Non-Comparison Sorts O(n+k) หรือ O(n)

ใช้การกระจายตัวของข้อมูล

- Bucket Sort
- Radix Sort
- Counting Sort

} O(n)

7. Bucket Sort

ขั้นตอน

- กระจายข้อมูลในอาร์เรย์ ไปยังที่เก็บข้อมูลชุดหนึ่งที่เรียกว่า "ถัง" หรือ "บัคเก็ต" (Buckets)
- จากนั้นจะจัดเรียงข้อมูลภายใน bucket
- และนำข้อมูลที่เรียงแล้วทั้งหมดมารวมกัน

Java → Quick Sort

Python →

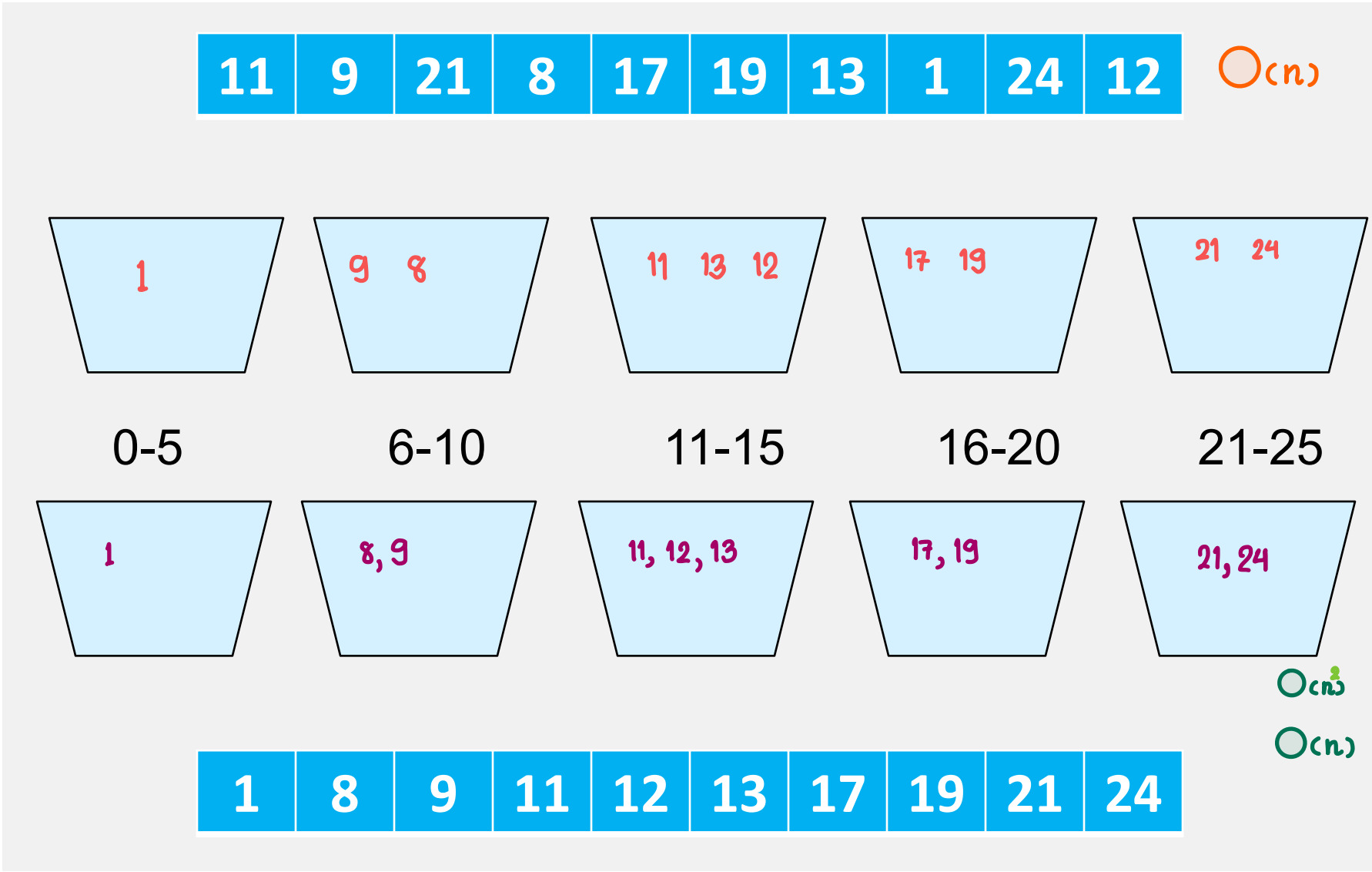
เมื่อข้อมูลนำเข้ามีการกระจายตัวอย่างสม่ำเสมอ (Uniformly Distributed) กรณีเฉลี่ยคือ O(n+k) k เป็นจำนวน bucket

- กรณีที่ k≈n = O(n)
- กรณีที่ k มีขนาดใหญ่/เล็กกว่า n มาก

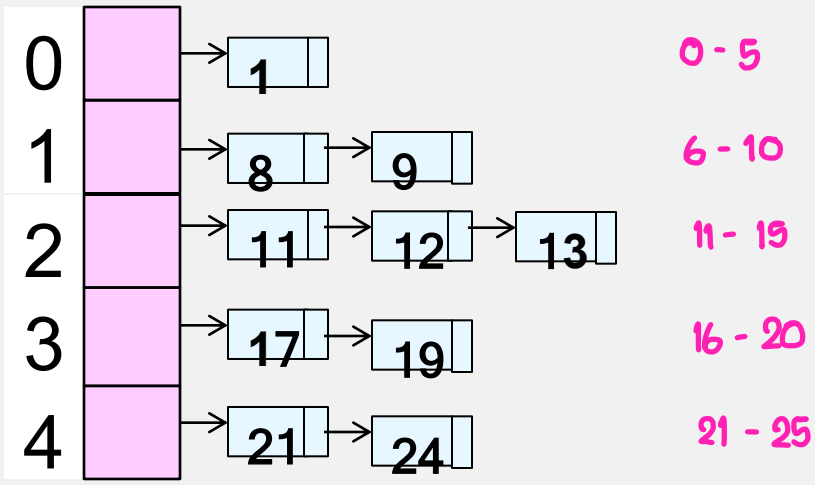
O(n+n)

O(2n)

กรณี worse case = $O(n^2)$


11 9 21 8 17 19 13 1 24 12
O(n)
1
9 8
11 13 12
17 19
21 24
0-5
6-10
11-15
16-20
21-25
1
8, 9
11, 12, 13
17, 19
21, 24
O(n)
O(n)
1 8 9 11 12 13 17 19 21 24

11	9	21	8	17	19	13	1	24	12



1	8	9	11	12	13	17	19	21	24

Bucket Sort

เลือกค่า k ใน Bucket Sort

1. ใช้จำนวนข้อมูล n

- กำหนดให้ $k \approx n$ หรือ $\sqrt{n}$
- เช่น ถ้ามีข้อมูล 100 ตัว เลือก k = 10 หรือ k = 100

2. พิจารณาจากช่วงของข้อมูล

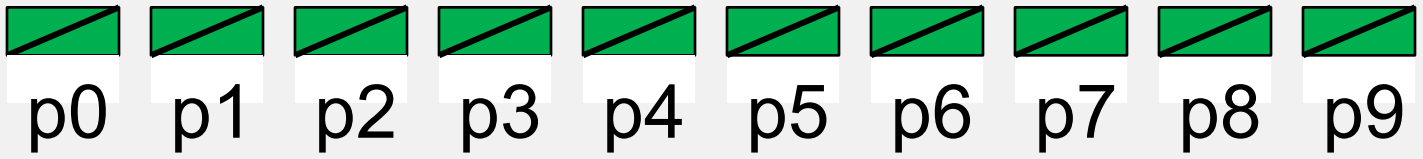
- ช่วง [min, max] เช่น [0, 100]
- สามารถแบ่ง bucket ตามช่วง เช่น bucket ละ 10 หน่วย → k = 10
- วิธีนี้ช่วยให้แต่ละ bucket มีขนาดใกล้เคียงกัน

3. จากการกระจายของข้อมูล

- ถ้าข้อมูลกระจุกตัวในบางช่วง → ใช้ k มากขึ้นเพื่อแยกให้ละเอียด
- ถ้าข้อมูลกระจายสม่ำเสมอ → ใช้ k น้อยลงก็ยังมีประสิทธิภาพ

8. Radix Sort

Input 64, 8, 216, 512, 27, 729, 0, 1, 343, 125



การหาหลักหน่วย n=x%10

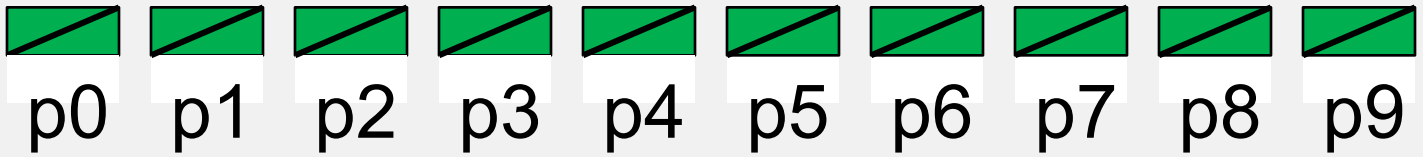
การหาหลักสิบ??

(1) ⟶ /10

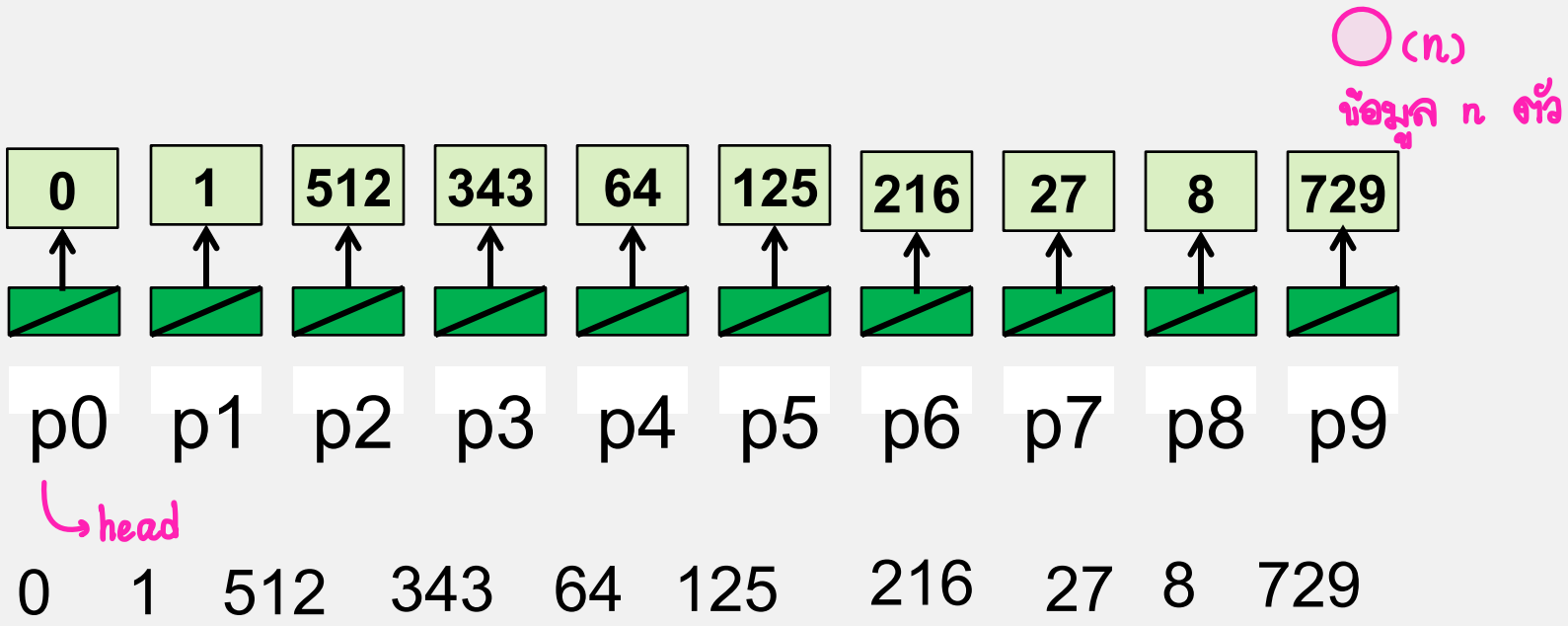
(2) ⟶ % 10

8. Radix Sort

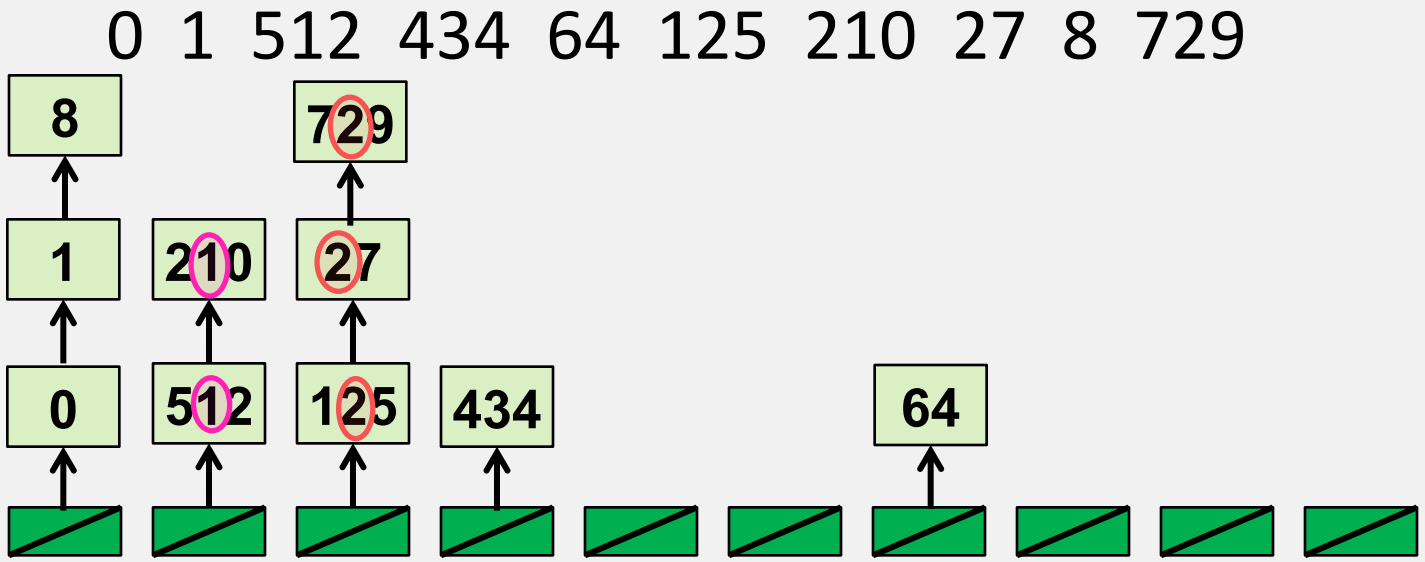
Input 64, 8, 216, 512, 27, 729, 0, 1, 343, 125



Input 64, 8, 216, 512, 27, 729, 0, 1, 343, 125



หลักหน่วย



0 1 8 512 210 125 27 729 434 64

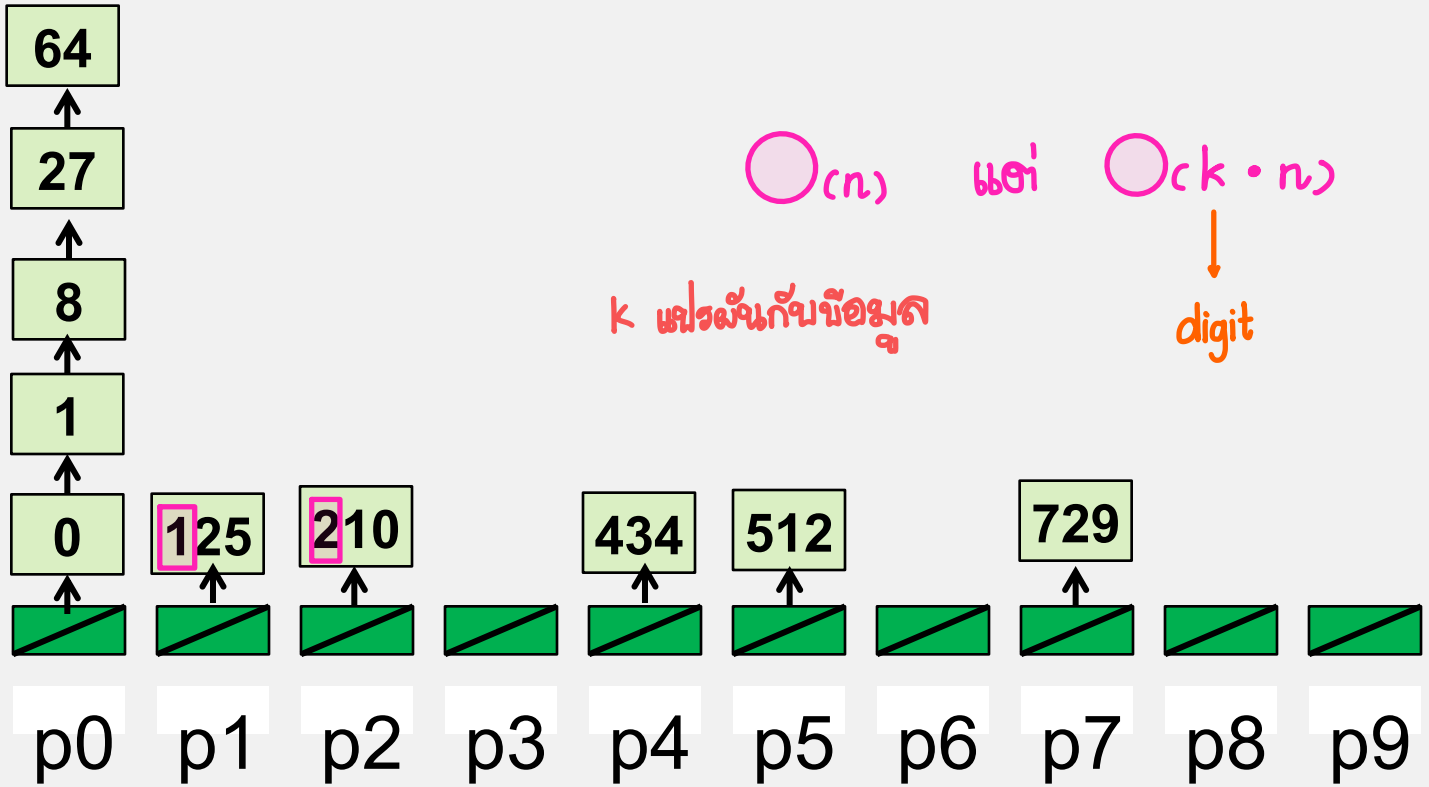
หลักสิบ

insert หลักสิบ

/10 แล้ว %10

0 1 8 512 210 125 27 729 434 64

หลักร้อย



ตาม digit คือ หน่วย , สิบ , ร้อย = 3

BigO = O(n*k) เมื่อ k คือจำนวน digit