به نام خدا

مرتب سازی خطی

counting sort زمان $\Theta(n+K)$

بازه اعداد $0 - K$ تعداد اعداد $n =$

صحیح

اگر $K = C$: مثلاً عدد مقدار ثابت داشته باشیم

این مقدار بین 0 تا 20 است.

$K = 20$ و تابعی از n نیست

$$\Theta(n+K) = \Theta(n+C) = \Theta(n)$$

$\Theta(n+k)$

$\Theta(n+k) = \Theta(n)$ اگر و تنها اگر $k \in O(n)$

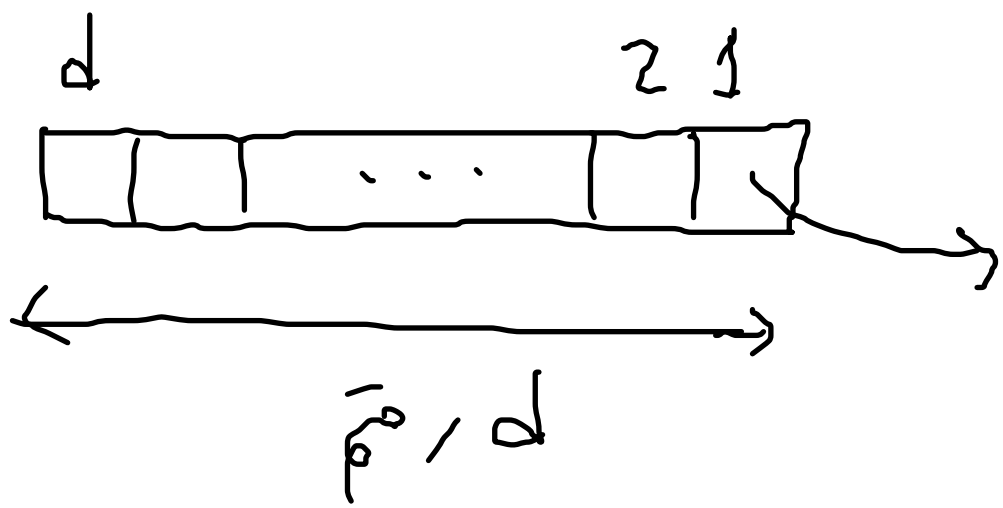
$\Theta(n+k) \in \Omega(n^2)$ $\longleftarrow$ $k \in \Omega(n^2)$

الگوریتم Radix sort

برای مرتب سازی n عدد صحیح d رقمی

در مبنای K

هر رقم $\in [0, K-1]$



روش معمول در مرتب سازی n عدد صحیح d رقمی :

ابتدا به ارقام یا رزشی بیشتر توجه میکنیم

329
457
657
839
436
720
355

گروه 329
355

گروه 457
436

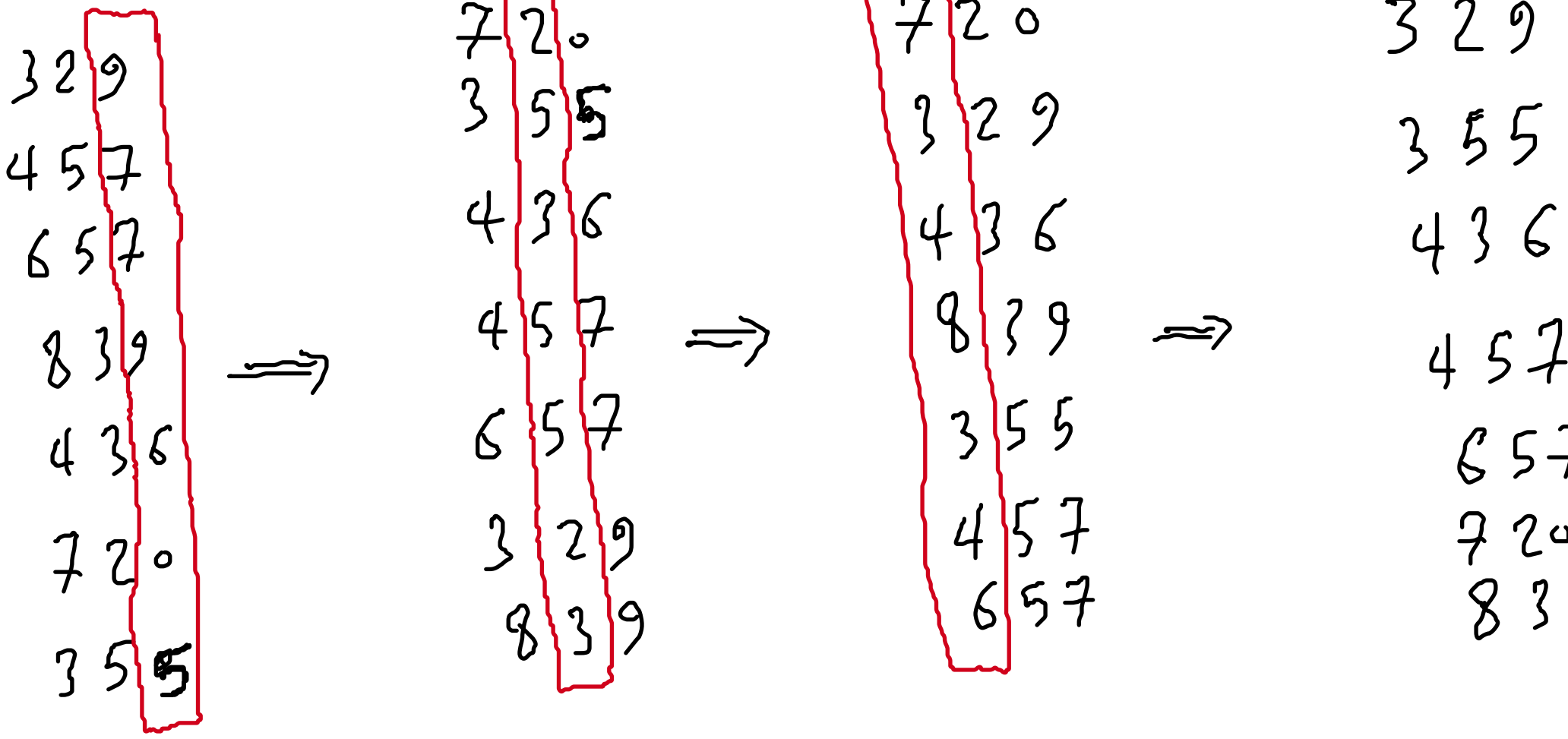
گروه 657

گروه 720

گروه 839

329
355
436
457
657
720
839

روش Radix : از ارقام کم ارزشی تر شروع می شود



Radix-Sort (A, d)

1. for $i \leftarrow 1$ to d do
2. use a stable sort to sort array A on digit i

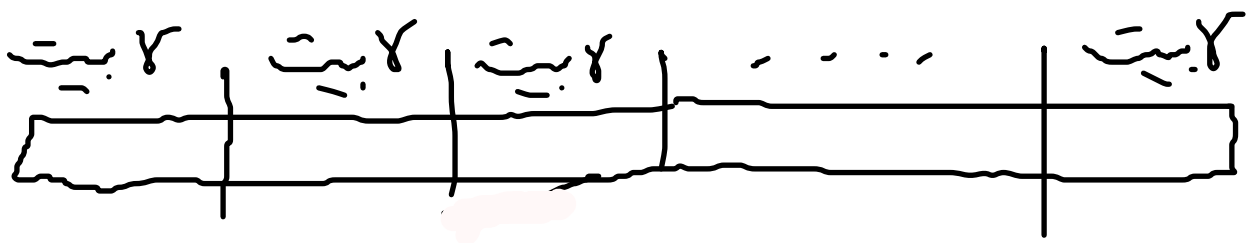
زمان: $\theta(d(n+k))$

ثابت در مورد d و k : $\Theta(d(n+k))$

اگر $d \in O(1)$ و $k + O(1)$
(d و k ثابت باشند)
$\Leftarrow \Theta(d(n+k)) = \Theta(n)$

انواع در تعداد ارقام و مبنا :

عدد n در b بیتی



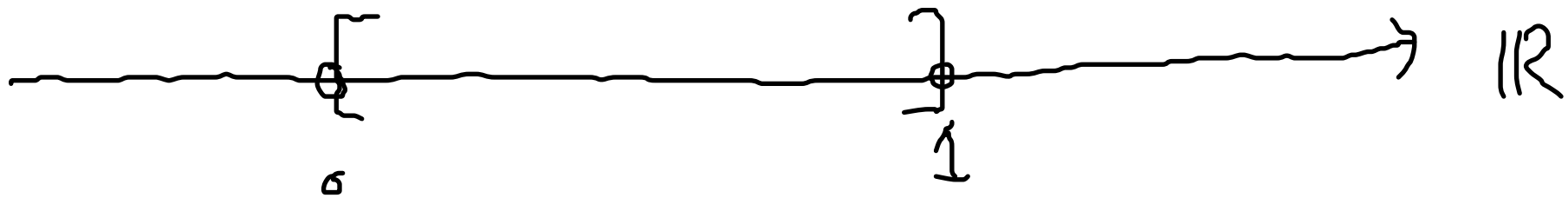
عدد b رقمی در مبنای ۲
هر توان ۲ بیت را
رقم فرض کرد.

عدد $\frac{b}{2}$ رقمی در مبنای 2^2

$$\Theta(b(n+2)) = \Theta\left(\frac{b}{2}(n+2^2)\right)$$

الگوریتم Bucket sort

n عدد حقیقی در بازه $0-1$
به صورت یکنواخت توزیع شده اند.
(یا احتمال داده در سراسر بازه پخش شده اند)



الگوریتم Bucket sort را میتوان تصحیح
Quick sort ارتقا گرفت. در مرتب سازی سریع 2 بخش
داشتیم، اینجا n بخش ایجاد میکنیم.

n تا باکت ارتقا میگیریم :

$$[0, \frac{1}{n}), [\frac{1}{n}, \frac{2}{n}), [\frac{2}{n}, \frac{3}{n}), \dots, [\frac{n-1}{n}, \frac{n}{n}]$$

B_0 B_1 B_2 B_{n-1}

n عدد ورودی را باید در بین باکت ها توزیع کنیم و هر عدد در باکت مربوطه به خودش قرار گیرد.

سپس اعداد هر باکت به صورت جداگانه با الگوریتم insertion sort مرتب میشود.

برای پارتیشن کردن اعداد اگر آنها را با مرزهای باکتها مقایسه کنیم $n-1$ تا مرز داریم و n تا عدد پس $n(n-1)$ مقایسه باید انجام شود که $O(n^2)$

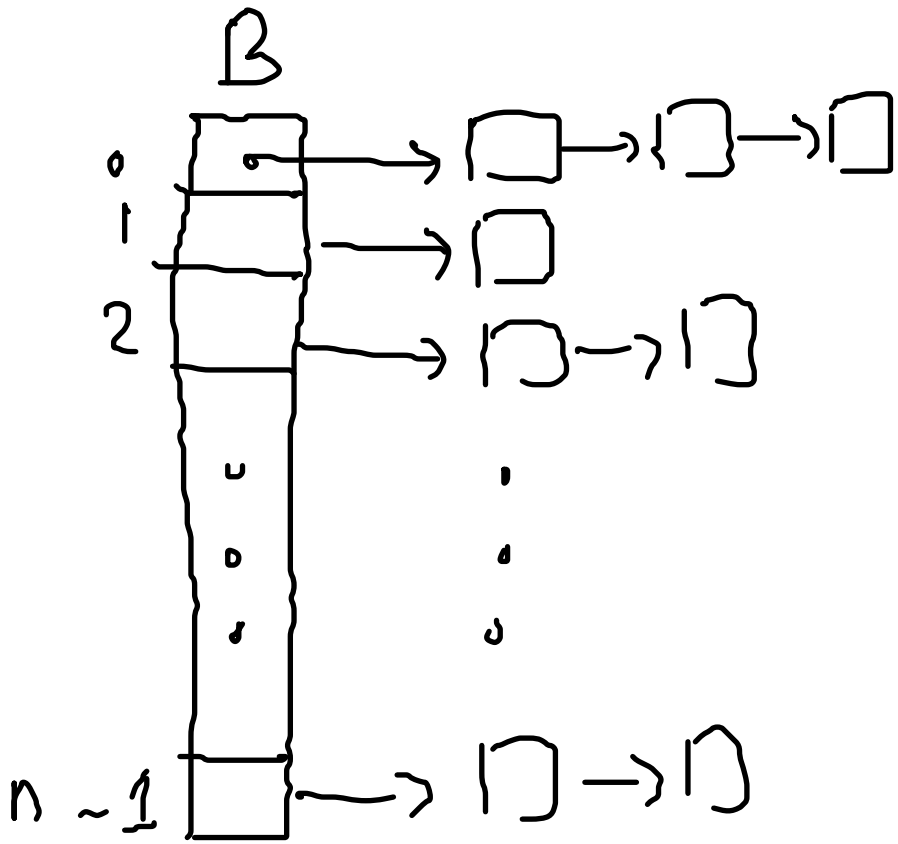
روش سطل:

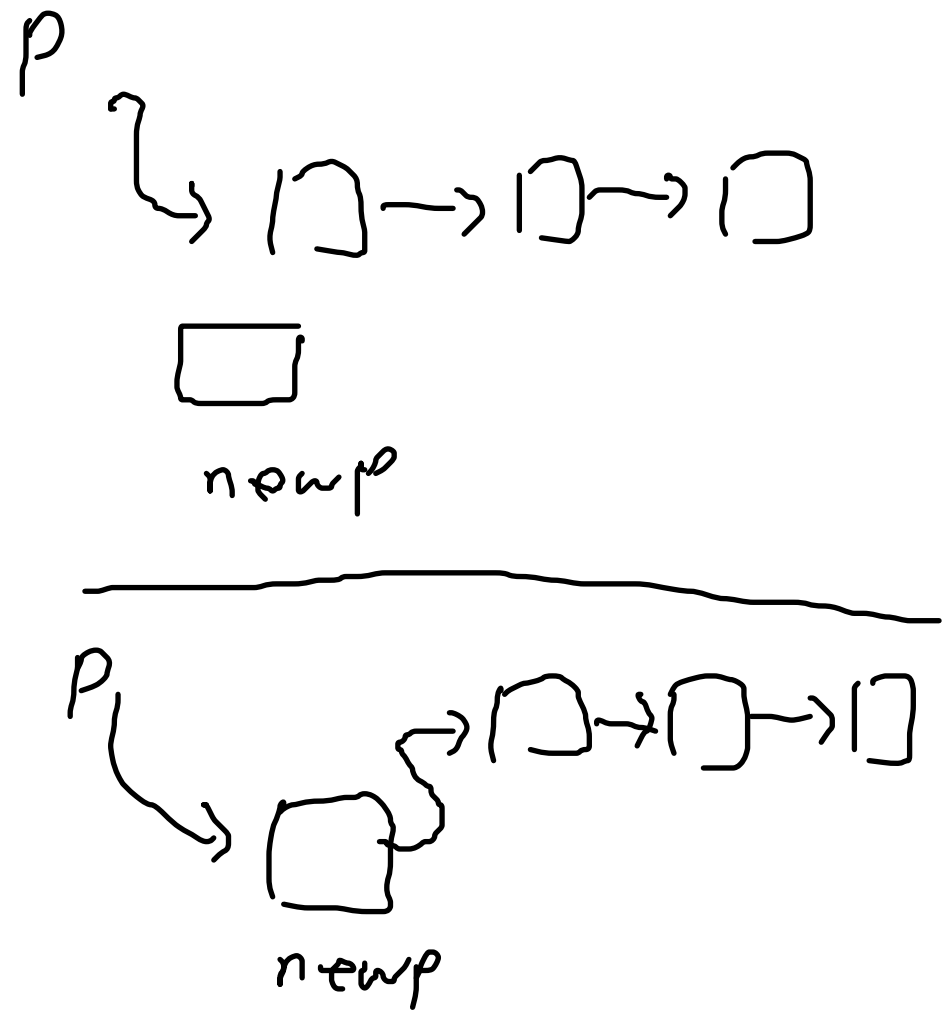
$B[\lfloor n * A[i] \rfloor] \xleftarrow{?} A[i]$

```
Bucket_sort (A)
c1  1. n ← length [A]
c2n 2. for i ← 1 to n do
    3.     insert A[i] into list B[⌊n*A[i]⌋]
    4. for i ← 0 to n-1 do
Σ_{i=0}^{n-1} O(n_i^2)  5.     Sort list B[i] with insertion sort
O(n)  6. concatenate the lists B[0], B[1], ..., B[n-1]
```

$$T(n) = c_1 + c_2 n + \sum_{i=0}^{n-1} O(n_i^2) + O(n)$$

n_i = تعداد اعداد در سطل i ام





newp->next = p

p = newp

시간: O(1)

زمان الگوریتم $T(n) = O(n) + \sum_{i=0}^{n-1} O(n_i^2)$

n_i تعداد اعداد در بالکت i ام است.

زمان الگوریتم به n_i بستگی دارد.

① اگر $n_i = 1$ ← یعنی در هر بالکت یک عدد

$$T(n) = O(n) + \sum_{i=0}^{n-1} O(1) = O(n) + O(n)$$

$$= O(n)$$

بهترین حالت

② اگر حداقل یکی از n_i ها $\Theta(n)$ یا بزرگ (یعنی متناسب از n) کسری از n

$$n_i = c \cdot n \longrightarrow O(n_i^2) = O((c \cdot n)^2) = O(n^2)$$

هر بایت ها می‌توانند مضرب از n عدد داشته باشند
چون کلاً n تا عدد داریم. پس تعداد بایت‌هایی
که مضرب از n عدد دارند ثابت است.

$$T(n) = O(n) + c \cdot O(n^2) + (n-c) \cdot O(1)$$

برای $n-c$ بایت که تعداد ثابت عدد دارند

برای c بایت که مضرب از n عدد دارند

$$\boxed{T(n) = O(n^2)}$$

● بدترین حالت
آنهایی که بایت بیشتر از یک عدد باشد ولی تعداد آنها
ثابت باشد باز هم $O(n)$ می‌شود.

اگر فرض کنیم اینکه اعداد در بازه ۰ تا ۱
به صورت یکنواخت با احتمال های توزیع شده باشند
را داشته باشیم میتوان نشان داد که:

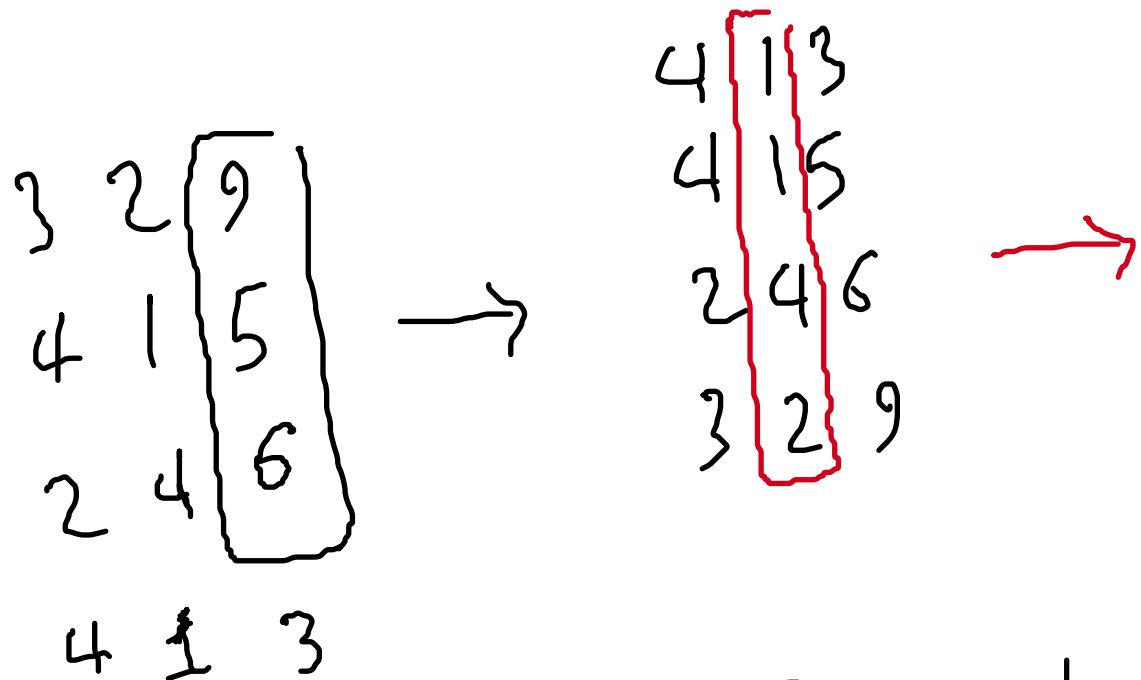
در حالت میانگین هم زمان $O(n)$ میشود.

$$E[T(n)] = O(n)$$

امید ریاضی یا
حالت میانگین

توضیح counting sort

استفاده از counting sort در radix sort



۲ نکته: ① رقم ها را حتماً در آرایه خروجی میسازیم چون این رقم جزئی از یک عدد است باید ببریم در ورودی و عدد را برداشته و در خروجی کپی بکنیم.

② دو رقم مساوی ترتیبشان مهم است زیرا قبلاً رو رقم های دیگر مرتب شده اند