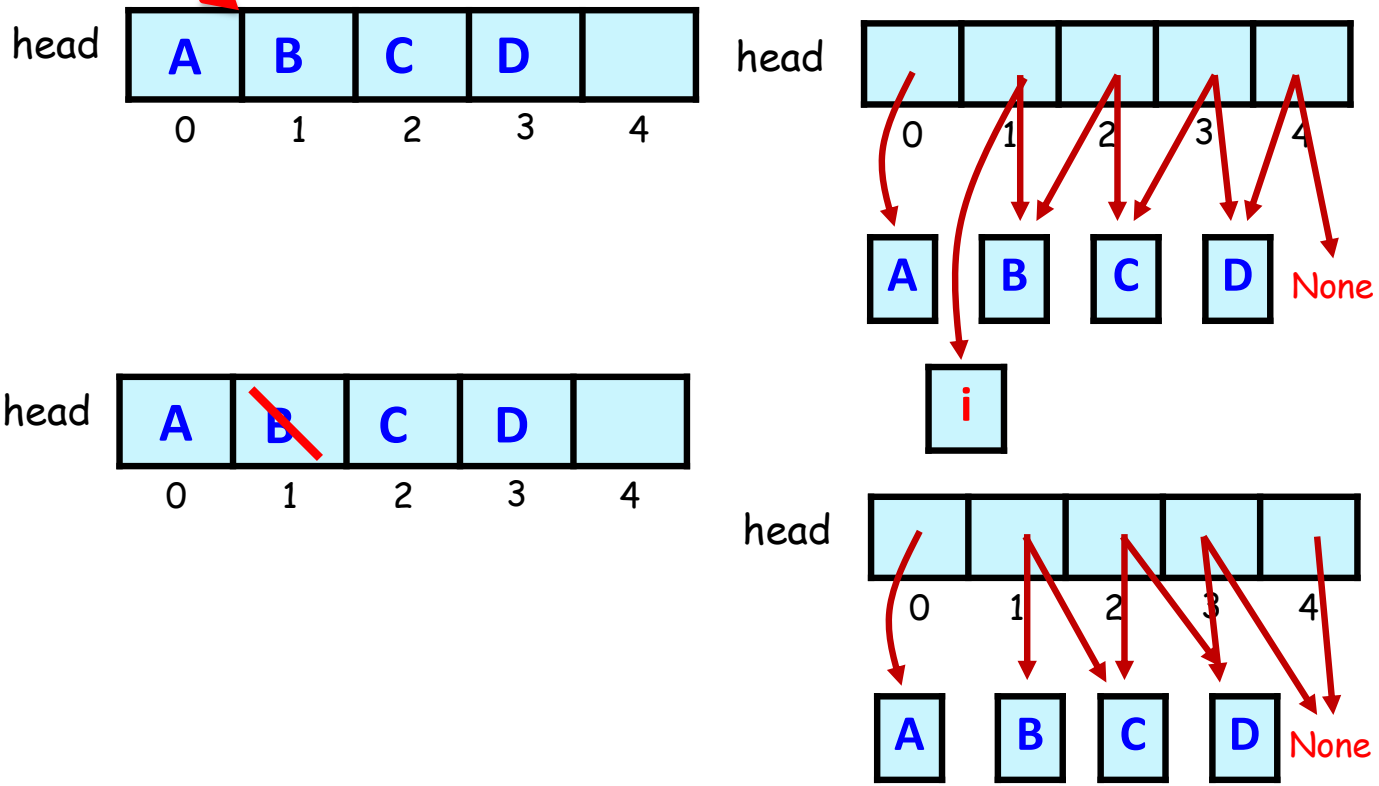
Problem : fix positions

i

C : Sequential Array

Python : List

insert i ? : shift out



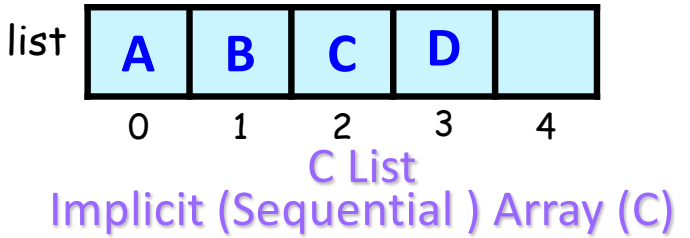
delete : shift in

Linked List

Problem : Fix Positions | List

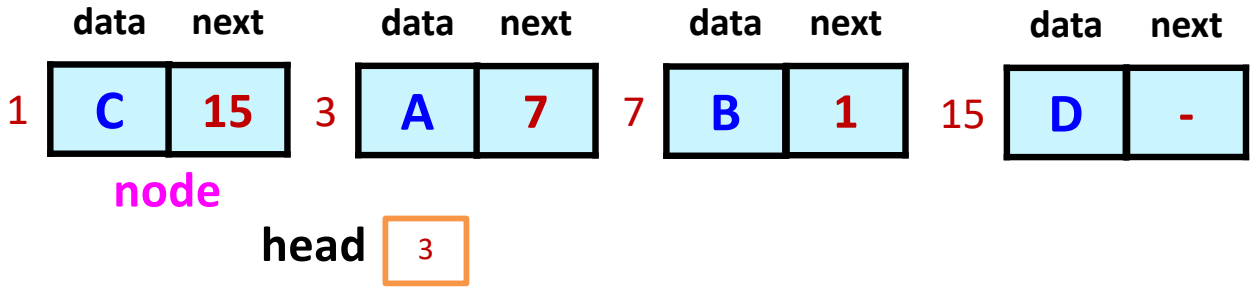
Unfix Positions | Linked List

ลำดับ order?



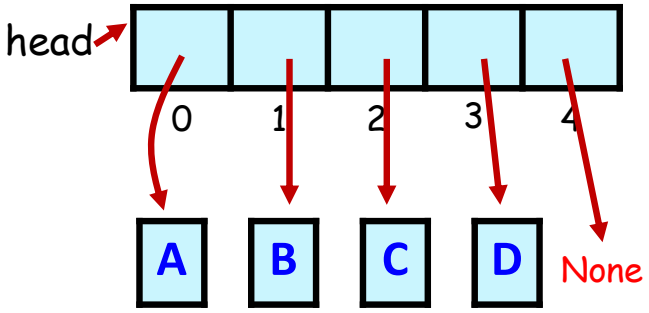
C List

Implicit (Sequential) Array (C)

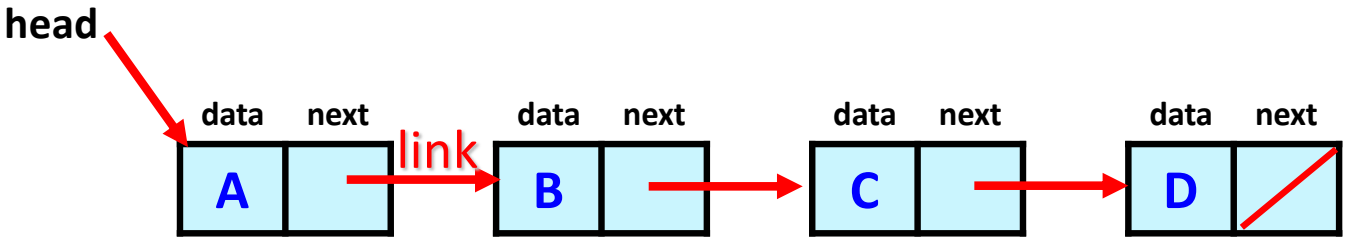


node

head 3



Python : List



node

Logical linked list

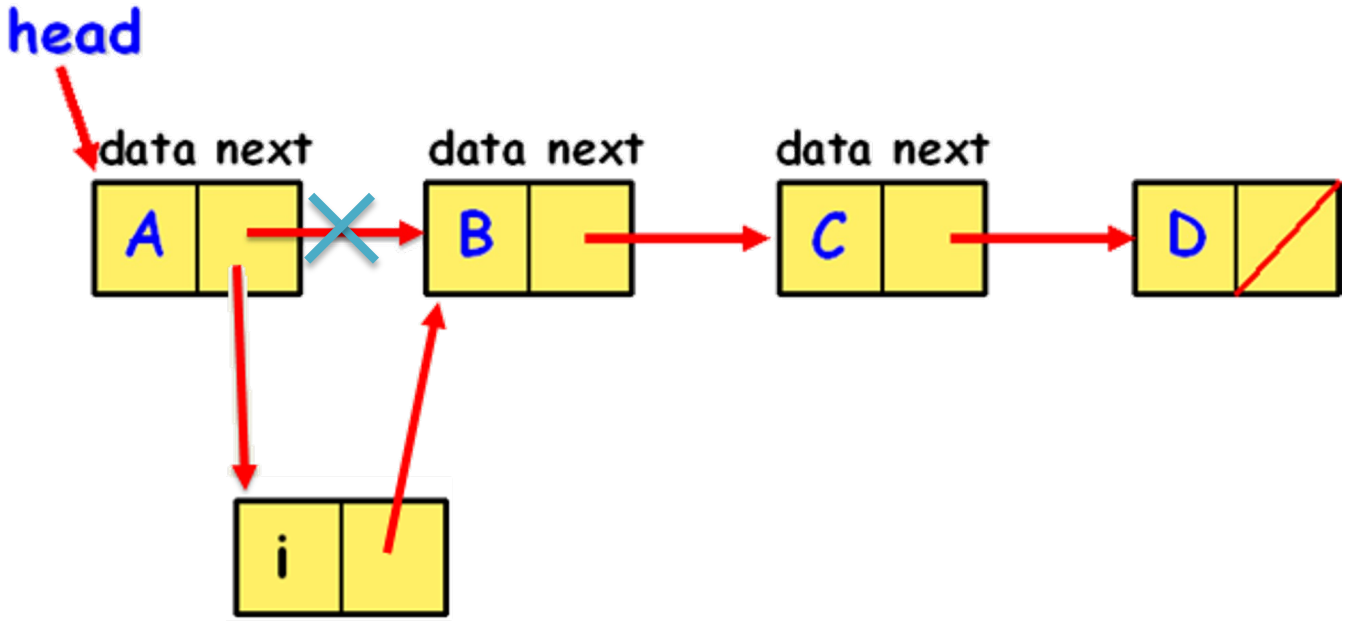
Logical คือในความคิดของเรา

เช่น link แทนด้วยลูกศร แทนการเชื่อมโยงกัน

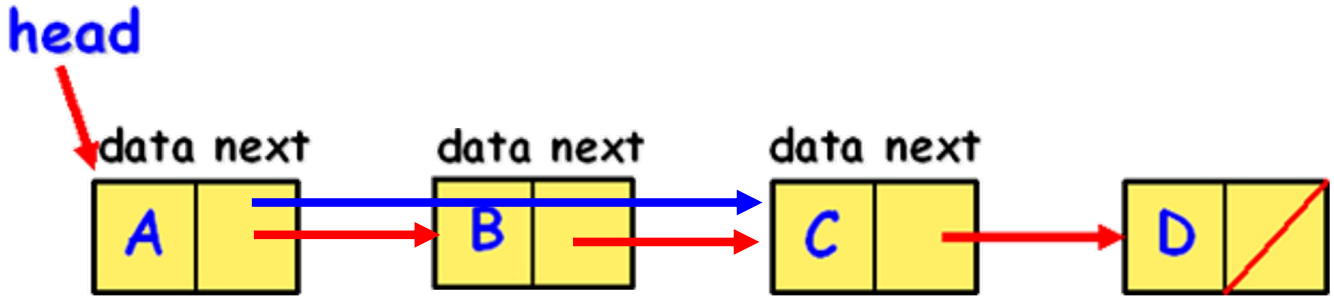
physical (implementation) โครงสร้างที่ใช้ในการสร้างจริง

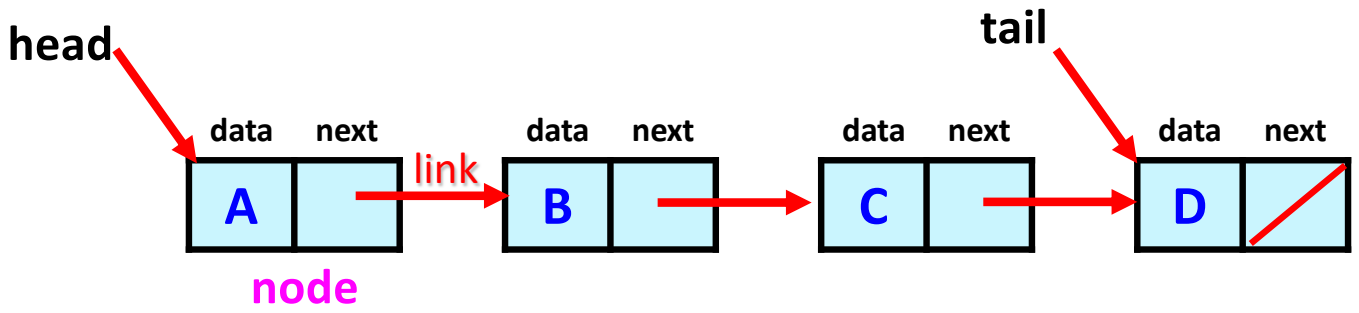
เช่น link อาจใช้ pointer หรือ index ของ array

Solve Inserting Expensive Shifting Problem



Solve Deleting Expensive Shifting Problem





Linked List

Data :

1. data
2. next (link)

} node class

3. head
4. tail ?

} list class

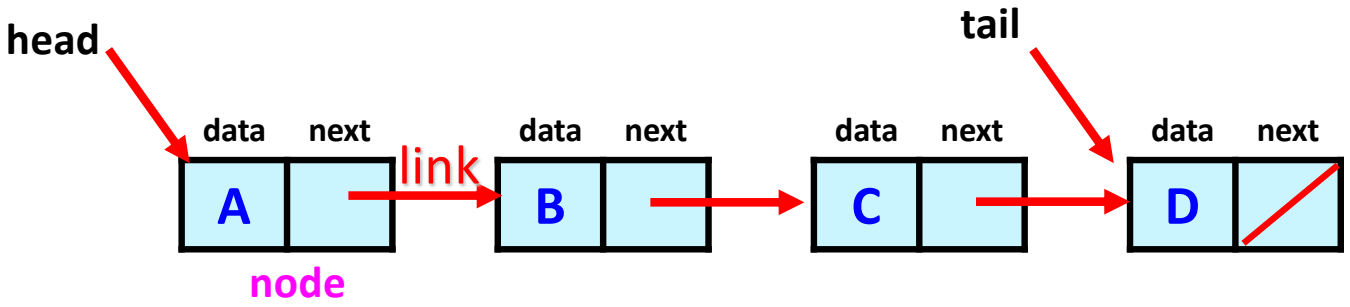
Node Class / List Class

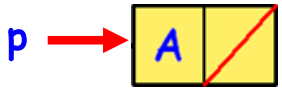
1. Data :

`__init__()` : **constructor** ให้ค่าตั้งต้น

2 underscores 2 underscores

Node Class






```
p =node('A', None)
```

```
class node:

    def __init__(self,  data,  next = None):

            self.data = data

            if next is None:
                  self.next = None
            else:
                  self.next = next


    def __str__(self):
            return str(self.data)
```

List Class

```
class list:
```

```
""" unordered singly linked list
    with head
"""
def __init__(self):
    self.head = None
```

```
l1 = list()
```

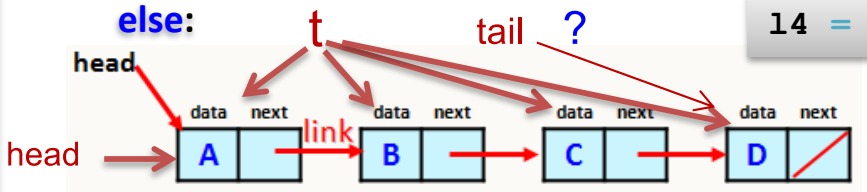
```
""" unordered singly linked list
    with head & tail
"""
def __init__(self):
    self.head = self.tail = None
```

```
l2 = list()
```

```
def __init__(self, head = None):
    """ unordered singly linked list
        can set default list
        with head, tail & size
    """
    if head == None:
        self.head = self.tail = None
        self.size = 0
    else:
        self.head = head
        t = self.head
        self.size = 1
        while t.next != None:   # locating tail & find size
            t = t.next
            self.size += 1
        self.tail = t
```

head
tail → None

```
l3 = list()
```

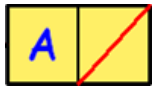



```
l4 = list(head)
```

Methods

1. __init__() : ให้ค่าตั้งต้น
2. size():
3. isEmpty():
4. append () : add at the end
5. __str__():
6. addHead() : ให้ค่าตั้งต้น
7. remove(item):
8. removeTail():
9. removeHead() :
10. isIn(item): / search(item)
11. . . .

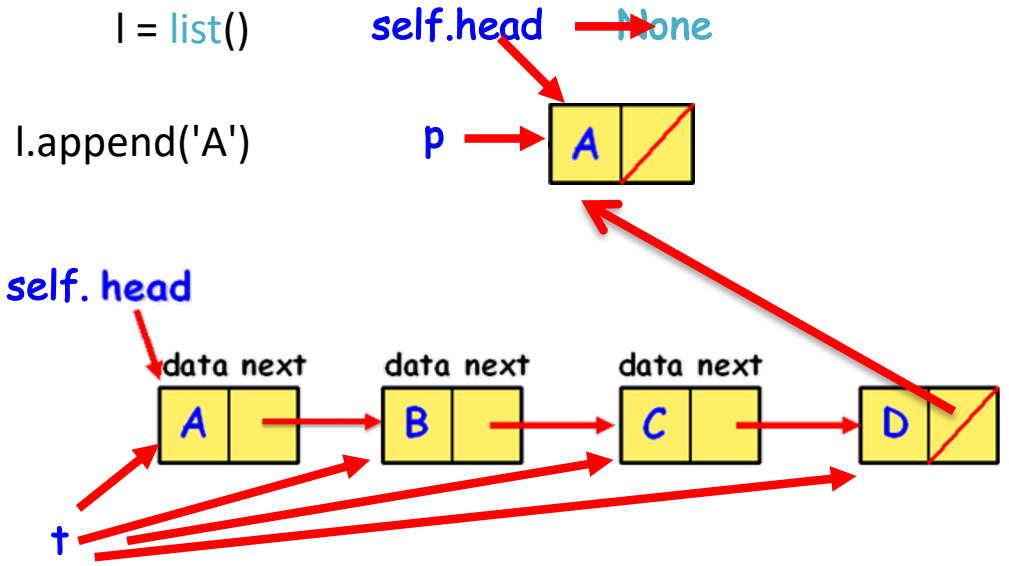
Creating a List



node('A', None)

```
class node:
    def __init__(self, data, next = None):
        self.data = data
        if next == None:
            self.next = None
        else:
            self.next = next
```

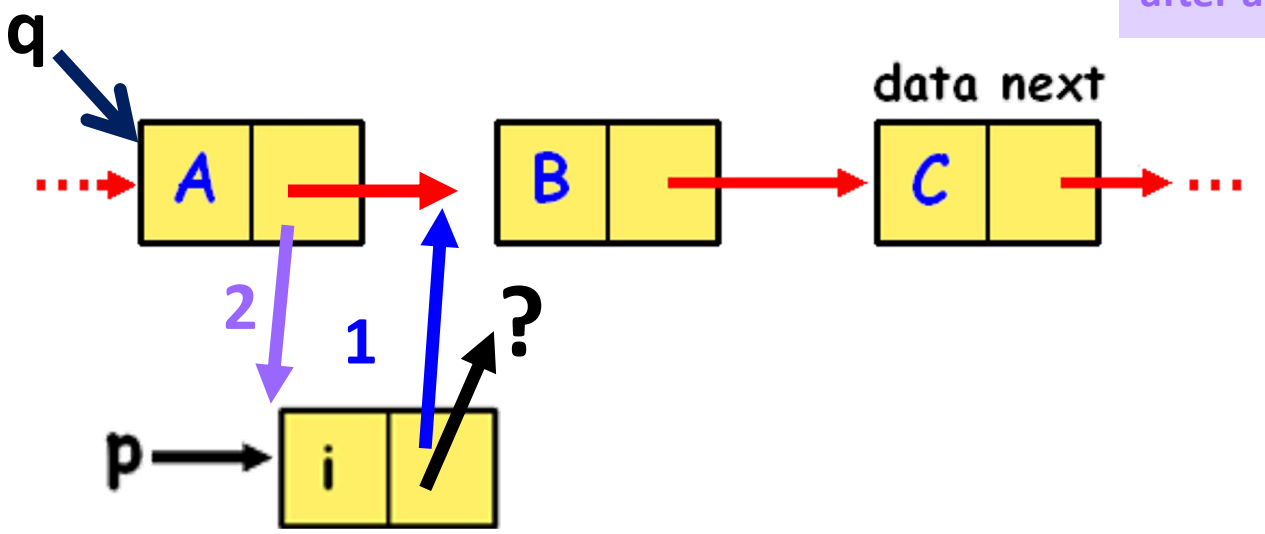
```
class list:

    """ unordered singly linked list
        with head """
    def __init__(self):
        self.head = None
    def append(self, data):

            """ add at the end of list"""
        p =  node(data)
        if self.head == None:  # null list
            self.head = p
        else:
            t = self.head
            while   t.next != None    :
                t = t.next
            t.next = p
```



Insert After

```
insertAfter('i',q)
```

insert node data
after a node pointed by q

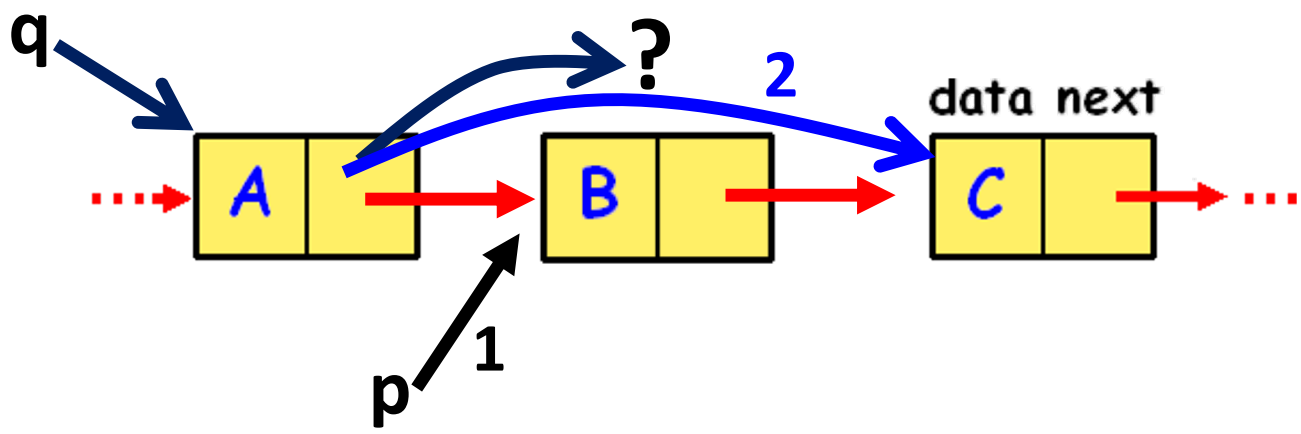


Why insert after ?
Can you insert before ?

Delete After

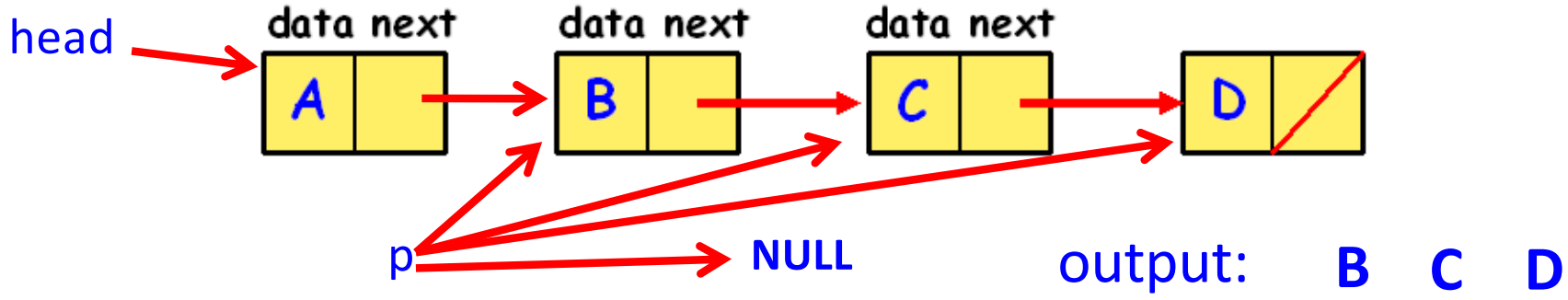
```
deleteAfter(q)
```

delete a node
after a node pointed by q



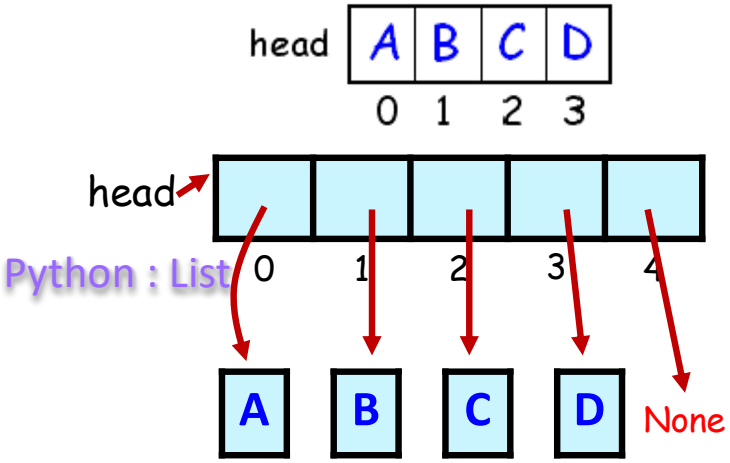
print list

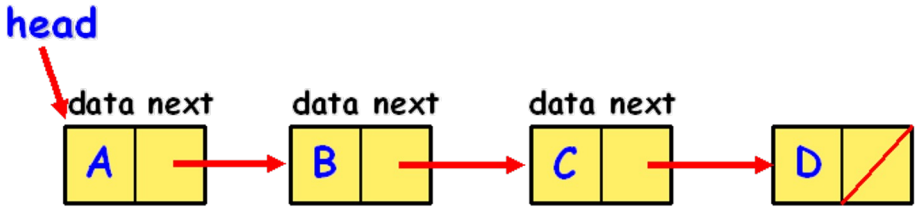
Design how to call.




```
        p is not None:
while p != None:
        print(p.data)
        p = p.next
}
```

Linked List VS Sequential Array

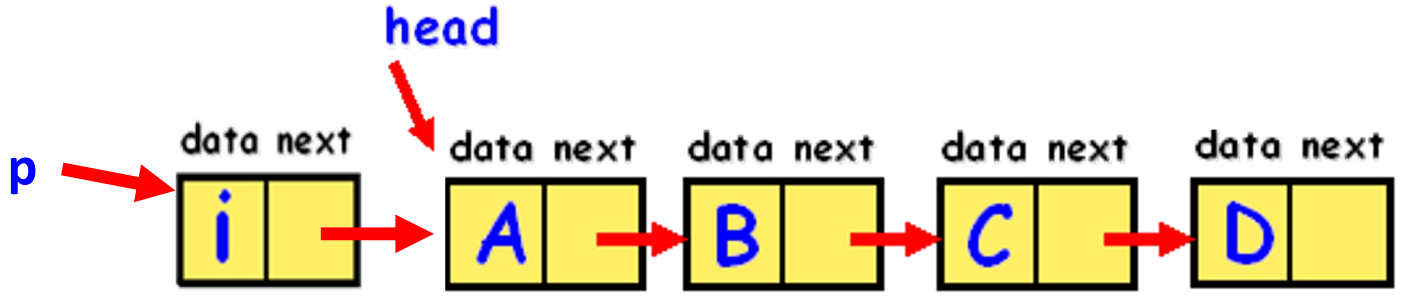




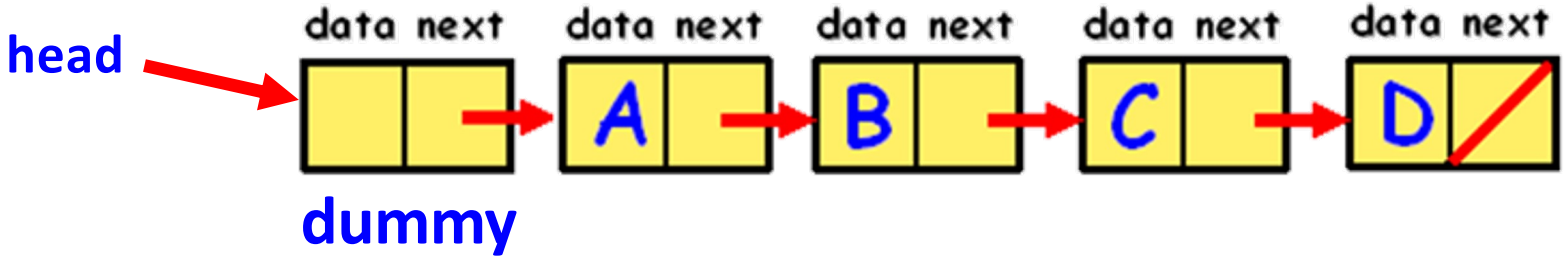
Sequential Array	Linked List
• Insertion / Deletion Shifting Problem. • Random Access. • C array : Automatic Allocation. Python List array : Dynamic Allocation • Lifetime : C-array, Python List • from defined until its scope finishes. • Only keeps data.	• Solved. • Sequential Access. • Node : Dynamic Allocation. • Node Lifetime : from allocated (C : malloc()/new, python: instantiate obj) until C: deallocated by free()/delete, Python : no reference. • Need spaces for linkes.

Dummy Node

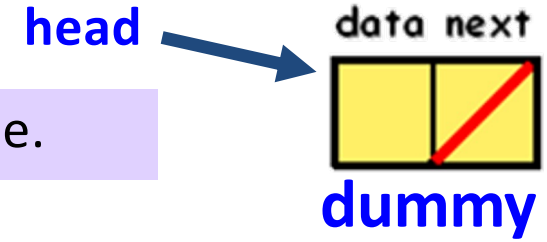
To insert & delete at 1st position change head ie. make special case.



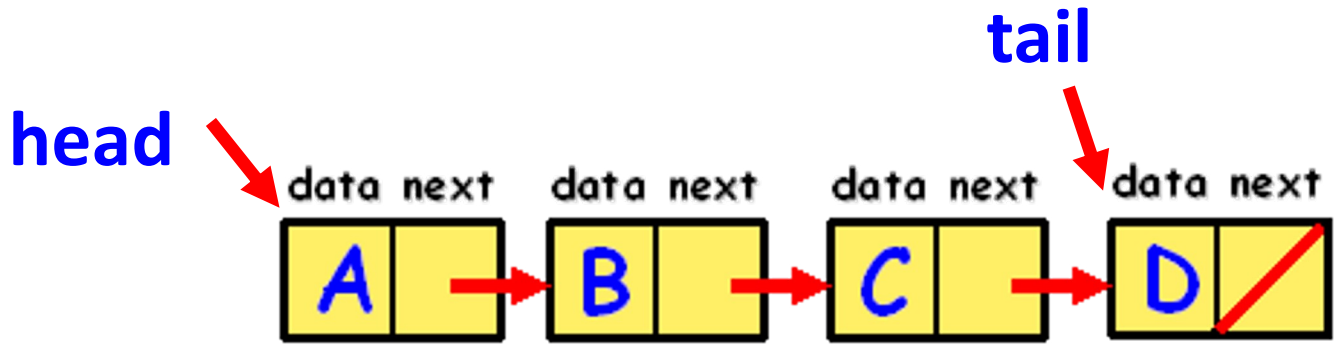
"Dummy Node" solves the problem.



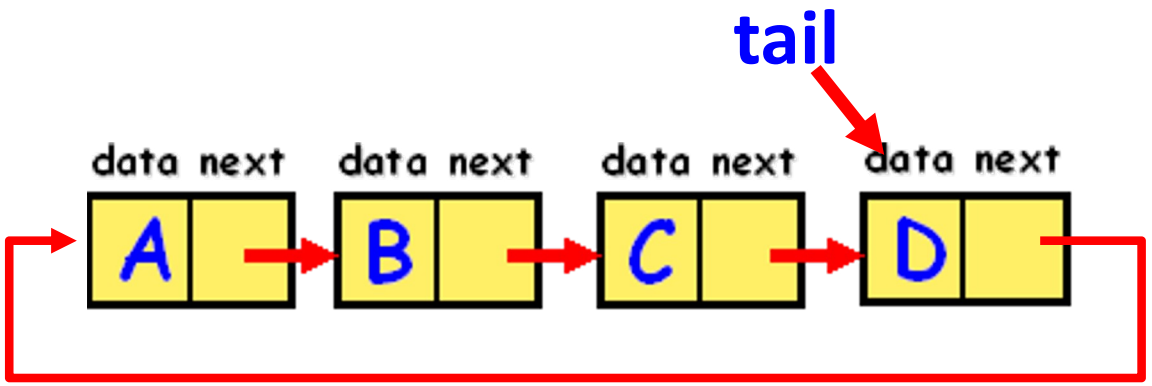
Empty List has a dummy node.



Head & Tail Nodes

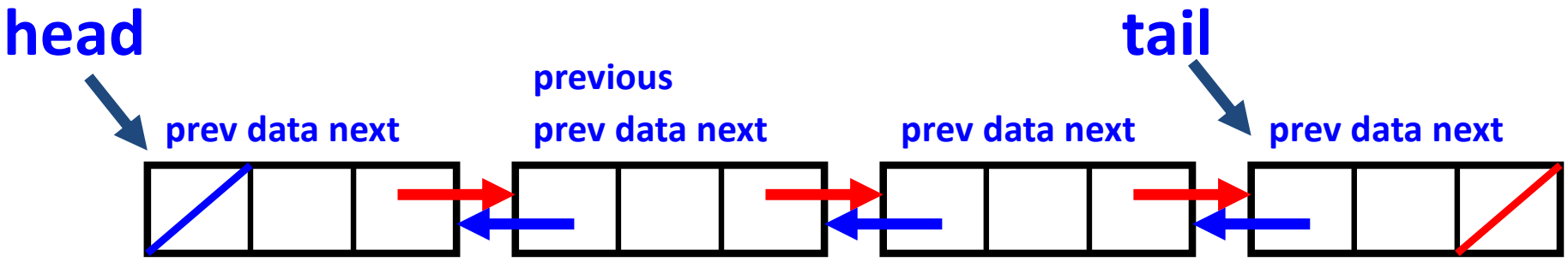


Circular List

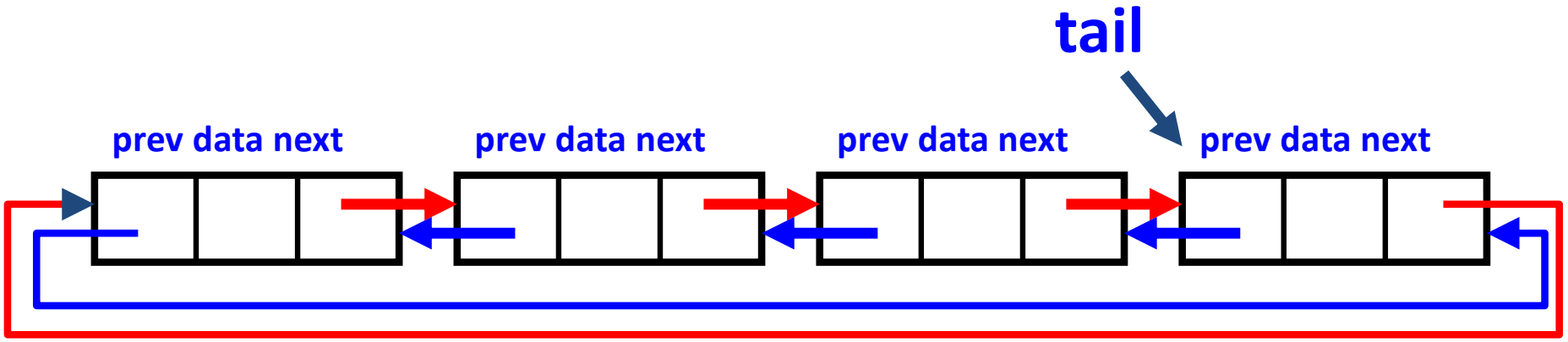


Why ptr to tail ? Why not ptr to head?

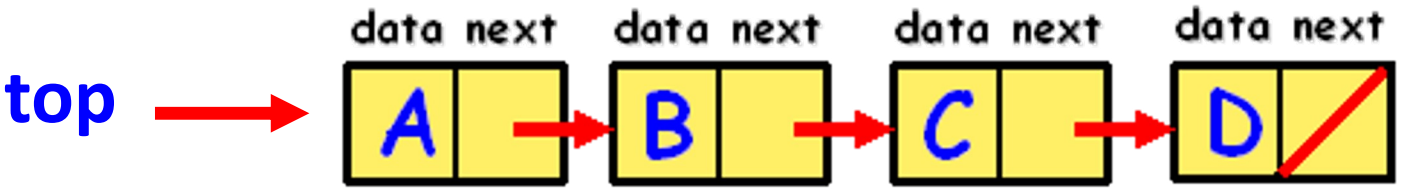
Doubly VS Singly Linked List



Doubly Circular List

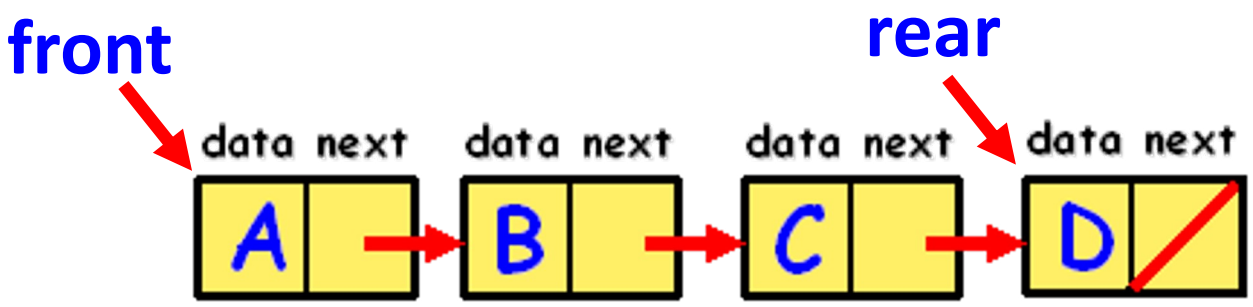


Linked Stack



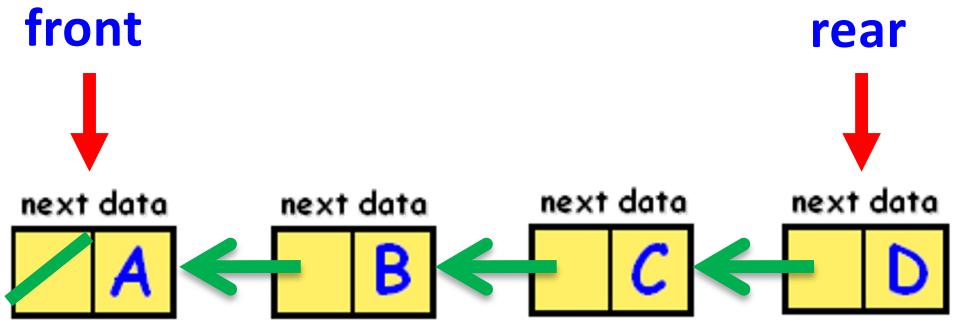
Check if it support every operations.

Linked Queue



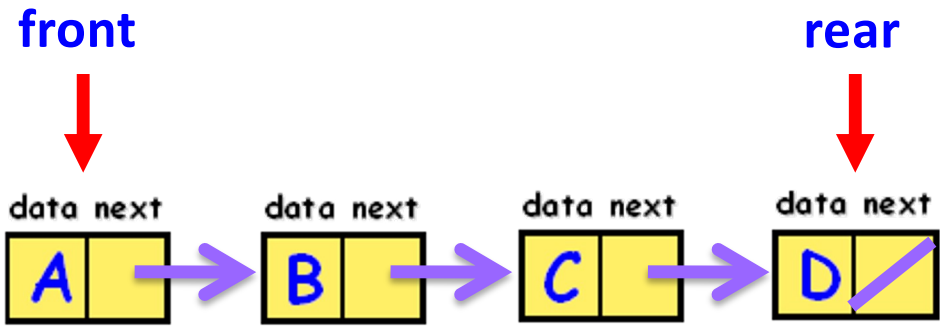
Can switch front & rear ?

Linked Queue



How do they link?

Support every operations ?



enQueue ? (insert)

deQueue ? (delete)

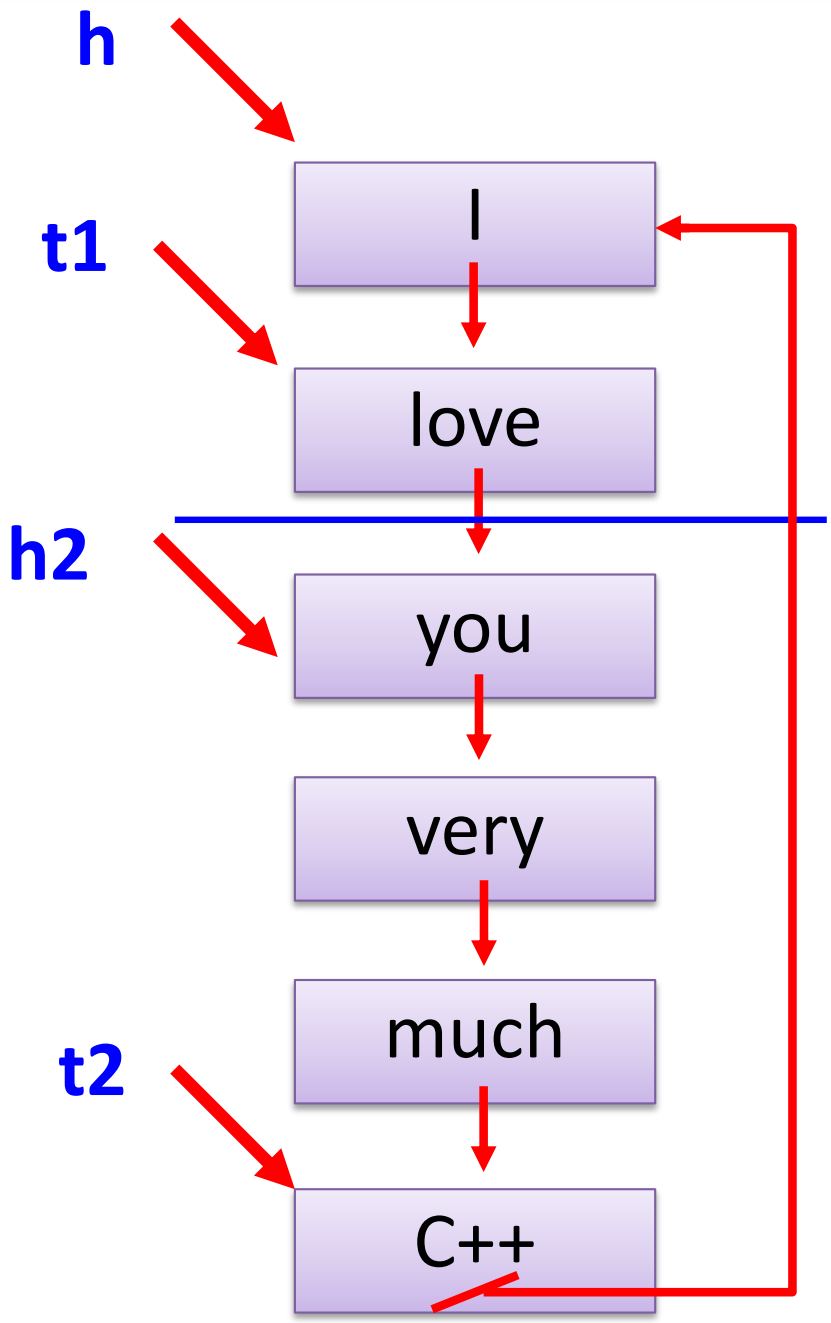
Every operations ?

Lab : Bottom Up

Lift it x%

Where to Lift_up ? x%

Take the bottom up.

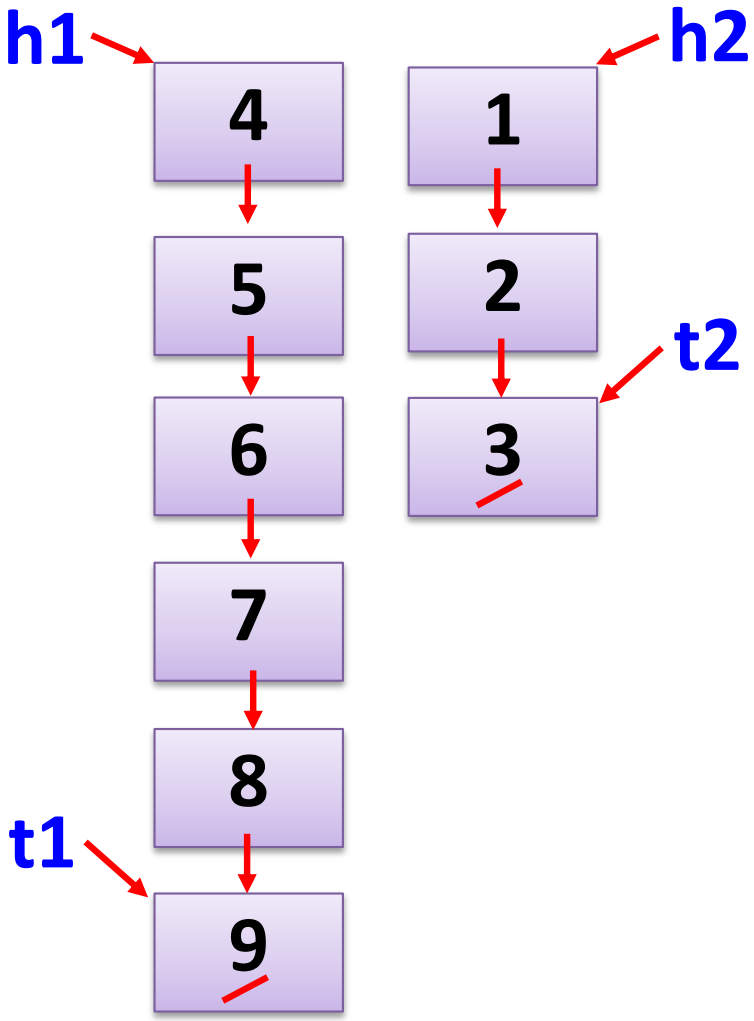


Try to print h2.

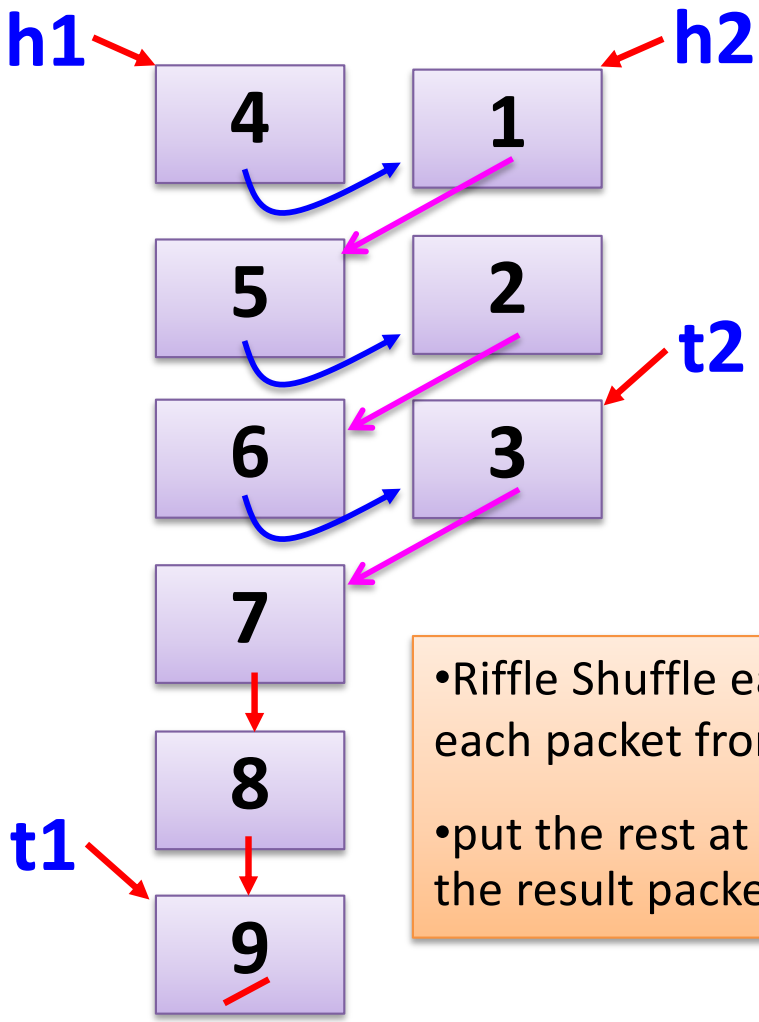
Opps !

infinite loop !

Lab : Riffle Shuffle



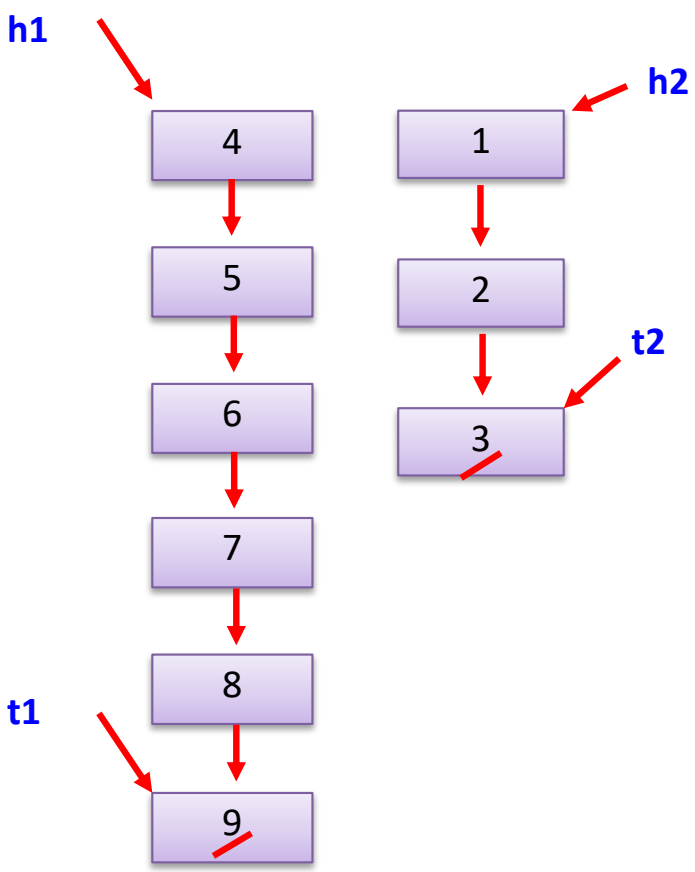
•Lift up to 2 packets.



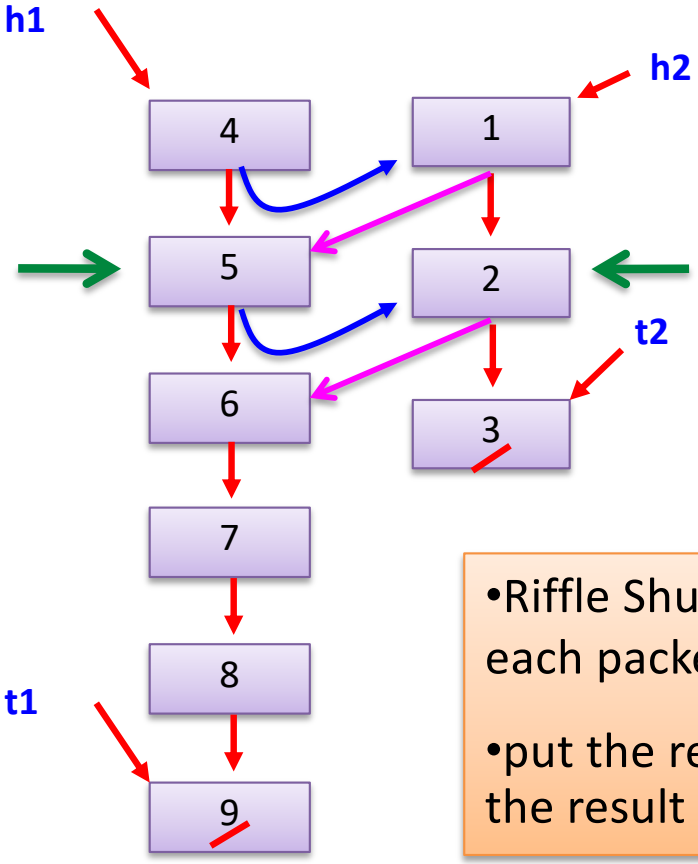
•Riffle Shuffle each node of each packet from the top.

•put the rest at the back of the result packet.

Lab : Riffle Shuffle



Lift up to 2 packets.



- Riffle Shuffle each node of each packet from the top.
- put the rest at the back of the result packet.

Applications

- Polynomial Expression
- Multilists
- Radix Sort

Polynomial expression

How about ... ?

$A = 5x^3 + 4x^2 - 7x + 10$

$+$

$B = x^3 + 2x - 8$

$C = 6x^3 + 4x^2 - 5x + 2$

$5x^{85} + 7x + 1$

$+$

$x^{76} - 8$

What data structure will you use?

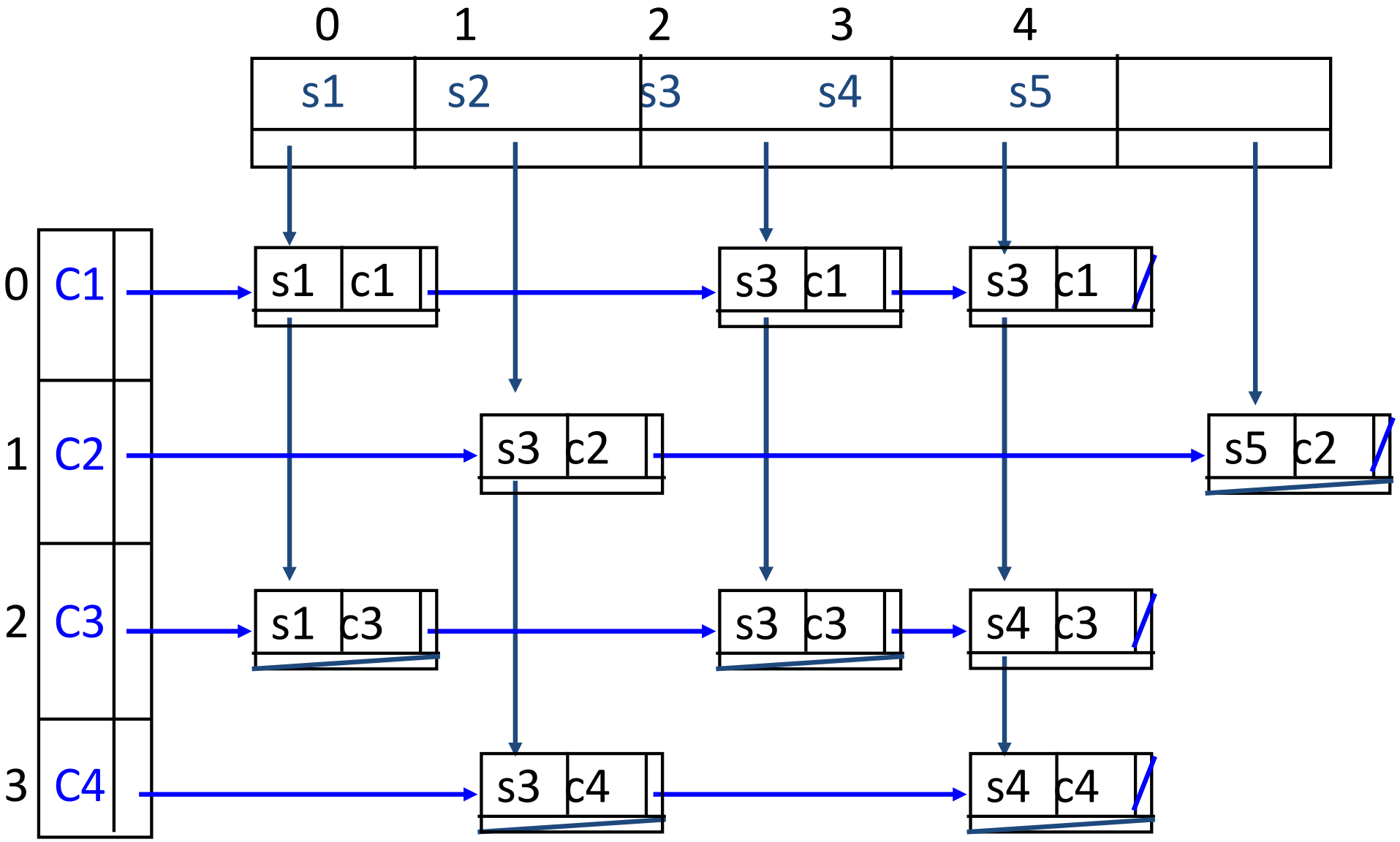
Array?

Array?

Sparse -> Linked List

(has lots of 0 data)

1. class หนึ่งๆ มีใครลงบ้าง 2. นร. คนหนึ่งๆ ลง class ใดบ้าง



Radix Sort

input: 64 8 216 512 27 729 0 1 343 125

0 1 512 343 64 125 216 27 8 729

0 1 2 3 4 5 6 7 8 9

8
729
1 216 27
0 512 125 343 64

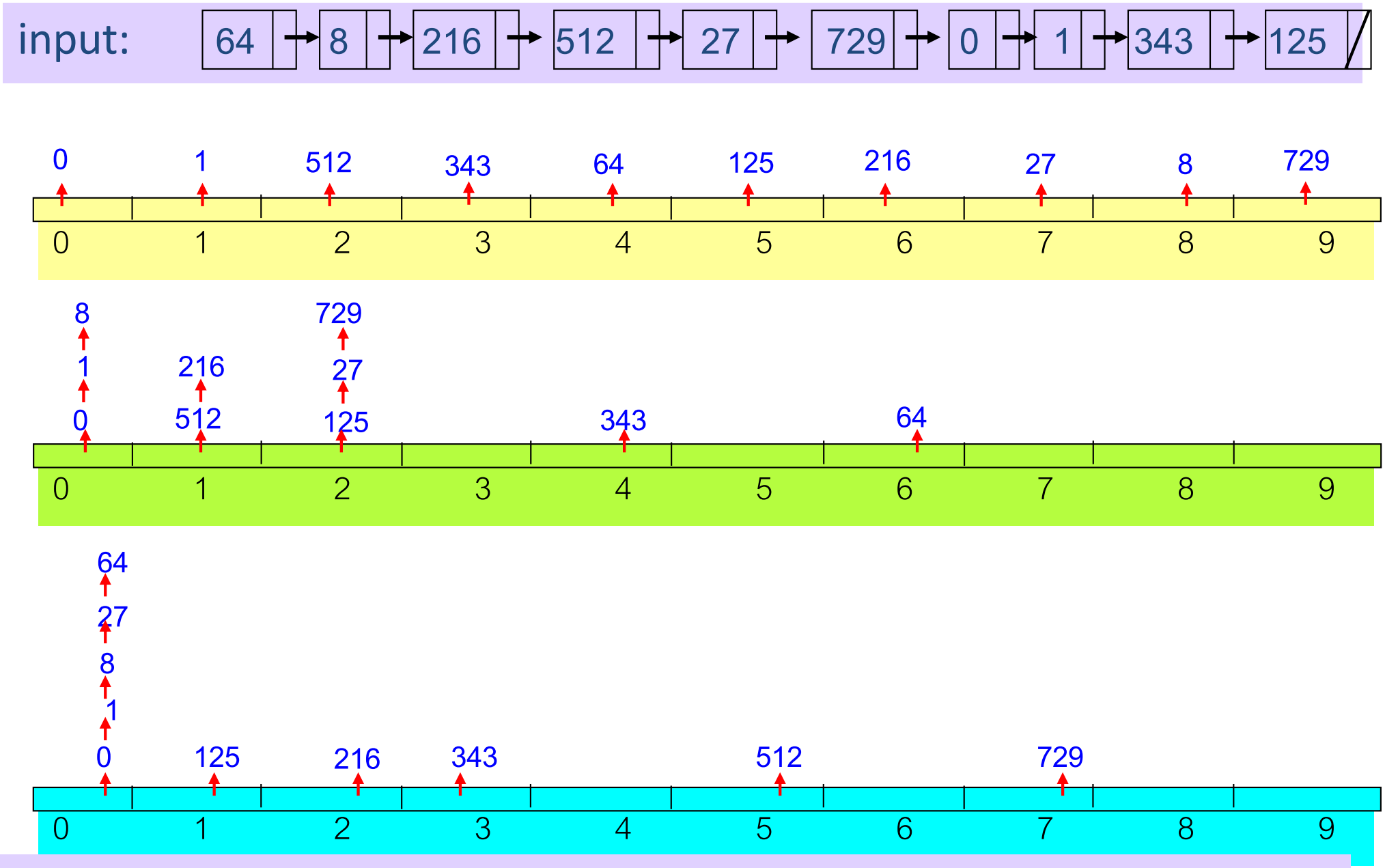
0 1 2 3 4 5 6 7 8 9

64
27
8
1
0 125 216 343 512 729

0 1 2 3 4 5 6 7 8 9

output: 0 1 8 27 64 125 216 343 512 729

Radix Sort


input:
64
8
216
512
27
729
0
1
343
125
0 1 512 343 64 125 216 27 8 729
0 1 2 3 4 5 6 7 8 9
8 729
1 216 27
0 512 125 343 64
0 1 2 3 4 5 6 7 8 9
64
27
8
1
0 125 216 343 512 729
0 1 2 3 4 5 6 7 8 9
output: 0 1 8 27 64 125 216 343 512 729