



Radix Sort 2

bucket Sort

طراحی الگوریتم‌ها – جلسه دهم

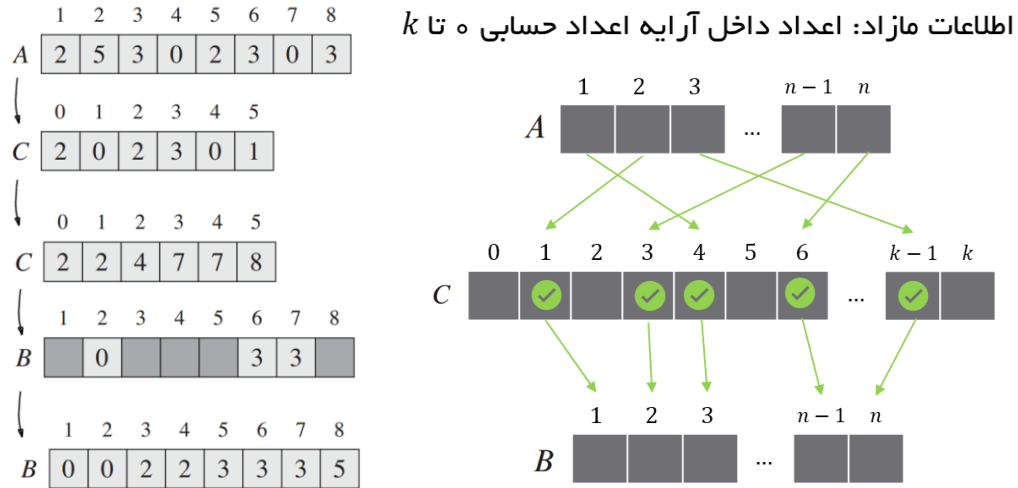
Introduction to Algorithm

استاد: جوانمردی

۱۳۹۹/۸/۵

مرور جلسه قبل

مرتب‌سازی خطی COUNTING SORT



مرتب‌سازی خطی RADIX SORT

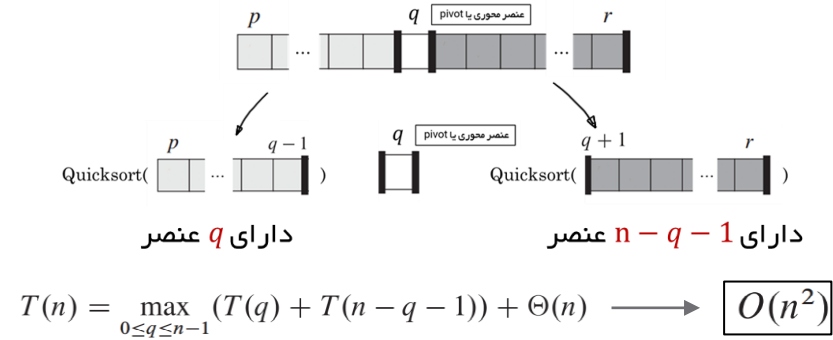
اطلاعات مزاد: اعداد دارای d رقم و هر رقم k مقدار متفاوت

329	720	720	329	RADIX-SORT(A, d)
457	355	329	355	1 for $i = 1$ to d
657	436	436	436	2 use a stable sort to sort array A on digit i
839	457	839	457	
436	657	355	657	
720	329	457	720	
355	839	657	839	

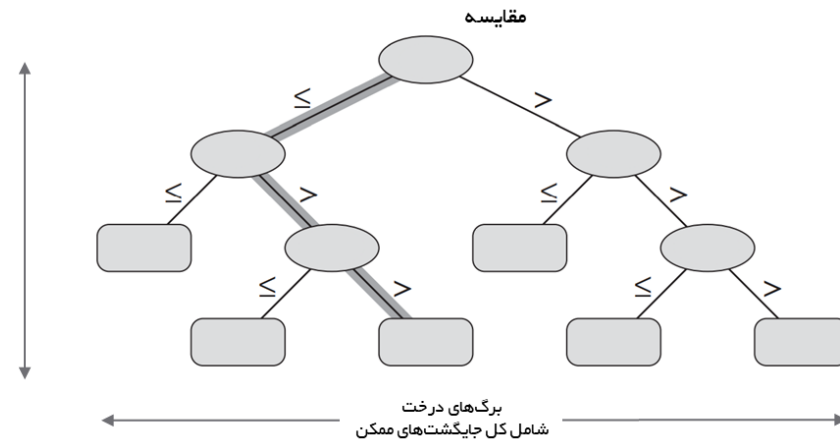
• زمان اجرا: $\theta(d(n + k))$

• اگر $k = O(n)$ و $d = O(1)$ باشد زمان اجرای radix sort $\theta(n)$ خواهد بود

تحلیل زمانی QUICK SORT



درخت تصمیم برای مرتب‌سازی مقایسه‌ای



Theorem 8.1

Any comparison sort algorithm requires $\Omega(n \lg n)$ comparisons in the worst case.

مرتب‌سازی مبنایی radix sort

- مرتب‌سازی ۱۰۰۰ عدد که مقدار هرکدام میتواند بین ۰ تا ۹۹۹،۹۹۹ باشد؟
- اطلاعات مازاد: اعداد داخل آرایه دارای d رقم که هر رقم k مقدار متفاوت به خود میگیرد
- ایده اصلی: مرتب‌سازی کل از طریق مرتب‌سازی رقم به رقم

روش شهودی – غلط

329	839	<u>839</u>
457	720	<u>720</u>
657	657	<u>657</u>
839	457	<u>457</u>
436	436	<u>436</u>
720	329	<u>329</u>
355	355	<u>355</u>

- مرتب‌سازی از رقم با ارزش بیشتر

- در بدترین حالت چندبار تابع sort فراخوانی می‌شود؟

d رقم که هر رقم k مقدار متفاوت

مرتب‌سازی مبنایی radix sort

رقم d که هر رقم k مقدار متفاوت

روش شهودی – غلط

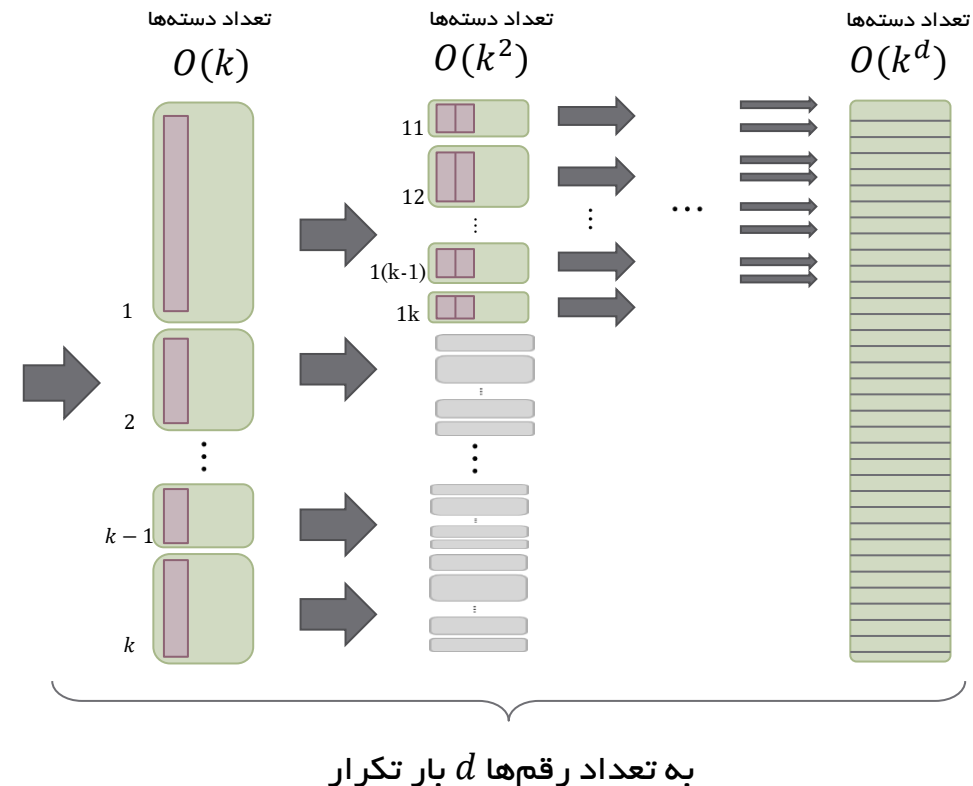
329	839	839
457	720	720
657	657	657
839	457	457
436	436	436
720	329	329
355	355	355

• مرتب‌سازی از رقم با ارزش بیشتر

• در بدترین حالت چندبار تابع sort فراخوانی می‌شود؟ $\Theta(k^{d-1})$

تعداد فراخوان sort در بدترین حالت

$$\leq \sum_{i=0}^{d-1} k^i = \frac{k^d - 1}{k - 1} = \Theta(k^{d-1})$$

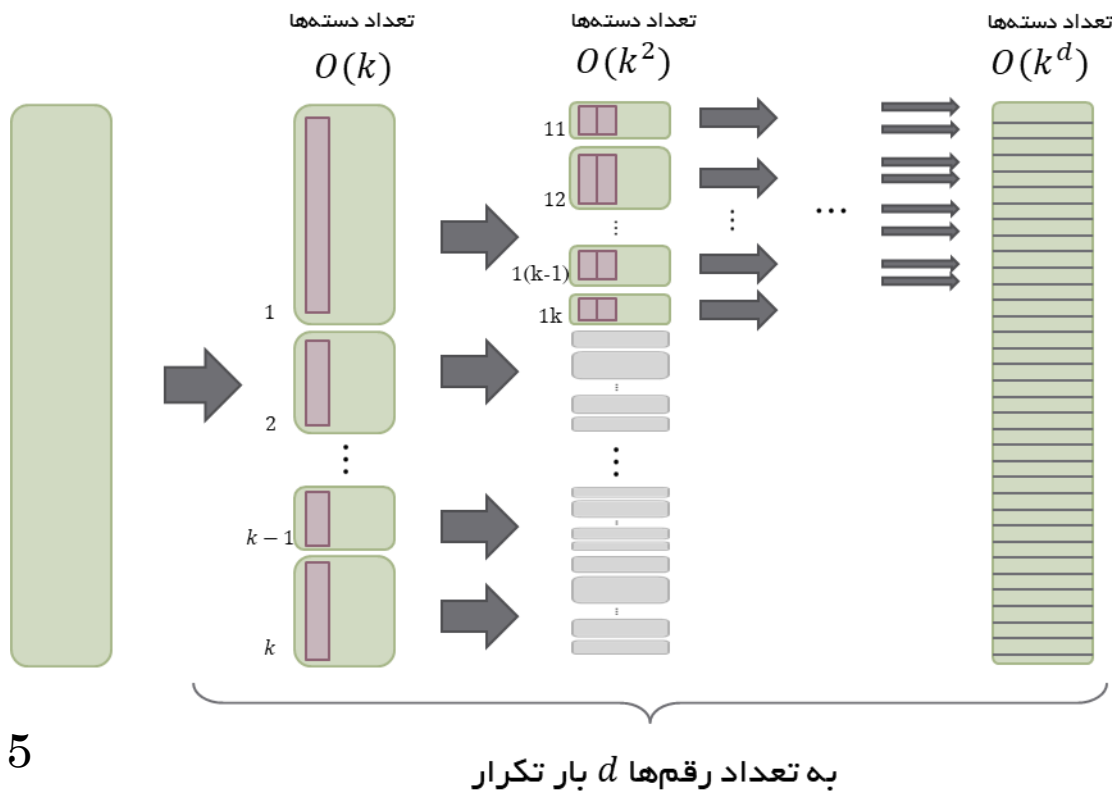


مرتب‌سازی مبنایی radix sort

$$\sum_{i=0}^{d-1} k^i = \frac{k^d - 1}{k - 1} = \Theta(k^{d-1})$$

• در بدترین حالت چندبار تابع sort فراخوانی می‌شود؟

تعداد $n = 16$ عدد $d = 3$ رقمی با $k = 2$ مقدار متفاوت هر رقم



000	000	000	000
001	001	001	000
010	010	000	001
011	011	001	001
100	000	010	010
101	001	011	010
110	010	010	011
111	011	011	011
000	100	100	100
001	101	101	100
010	110	100	101
011	111	101	101
100	100	110	110
101	101	111	110
110	110	110	111
111	111	111	111

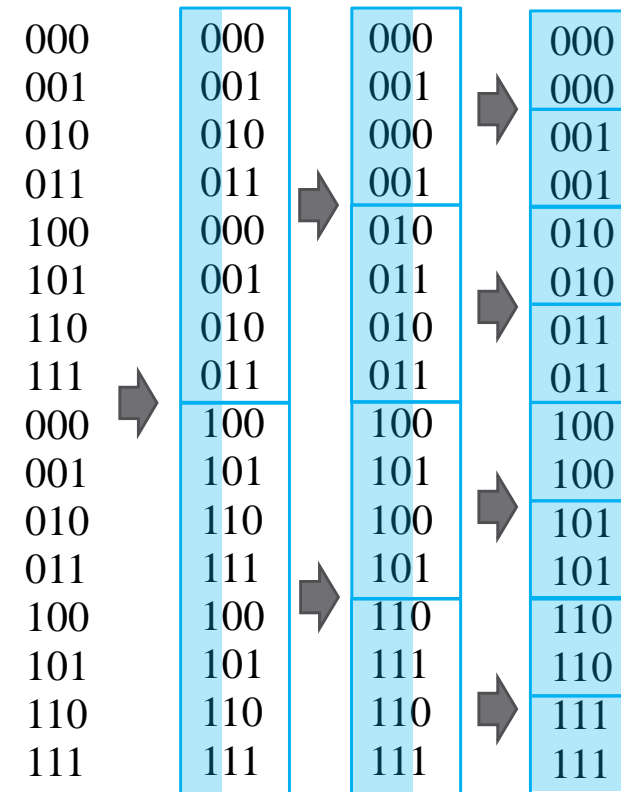
مرتب‌سازی مبنایی radix sort

$$\sum_{i=0}^{d-1} k^i = \frac{k^d - 1}{k - 1} = \Theta(k^{d-1})$$

• در بدترین حالت چندبار تابع sort فراخوانی می‌شود؟

تعداد $n = 16$ عدد $d = 3$ رقمی با $k = 2$ مقدار متفاوت هر رقم

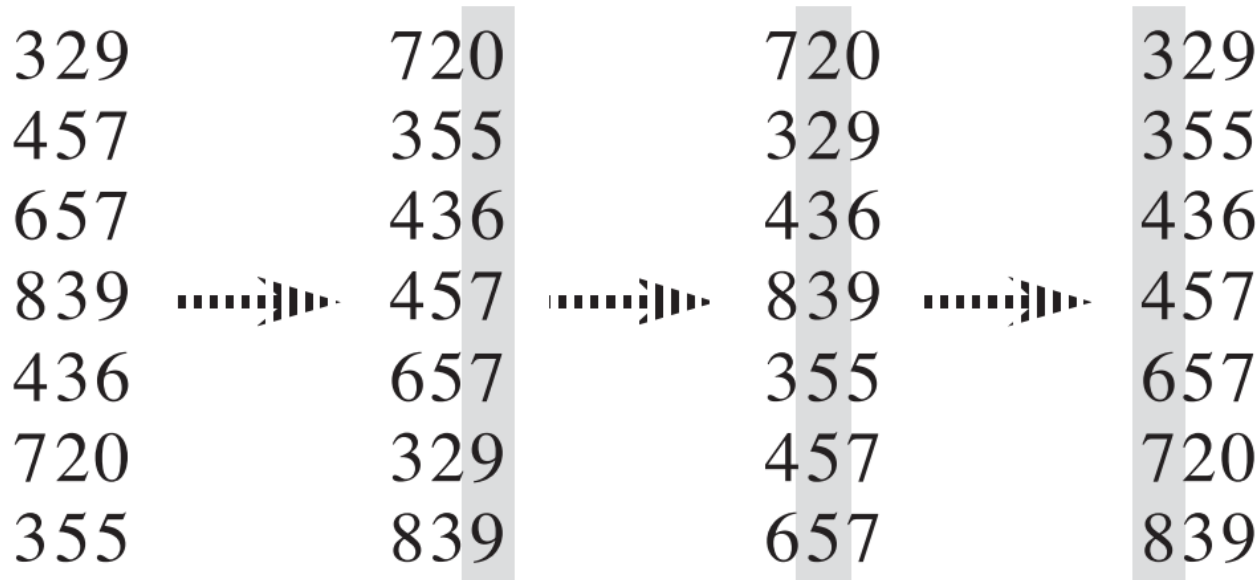
برای حل الگوریتم، می‌بایست در هر لحظه ماکسیمم اطلاعات چند دسته در حافظه نگهداری شود؟



مرتب‌سازی مبنایی radix sort

عکس روش شهودی

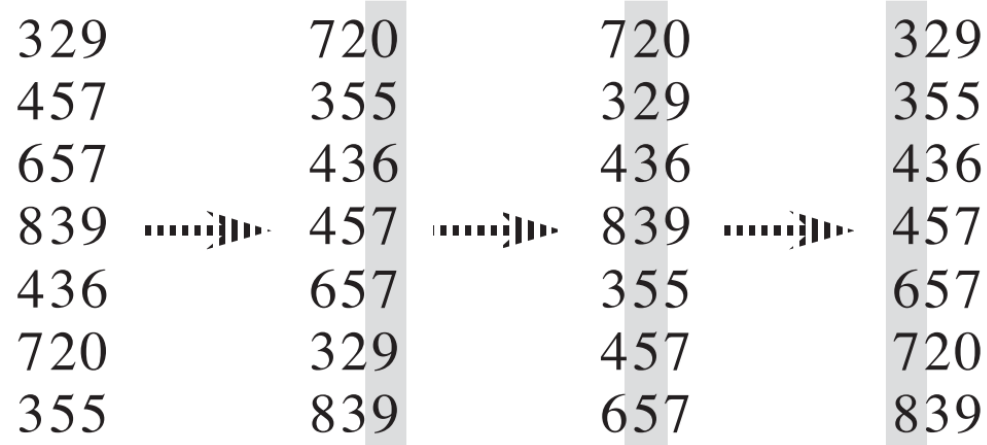
- مرتب‌سازی از رقم با ارزش کمتر
- مرتب‌سازی هر رقم باید پایدار باشد! ← استفاده از counting sort



پیاده‌سازی radix sort و زمان اجرا

RADIX-SORT(A, d)

```
1  for  $i = 1$  to  $d$ 
2      use a stable sort to sort array  $A$  on digit  $i$ 
```



• زمان اجرا: $\theta(d(n + k))$

• اگر $k = O(n)$ و $d = O(1)$ باشد زمان اجرای radix sort $\theta(n)$ خواهد بود

تحلیل دقیق‌تر مرتبه زمانی radix sort

• اگر n عدد b بیتی داشته باشیم، برای هر عدد مثبت دلخواه $r \leq b$ خواهیم داشت:

زمان اجرایی radix sort برای این اعداد $\Theta((b/r)(n + 2^r))$ خواهد بود

به شرطی که مرتب‌سازی پایدار استفاده شده $\Theta(n + k)$ باشد

• اثبات:

برای $r \leq b$ فرض کنید که هر رقم r بیتی باشد و تعداد ارقام برابر $d = \lceil b/r \rceil$
در این صورت هر رقم یک عدد صحیح بین 0 تا $2^r - 1$ خواهد بود
و می‌توان از counting sort با $k = 2^r - 1$ استفاده کرد

$$b = 32, r = 8, k = 2^r - 1 = 255, \text{ and } d = b/r = 4$$



زمان counting sort: $\Theta(n + k) = \Theta(n + 2^r)$

زمان اجرای radix sort: $\Theta(d(n + 2^r)) = \Theta((b/r)(n + 2^r))$ ←

تحلیل دقیق‌تر مرتبه زمانی radix sort

• اگر n عدد b بیتی داشته باشیم، برای هر عدد مثبت دلخواه $r \leq b$ خواهیم داشت:

زمان اجرایی radix sort برای این اعداد $\Theta((b/r)(n + 2^r))$ خواهد بود

به شرطی که مرتب‌سازی پایدار استفاده شده $\theta(n + k)$ باشد

• برای اعداد n و b تعیین $r \leq b$ بگونه‌ای که زمان اجرای $(b/r)(n + 2^r)$ را کمینه کند

حالت اول: $b < \lfloor \lg n \rfloor \leftarrow (n + 2^r) = \Theta(n) \leftarrow$ برای $r = b$ خواهیم داشت: $(b/b)(n + 2^b) = \Theta(n)$

حالت دوم: $b \geq \lfloor \lg n \rfloor \leftarrow$ برای $r = \lfloor \lg n \rfloor$ خواهیم داشت: $(b/\lfloor \lg n \rfloor)(n + n) = \Theta(bn/\lg n)$

radix sort یا انواع مرتب‌سازی مقایسه‌ای؟

- در صورتی که $b = O(\lg n)$ باشد با انتخاب $r = \lfloor \lg n \rfloor$ زمان اجرای radix sort برابر $O(n)$ است.
- در این صورت، تعداد فراخوان‌های radix sort به مراتب کمتر از quick sort خواهد بود در حالی که زمان اجرای هر فراخوان radix sort به مراتب طولانی‌تر از دیگری است.
- اینکه کدام روش مرتب‌سازی بهتر است بستگی به نحوه پیاده‌سازی و سخت‌افزار دارد.
- برای مثال quick sort از نسبت به radix از حافظه نهفته بهتر استفاده می‌کند.
- radix sort درجا عمل نمی‌کند (در صورت استفاده از counting sort).
- در حالی که اکثر مرتب‌سازی‌های مقایسه‌ای درجا عمل می‌کنند.

مرتب‌سازی سطلی bucket sort

- اطلاعات مازاد: اعداد آرایه در یک فرایند تصادفی با توزیع یکنواخت تولید شده باشد
- ایده اصلی: مشابه مرتب‌سازی شمارشی اما با آزادی عمل بیشتر

n -element array A

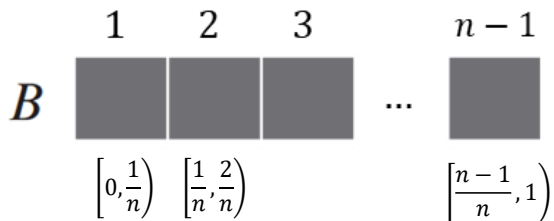
$$0 \leq A[i] < 1$$

توزیع نرمال در بازه $[0,1)$



$B[0 \dots n - 1]$

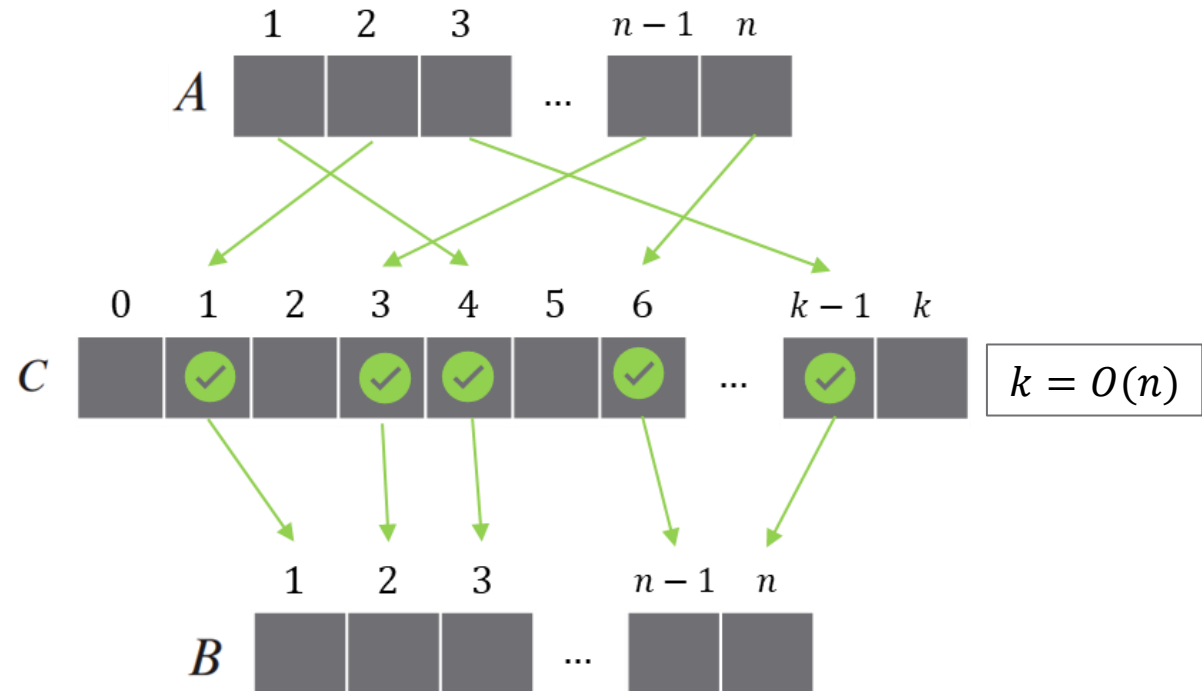
سطل



$A[i]$



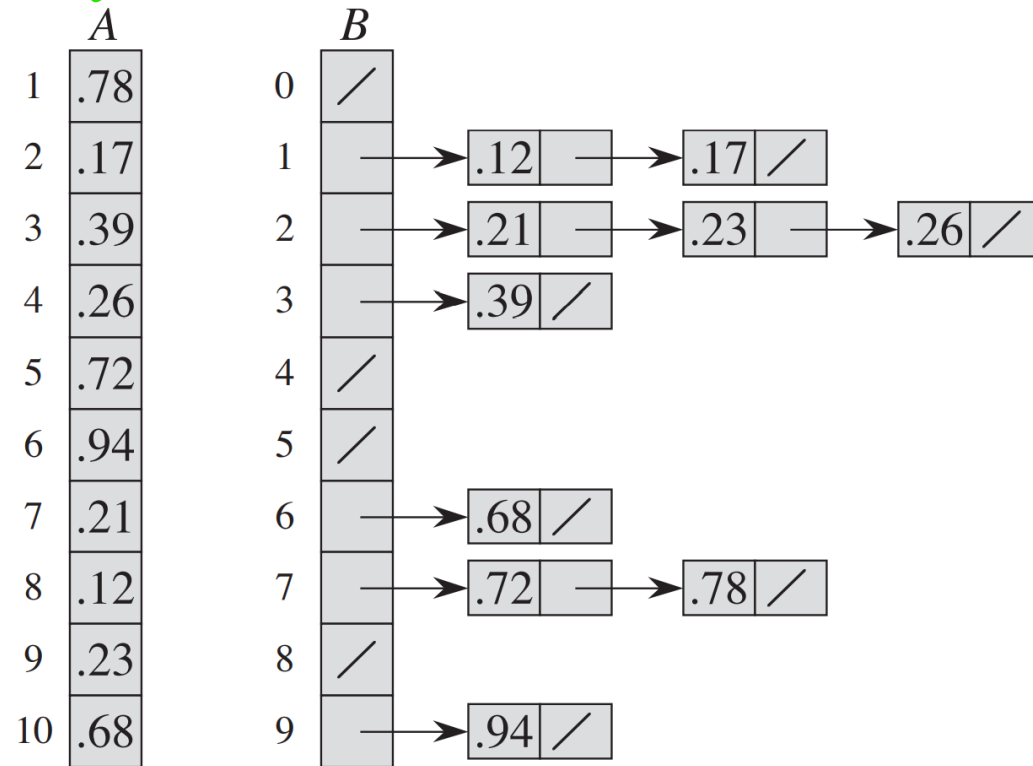
$\lfloor nA[i] \rfloor$



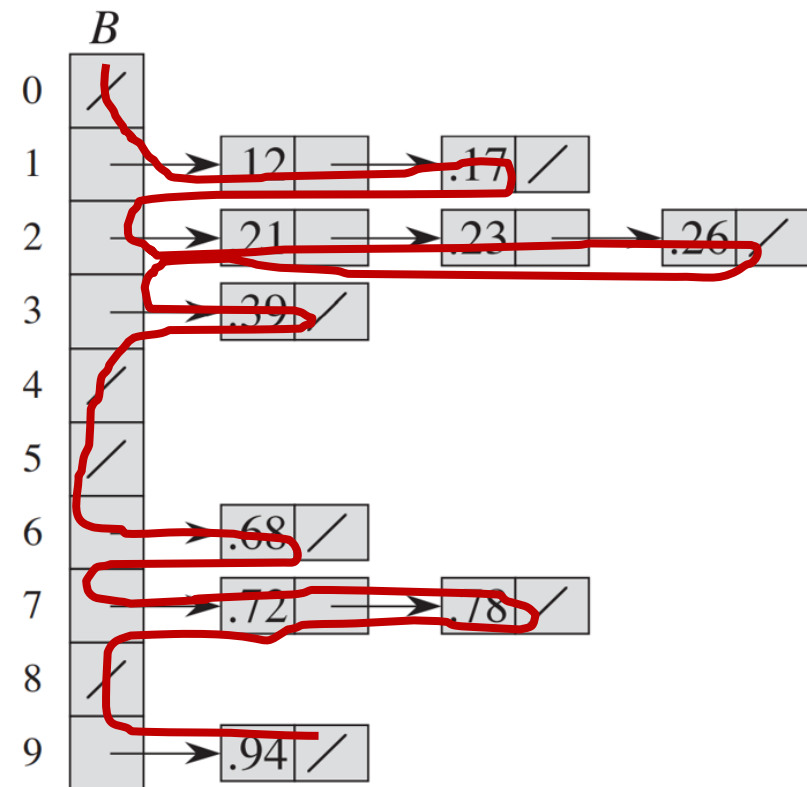
مثال bucket sort

same size

• ساختار آرایه B بصورت linked-list



(b)



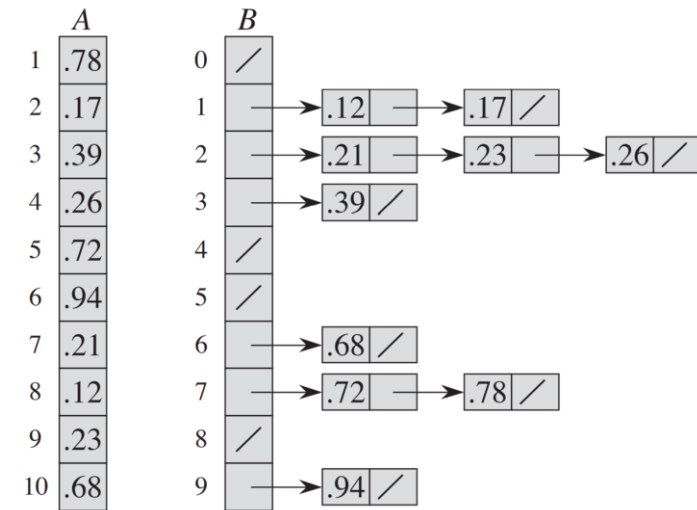
شبهه کد bucket sort

BUCKET-SORT(A)

```

1  let  $B[0..n-1]$  be a new array
2   $n = A.length$ 
3  for  $i = 0$  to  $n - 1$ 
4      make  $B[i]$  an empty list
5  for  $i = 1$  to  $n$ 
6      insert  $A[i]$  into list  $B[\lfloor nA[i] \rfloor]$ 
7  for  $i = 0$  to  $n - 1$ 
8      sort list  $B[i]$  with insertion sort
9  concatenate the lists  $B[0], B[1], \dots, B[n-1]$  together in order
    
```

• ساختار آرایه B بصورت linked-list



$A[i] \leq A[j] \longrightarrow \lfloor nA[i] \rfloor \leq \lfloor nA[j] \rfloor \longrightarrow$
سطل $A[i]$ یا از سطل $A[j]$ کوچکتر است یا برابر

اگر کوچکتر باشد خط ۹ آن را در جایگاه مناسب قرار می‌دهد

اگر برابر باشد خطوط ۷ الی ۸ آن را مرتب می‌کند

تحلیل زمانی bucket sort

- به غیر از خط ۸ شبه شد، همگی با $O(n)$ اجرا می شود
- برای تحلیل زمانی الگوریتم باید مجموع تعداد اجرای خط ۸ محاسبه شود

BUCKET-SORT(A)

```

1  let  $B[0..n-1]$  be a new array
2   $n = A.length$ 
3  for  $i = 0$  to  $n - 1$ 
4      make  $B[i]$  an empty list
5  for  $i = 1$  to  $n$ 
6      insert  $A[i]$  into list  $B[\lfloor nA[i] \rfloor]$ 
7  for  $i = 0$  to  $n - 1$ 
8      sort list  $B[i]$  with insertion sort
9  concatenate the lists  $B[0], B[1], \dots, B[n-1]$  together in order
```

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

n_i متغیر تصادفی نشانگر تعداد المان‌ها در سطل B_i

تحلیل متوسط زمان اجرای bucket sort

```

7  for i = 0 to n - 1
8      sort list B[i] with insertion sort
    
```

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

n_i متغیر تصادفی نشانگر تعداد المان‌ها در سطل B_i

امیدریاضی و برخی خواص آن

$$E(X) = \sum_{i=1}^n p_i x_i$$

تعریف امیدریاضی برای متغیر تصادفی x_i

$$E(aX + b) = aE(X) + b$$

ویژگی خطی بودن امیدریاضی

Letting $g(x) = ax$, we have for any constant a ,

$$E[aX] = aE[X] .$$

$$\begin{aligned}
 E[T(n)] &= E\left[\Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)\right] \\
 &= \Theta(n) + \sum_{i=0}^{n-1} E[O(n_i^2)] \\
 &= \Theta(n) + \sum_{i=0}^{n-1} O(E[n_i^2])
 \end{aligned}$$

تحلیل متوسط زمان اجرای bucket sort

```

7  for i = 0 to n - 1
8      sort list B[i] with insertion sort
    
```

$$E[T(n)] = \Theta(n) + \sum_{i=0}^{n-1} O(E[n_i^2])$$

ثابت خواهیم کرد:

$$E[n_i^2] = 2 - 1/n \text{ for } i = 0, 1, \dots, n-1$$

تعریف یک متغیر تصادفی شاخص جدید:

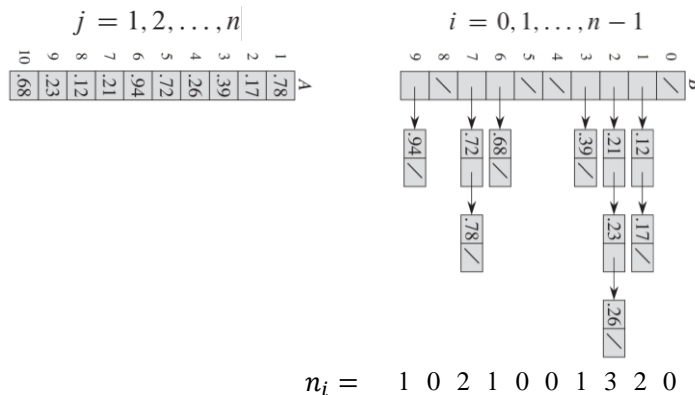
$$X_{ij} = I\{A[j] \text{ falls in bucket } i\}$$

$$n_i = \sum_{j=1}^n X_{ij} \text{ for } i = 0, 1, \dots, n-1 \text{ and } j = 1, 2, \dots, n$$

n_i متغیر تصادفی نشانگر تعداد امانها در سطل B_i

آمار و احتمالات

$$I(A) = \begin{cases} 1, & \text{if } A \text{ happen} \\ 0, & \text{if } A \text{ not happen} \end{cases} \quad \text{متغیر تصادفی شاخص}$$



$$B = \begin{matrix} 1 & 2 & 3 & \dots & n-1 \\ \left[0, \frac{1}{n}\right) & \left[\frac{1}{n}, \frac{2}{n}\right) & \left[\frac{2}{n}, \frac{3}{n}\right) & \dots & \left[\frac{n-1}{n}, 1\right) \end{matrix}$$

تحلیل متوسط زمان اجرای bucket sort

ثابت خواهیم کرد:

$$\boxed{E[n_i^2] = 2 - 1/n} \text{ for } i = 0, 1, \dots, n-1$$

$$\boxed{n_i = \sum_{j=1}^n X_{ij}}$$

$$E[T(n)] = \Theta(n) + \sum_{i=0}^{n-1} O(E[n_i^2])$$

$$\begin{aligned} E[n_i^2] &= E\left[\left(\sum_{j=1}^n X_{ij}\right)^2\right] \\ &= E\left[\sum_{j=1}^n \sum_{k=1}^n X_{ij} X_{ik}\right] \\ &= E\left[\sum_{j=1}^n X_{ij}^2 + \sum_{1 \leq j \leq n} \sum_{\substack{1 \leq k \leq n \\ k \neq j}} X_{ij} X_{ik}\right] \\ &= \sum_{j=1}^n E[X_{ij}^2] + \sum_{1 \leq j \leq n} \sum_{\substack{1 \leq k \leq n \\ k \neq j}} E[X_{ij} X_{ik}] \end{aligned}$$

تحلیل متوسط زمان اجرای bucket sort

ثابت خواهیم کرد:

$$E[n_i^2] = 2 - 1/n \quad \text{for } i = 0, 1, \dots, n-1$$

$$E[T(n)] = \Theta(n) + \sum_{i=0}^{n-1} O(E[n_i^2])$$

امیدریاضی و برخی خواص آن

$$E(X) = \sum_{i=1}^n p_i x_i$$

تعریف امیدریاضی برای متغیر تصادفی x_i

$$E(XY) = E(X)E(Y) \quad \text{برای متغیر تصادفی مستقل } X \text{ و } Y \text{ داریم:}$$

$$E[n_i^2] = \sum_{j=1}^n E[X_{ij}^2] + \sum_{1 \leq j \leq n} \sum_{\substack{1 \leq k \leq n \\ k \neq j}} E[X_{ij} X_{ik}]$$

$$\begin{aligned} E[X_{ij}^2] &= 1^2 \cdot \frac{1}{n} + 0^2 \cdot \left(1 - \frac{1}{n}\right) \\ &= \frac{1}{n} \end{aligned}$$

When $k \neq j$, the variables X_{ij} and X_{ik} are independent

$$\begin{aligned} E[X_{ij} X_{ik}] &= E[X_{ij}] E[X_{ik}] \\ &= \frac{1}{n} \cdot \frac{1}{n} \\ &= \frac{1}{n^2} \end{aligned}$$

تحلیل متوسط زمان اجرای bucket sort

ثابت خواهیم کرد:

$$\boxed{E[n_i^2] = 2 - 1/n \text{ for } i = 0, 1, \dots, n-1}$$

$$E[T(n)] = \Theta(n) + \sum_{i=0}^{n-1} O(E[n_i^2])$$

$$E[n_i^2] = \sum_{j=1}^n E[X_{ij}^2] + \sum_{1 \leq j \leq n} \sum_{\substack{1 \leq k \leq n \\ k \neq j}} E[X_{ij} X_{ik}]$$

$$\begin{aligned} E[n_i^2] &= \sum_{j=1}^n \frac{1}{n} + \sum_{1 \leq j \leq n} \sum_{\substack{1 \leq k \leq n \\ k \neq j}} \frac{1}{n^2} \\ &= n \cdot \frac{1}{n} + n(n-1) \cdot \frac{1}{n^2} \\ &= 1 + \frac{n-1}{n} \\ &= 2 - \frac{1}{n}, \end{aligned}$$

$$\boxed{E[T(n)] = \Theta(n) + \sum_{i=0}^{n-1} O\left(2 - \frac{1}{n}\right) = \Theta(n)}$$