طراحی الگوریتم‌ها – مرتب‌سازی خطی

Introduction to Algorithm

مهدی جوانمردی

۱۴۰۱

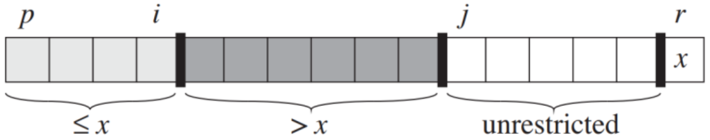
مرور جلسه قبل

پیاده سازی PARTITION

$\text{PARTITION}(A, p, r)$

```
x = A[r]
i = p − 1
for j = p to r − 1
      if A[j] ≤ x
            i = i + 1
            exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```

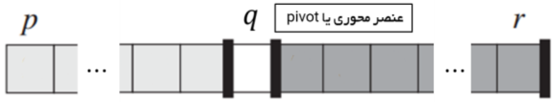
- انتخاب آخرین عنصر به عنوان pivot
- تغییر چیدمان عناصر آرایه با رعایت قوانین چهار بخش
- قرارداد pivot در جایگاه مناسب



QUICK SORT

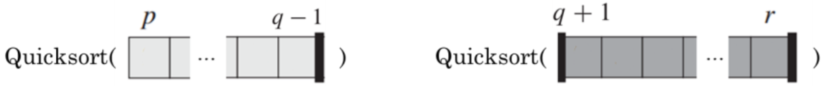
Partition (rearrange) the array $A[p..r]$

تقسیم



$A[p..q-1] \le A[q] \le A[q+1..r]$

حل



ترکیب

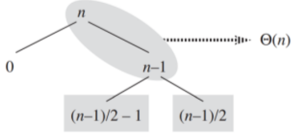
$A[p..r]$ is now sorted

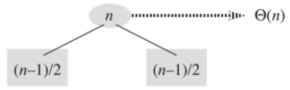
تحلیل زمانی شهودی مرتب‌سازی سریع

$$\begin{aligned} T(n) &= T(n-1) + T(0) + \Theta(n) \\ &= T(n-1) + \Theta(n)\,. \end{aligned} \longrightarrow \begin{aligned} \sum_{k=1}^{n} k &= \frac{1}{2}n(n+1) \\ &= \Theta(n^2)\,. \end{aligned} \longrightarrow \Theta(n^2)$$

$$T(n) = 2T(n/2) + \Theta(n)\,, \longrightarrow \boxed{\text{حالت دوم قضیه اصلی}} \longrightarrow T(n) = \Theta(n \lg n)$$

$$T(n) = T(9n/10) + T(n/10) + cn$$





پیاده‌سازی مرتب‌سازی سریع

$\text{QUICKSORT}(A, p, r)$

```
if p < r
      q = PARTITION(A, p, r)
      QUICKSORT(A, p, q − 1)
      QUICKSORT(A, q + 1, r)
```

اولین فراخوانی تابع:

$\boxed{\text{QUICKSORT}(A, 1, A.length).}$

مرتب‌سازی مقایسه ای

- الگوریتم‌های معرفی شده تا الان ← مرتب‌سازی n عدد در $O(n\lg n)$
- مرتب‌سازی ادغامی و هرمی در بدترین حالت در $O(n\lg n)$
- مرتب‌سازی سریع در حالت متوسط در $O(n\lg n)$
- خاصیت مشترک الگوریتم‌های معرفی شده تا کنون:

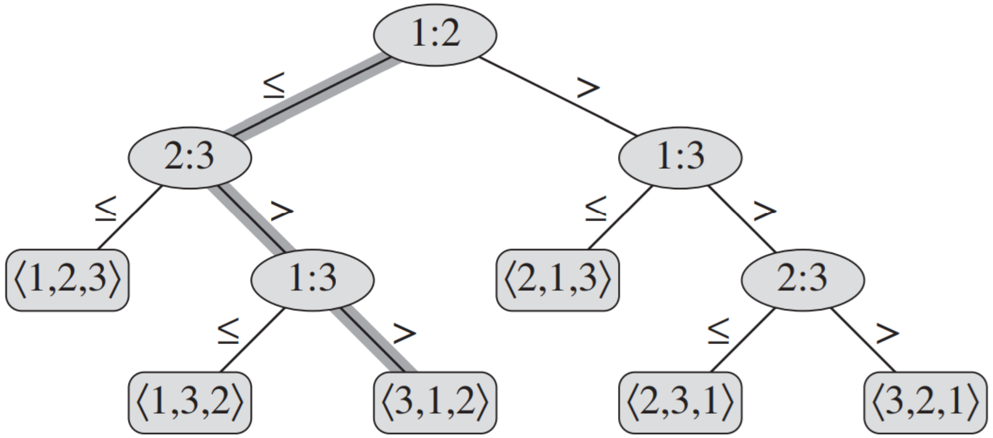
مرتب‌سازی اعداد صرفا بر اساس مقایسه المان‌ها با یکدیگر ← مرتب‌سازی مقایسه‌ای ← بدترین زمان اجرا $\Omega(n \lg n)$

عدم دسترسی به هرگونه اطلاعات مازاد در مورد خود اعداد

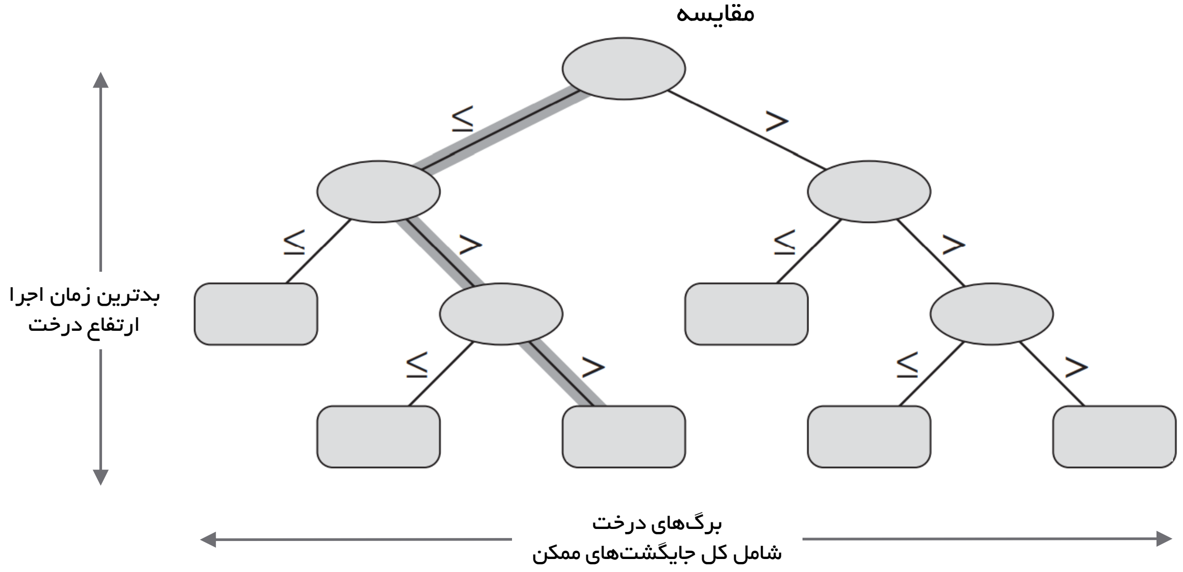
تحلیل رفتار مرتب‌سازی مقایسه با استفاده از درخت تصمیم‌گیری دودویی

بهترین بدترین زمان اجرای مرتب‌سازی مقایسه‌ای

درخت تصمیم‌گیری برای مرتب‌سازی درجی با ۳ عدد



درخت تصمیم‌گیری برای الگوریتم‌های مرتب‌سازی



Theorem 8.1
Any comparison sort algorithm requires $\Omega(n \lg n)$ comparisons in the worst case.

تعداد کل جایگشت‌ها n عدد

ماکسیمم برگ برای ارتفاع h

$$n! \le l \le 2^h \xrightarrow{\text{لگاریتم از طرفین}} h \ge \lg(n!) = \Omega(n \lg n)$$

تعداد برگ‌ها

Stirling's approximation

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n\left(1 + \Theta\left(\frac{1}{n}\right)\right)$$

مرتب‌سازی خطی

- درصورت وجود برخی اطلاعات در مورد اعداد می‌توان از حد $\Omega(n \lg n)$ عبور کرد

- الگوریتم Counting sort

the input numbers are in the set $\{0, 1, \ldots, k\}$

worst-case running time	$\Theta(k + n)$	مرتبه زمانی
expected running time	$\Theta(k + n)$	

- الگوریتم Radix sort

there are n integers to sort

integer has d digits

digit can take on up to k possible values

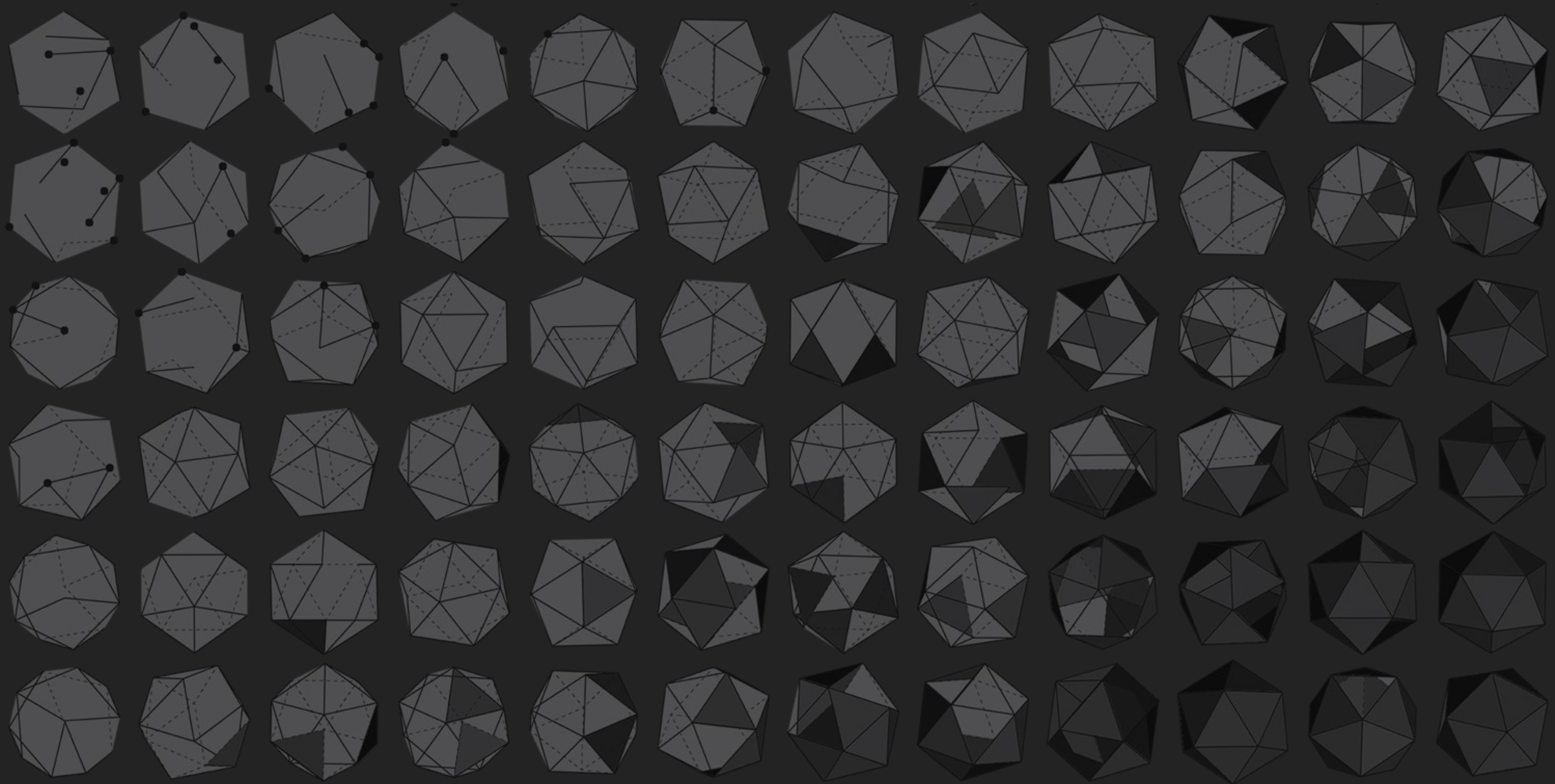
worst-case running time	$\Theta(d(n + k))$	مرتبه زمانی
expected running time	$\Theta(d(n + k))$	

- الگوریتم Bucket sort

requires knowledge of the probabilistic distribution of numbers in the input array

real numbers uniformly distributed in the half-open interval $[0, 1)$

worst-case running time	$\Theta(n^2)$	مرتبه زمانی
average-case running time	$\Theta(n)$	

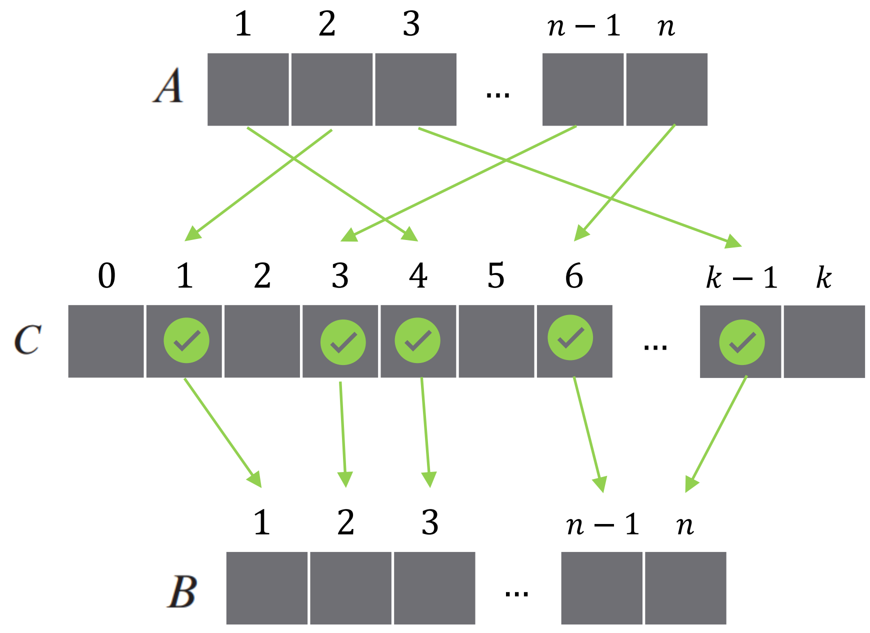


مرتب‌سازی شمارشی Counting Sort

مرتب‌سازی شمارشی counting sort

- اطلاعات مازاد: اعداد داخل آرایه اعداد حسابی ۰ تا k
- زمان اجرا: در صورتی که $k = O(n)$ باشد زمان اجرای counting sort $\theta(n)$ خواهد بود

با فرض تکراری نبودن اعداد



$k = O(n)$

در حال کلی و وجود اعداد تکراری؟

- ایده کار: شمارش تعداد اعداد کوچکتر برای هر عدد و قراردادن آن مستقیماً در جایگاه خود

counting sort پیاده‌سازی

```
COUNTING-SORT(A, B, k)
let C[0..k] be a new array
for i = 0 to k
    C[i] = 0
for j = 1 to A.length
    C[A[j]] = C[A[j]] + 1
// C[i] now contains the number of elements equal to i.
for i = 1 to k
    C[i] = C[i] + C[i − 1]
// C[i] now contains the number of elements less than or equal to i.
for j = A.length downto 1
    B[C[A[j]]] = A[j]
    C[A[j]] = C[A[j]] − 1
```

ورودی	$A[1..n]$
مرتب	$B[1..n]$
موقت	$C[0..k]$

	0	1	2	3	4	5	6		$k-1$	k
C	0	1	1	3	4	4	7	…	12	12

counting sort مثال

(a)

	1	2	3	4	5	6	7	8
A	2	5	3	0	2	3	0	3

	0	1	2	3	4	5
C	2	0	2	3	0	1

(b)

	0	1	2	3	4	5
C	2	2	4	7	7	8

(c)

	1	2	3	4	5	6	7	8
B							3	

	0	1	2	3	4	5
C	2	2	4	6	7	8

(d)

	1	2	3	4	5	6	7	8
B		0					3	

	0	1	2	3	4	5
C	1	2	4	6	7	8

(e)

	1	2	3	4	5	6	7	8
B		0				3	3	

	0	1	2	3	4	5
C	1	2	4	5	7	8

(f)

	1	2	3	4	5	6	7	8
B	0	0	2	2	3	3	3	5

تحلیل زمان اجرای counting sort

```
COUNTING-SORT(A, B, k)
let C[0..k] be a new array
for i = 0 to k
    C[i] = 0
for j = 1 to A.length
    C[A[j]] = C[A[j]] + 1
// C[i] now contains the number of elements equal to i.
for i = 1 to k
    C[i] = C[i] + C[i-1]
// C[i] now contains the number of elements less than or equal to i.
for j = A.length downto 1
    B[C[A[j]]] = A[j]
    C[A[j]] = C[A[j]] - 1
```

$\theta(k)$

$\theta(n)$

$\theta(k)$

$\theta(n)$

$\theta(n+k)$

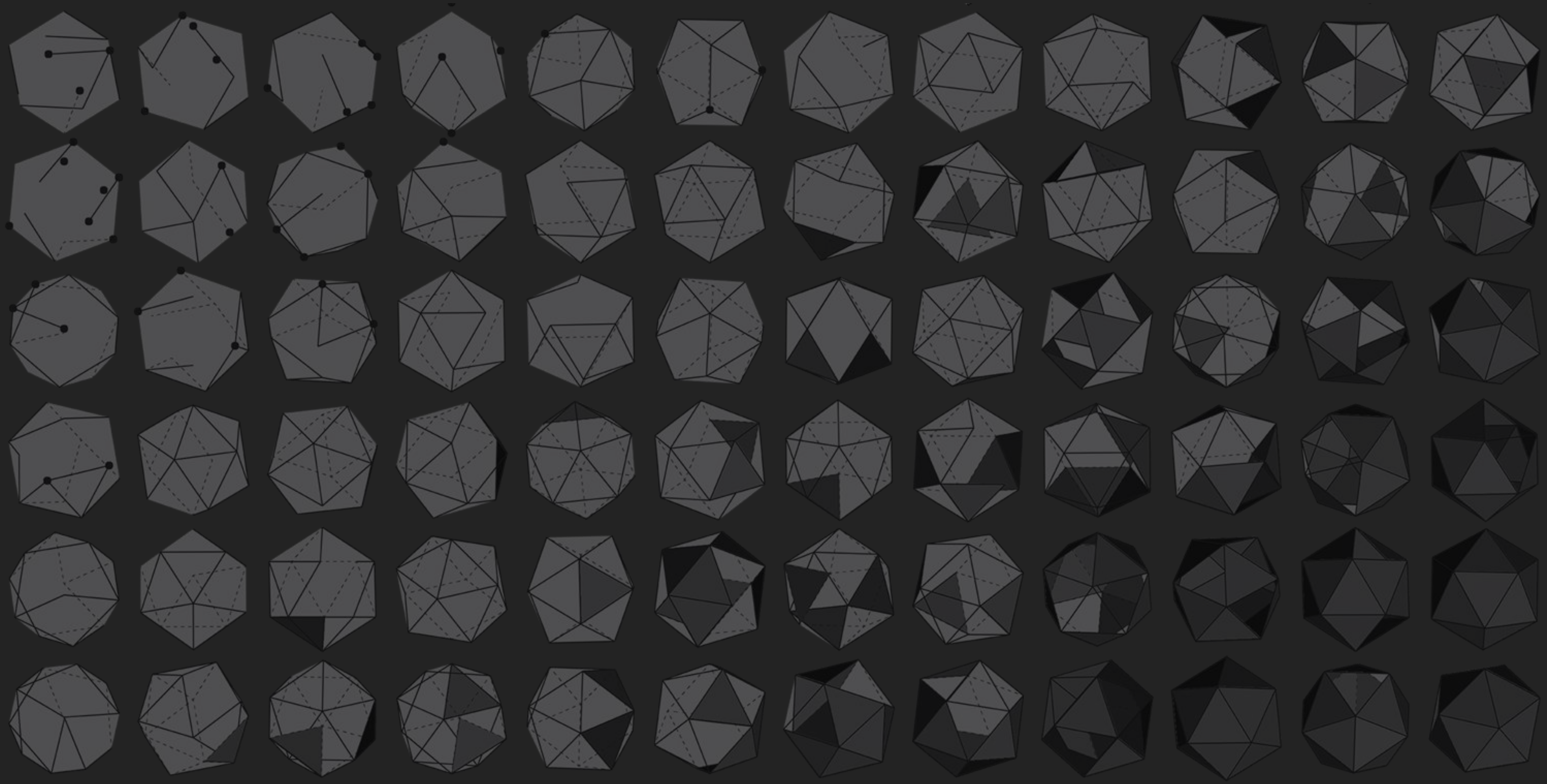
$k = O(n)$

$\theta(n)$

دلیل اینکه زمان اجرای counting sort توانست از حد $\Omega(n\lg n)$ عبور کند اینست که counting sort یک مرتب‌سازی مقایسه ای نیست

در عوض counting sort از مقدار خود اعداد بصورت مستقیم برای تعیین محلشان استفاده میکند

ویژگی پایداری: ترتیب اعداد برابر در آرایه ورودی و آرایه مرتب شده تغییر پیدا نمی کند



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

INTRODUCTION TO ALGORITHM | درس طراحی الگوریتم (ترم اول ۱۳۹۹)

مرتب‌سازی مبنایی Radix Sort

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	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؟

d رقم که هر رقم k مقدار متفاوت

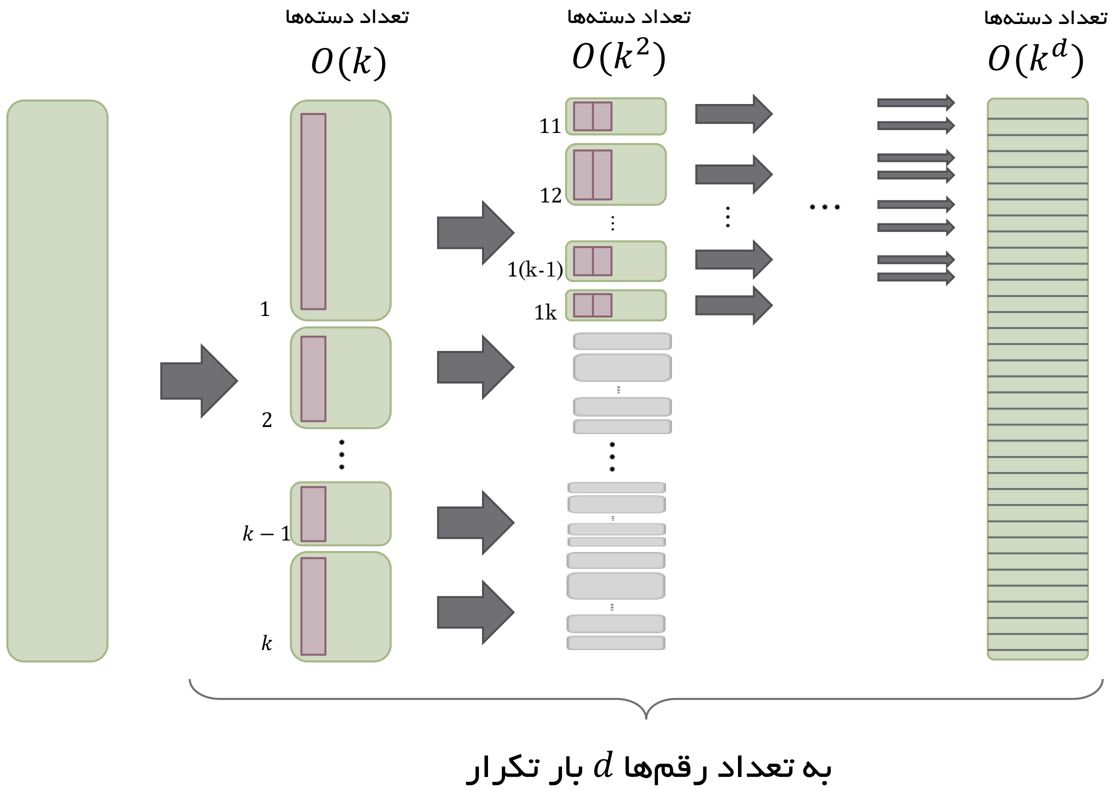
مرتب‌سازی مبنایی radix sort

روش شهودی – غلط

- مرتب‌سازی از رقم با ارزش بیشتر
- در بدترین حالت چندبار تابع sort فراخوانی می‌شود؟ $\Theta(k^{d-1})$

d رقم که هر رقم k مقدار متفاوت

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



تعداد فراخوان sort در بدترین حالت

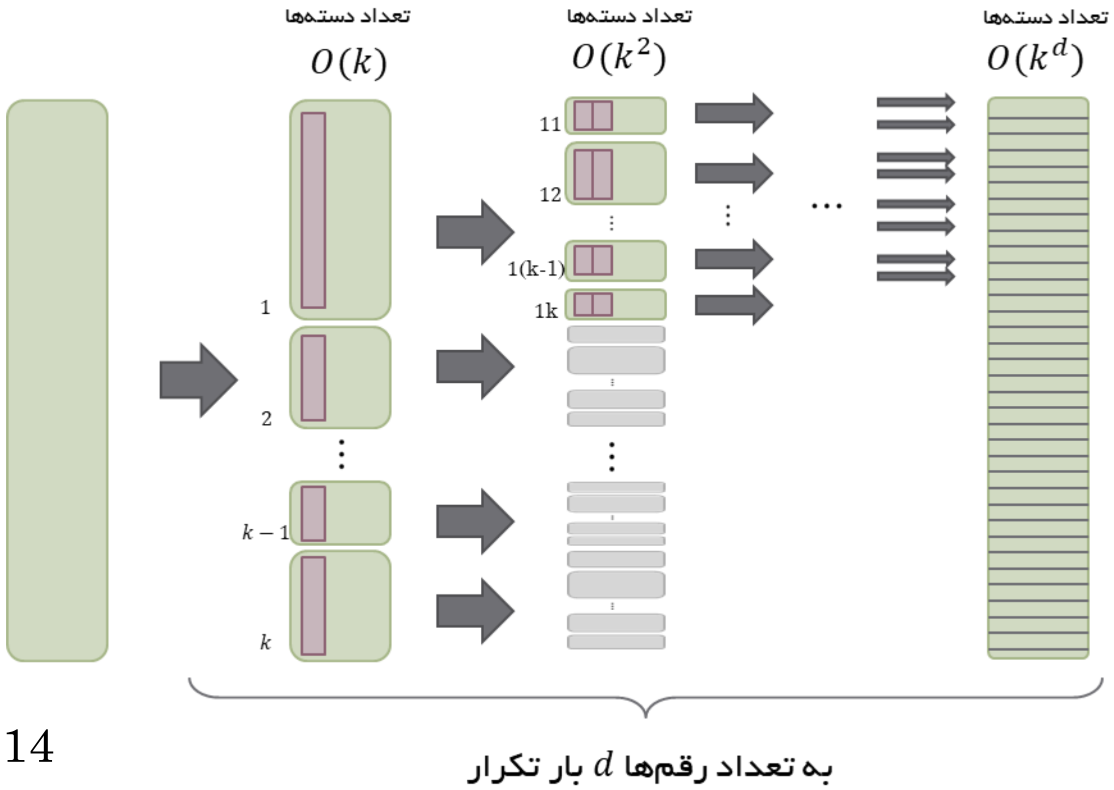
$$\leq \sum_{i=0}^{d-1} k^i = \frac{k^d - 1}{k-1} = \Theta(k^{d-1})$$

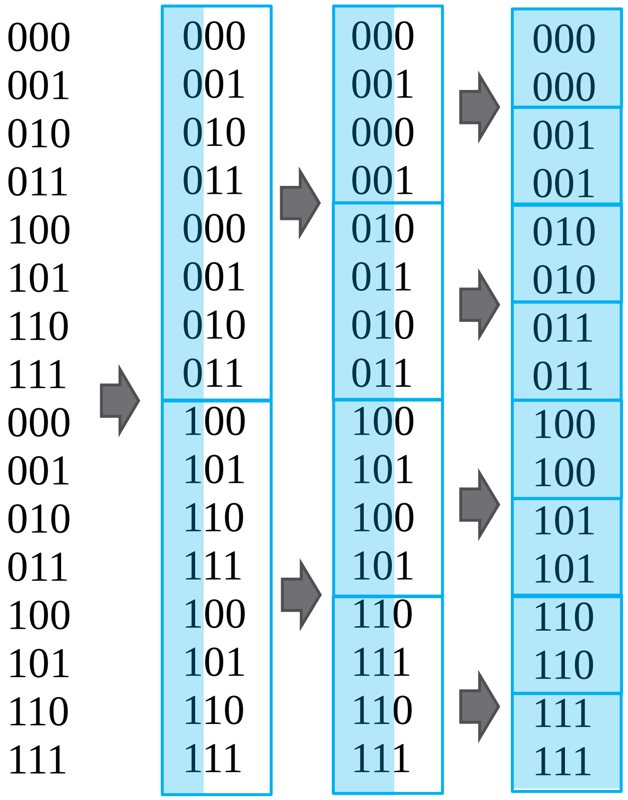
مرتب‌سازی مبنایی radix sort

- در بدترین حالت چندبار تابع sort فراخوانی می‌شود؟

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

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





مرتب‌سازی مبنایی radix sort

- در بدترین حالت چندبار تابع sort فراخوانی می‌شود؟

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

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

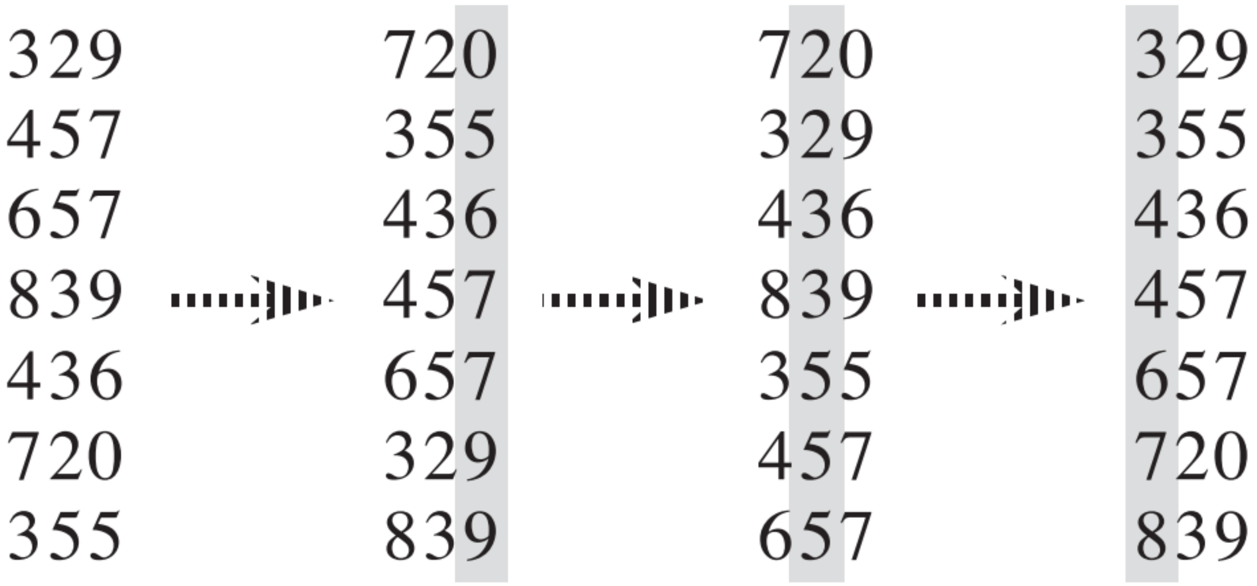
برای حل الگوریتم، می بایست در هر لحظه ماکسیمم اطلاعات چند دسته در حافظه نگهداری شود؟

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

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

329		720		720		329
457		355		329		355
657		436		436		436
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 دقیق‌تر مرتبه زمانی radix sort

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

$$\boxed{\text{زمان اجرایی } \mathrm{radix\ sort} \text{ برای این اعداد } \theta((b/r)(n+2^r)) \text{ خواهد بود}}$$

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

- اثبات:

برای $r \leq b$ فرض کیند که هر رقم r بیتی باشد و تعداد ارقام برابر $d = \lceil b/r \rceil$

در این صورت هر رقم یک عدد صحیح بین ۰ 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 $\mathrm{radix\ sort}$ 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

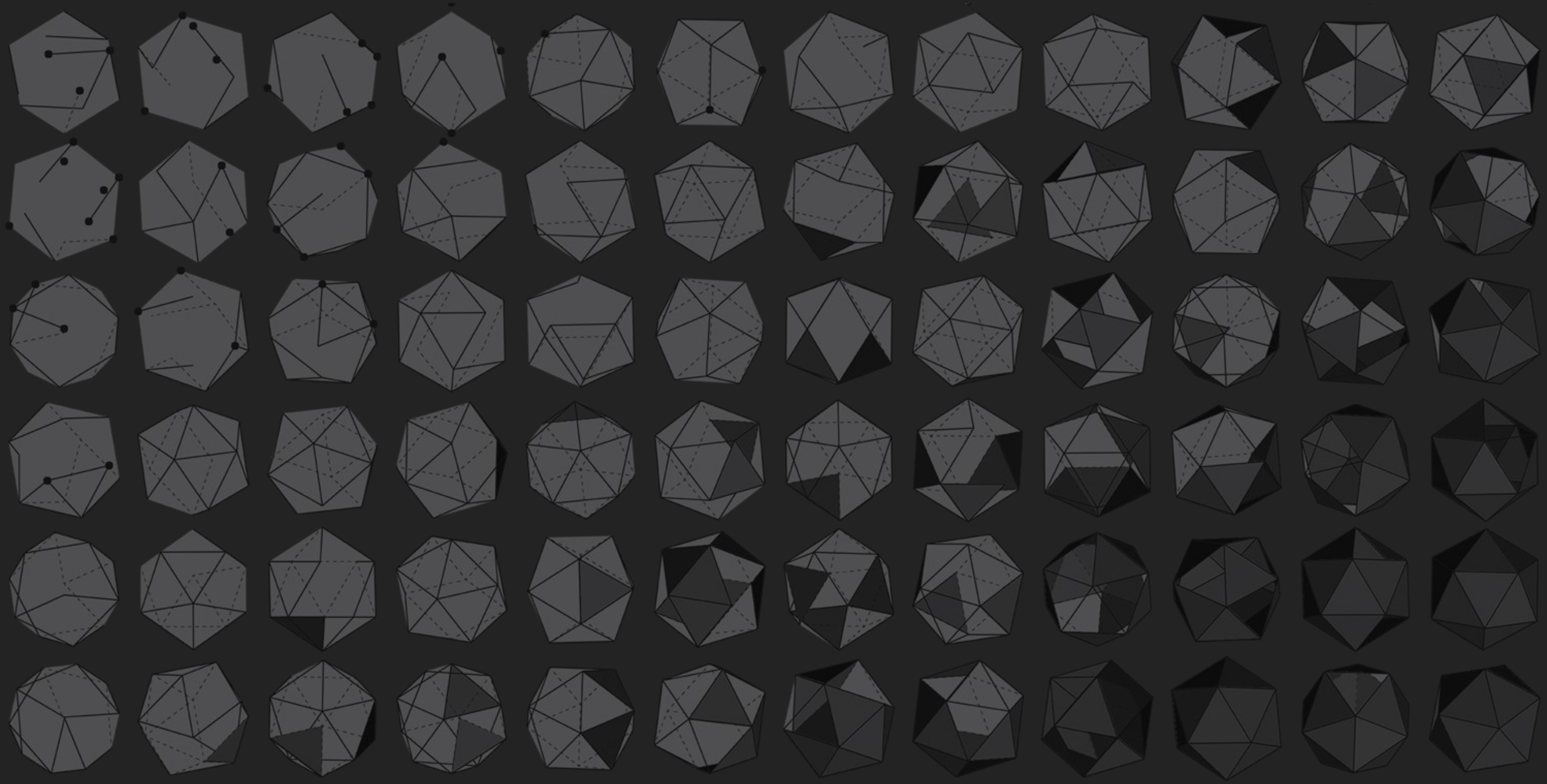
حالت اول: $b < \lfloor \lg n \rfloor$ ← $(n+2^r) = \Theta(n)$ ← برای $r = b$ خواهیم داشت: $(b/b)(n+2^b) = \Theta(n)$

حالت دوم: $b \geq \lfloor \lg n \rfloor$ ← 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؟

- درصورتی که $b = O(\lg n)$ باشد با انتخاب $r = \lfloor \lg n \rfloor$ زمان اجری radix sort برابر $O(n)$
- در این صورت، تعداد فراخوان‌های radix sort به مراتب کمتر از quick sort خواهد بود درحالی که زمان اجرای هر فراخوان radix sort به مراتب طولانی تر 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 (درصورت استفاده از counting sort)
- در حالی که اکثر مرتب‌سازی‌های مقایسه‌ای درجا عمل میکنند



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

INTRODUCTION TO ALGORITHM | درس طراحی الگوریتم (ترم اول ۱۳۹۹)

مرتب‌سازی سطلی Bucket Sort

مرتب‌سازی سطلی bucket sort

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

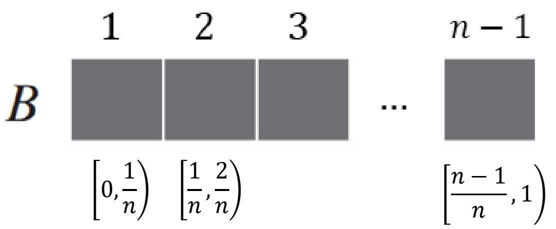
n-element array A

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

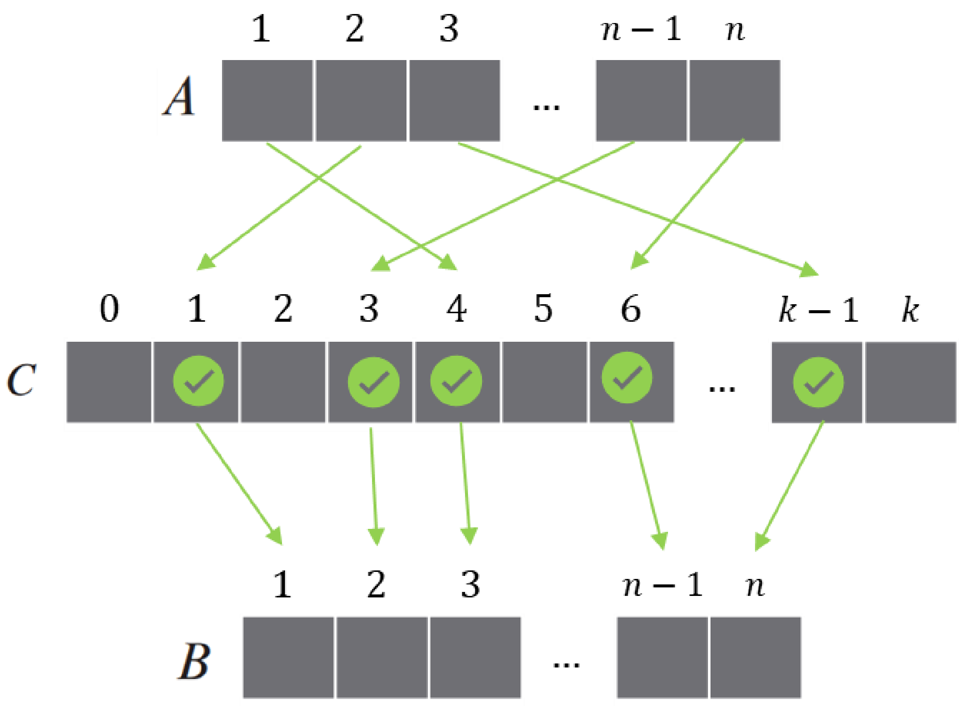
توزیع نرمال در بازه [0,1)

$$B[0\,..\,n-1]$$

سطل



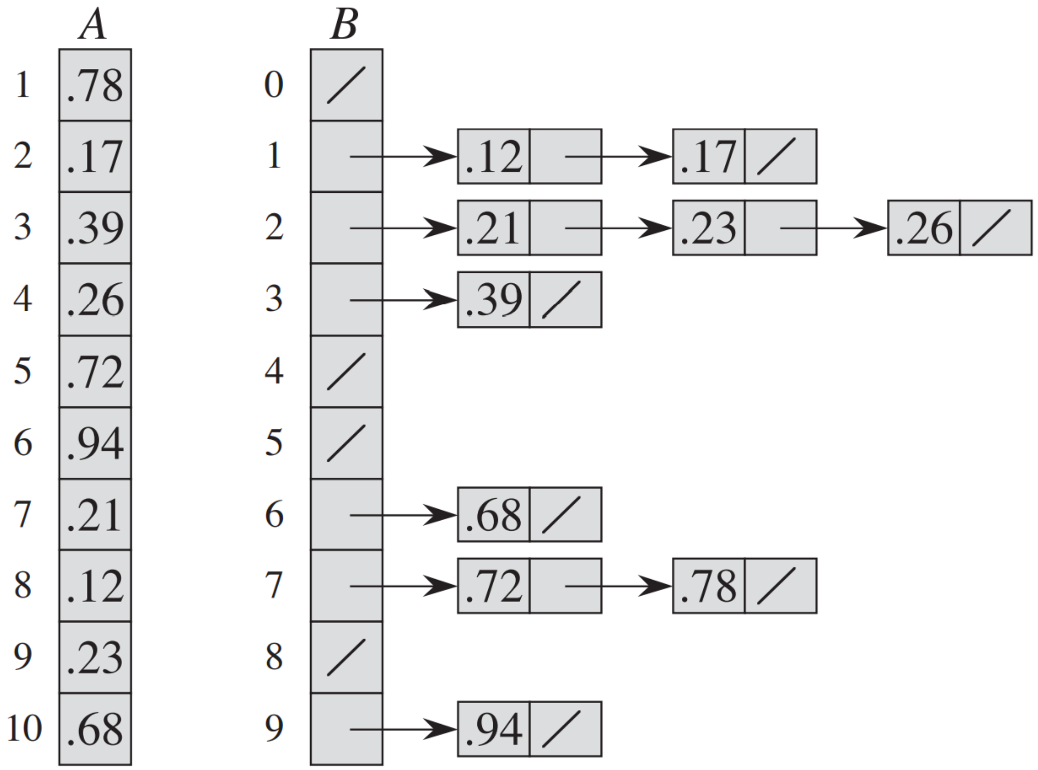
$$A[i] \rightarrow \lfloor nA[i] \rfloor$$



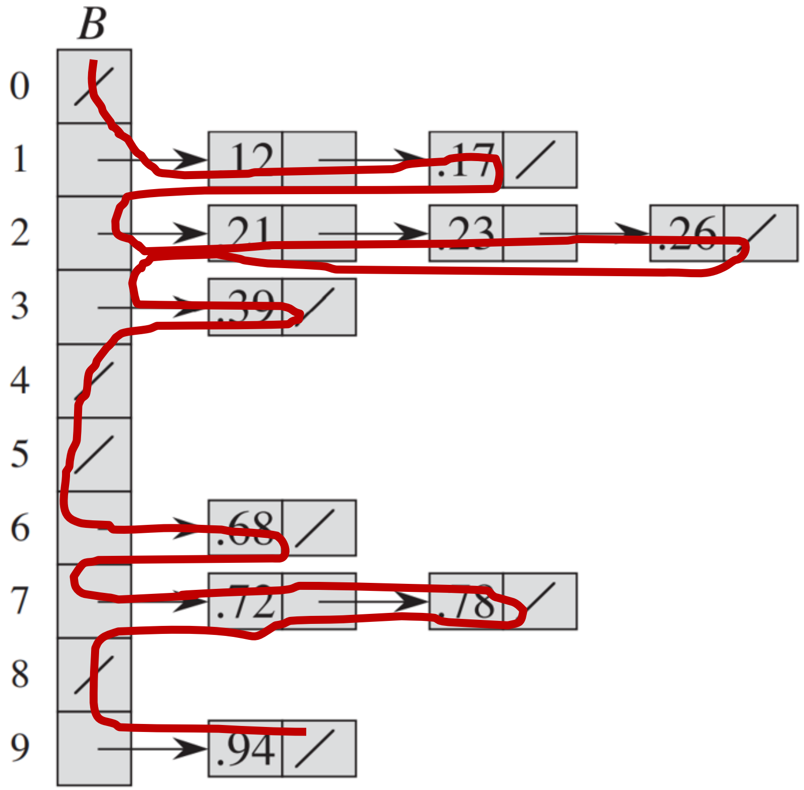
$k = O(n)$

مثال bucket sort

- ساختار آرایه B بصورت linked-list



(a) (b)

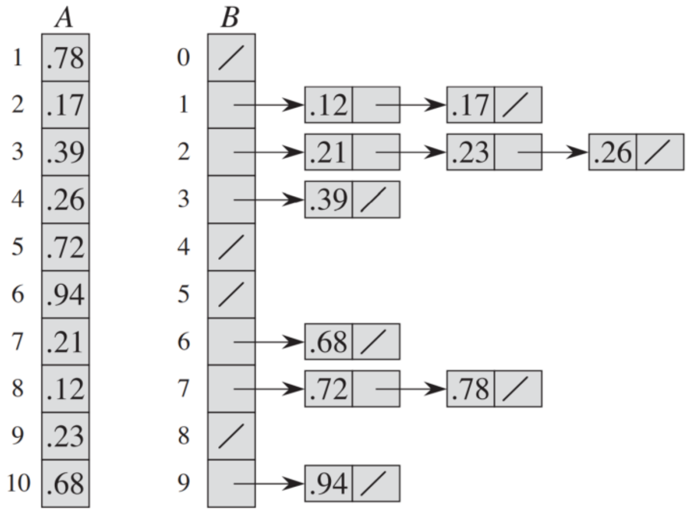


شبه کد bucket sort

- ساختار آرایه B بصورت linked-list

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[⌊n A[i]⌋]
7  for i = 0 to n − 1
8       sort list B[i] with insertion sort
9  concatenate the lists B[0], B[1], ..., B[n − 1] together in order
```



$A[i] \le A[j] \longrightarrow \lfloor nA[i] \rfloor \le \lfloor nA[j] \rfloor \longrightarrow$ 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 می کند

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[⌊nA[i]⌋]
7  for i = 0 to n-1
8      sort list B[i] with insertion sort
9  concatenate the lists B[0], B[1], ..., B[n-1] together in order
```

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

n_i متغیر تصادفی نشانگر تعداد المان‌ها در سطل B_i

تحلیل متوسط زمان اجرای bucket sort

for $i = 0$ **to** $n - 1$
    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

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

امیدریاضی و برخی خواص آن

تعریف امیدریاضی برای متغیر تصادفی x_i

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

ویژگی خطی بودن امیدریاضی

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

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

$$\mathrm{E}[aX] = a\mathrm{E}[X] .$$

تحلیل متوسط زمان اجرای bucket sort

```
for i = 0 to n − 1
        sort list B[i] with insertion sort
```

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

ثابت خواهیم کرد:

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

آمار و احتمالات

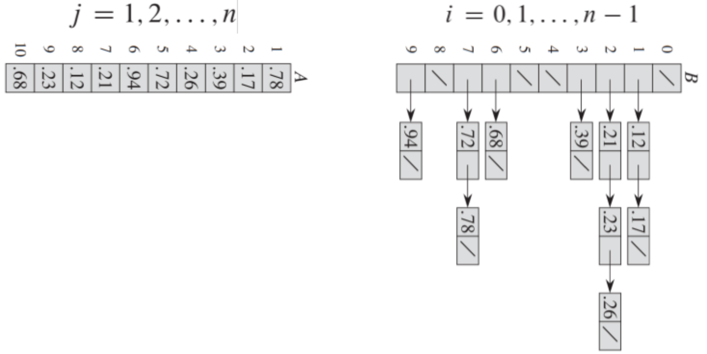
متغیر تصادفی شاخص

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

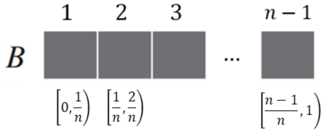
تعریف یک متغیر تصادفی شاخص جدید:

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

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



$n_i =$ 1 0 2 1 0 0 1 3 2 0



متغیر تصادفی n_i نشانگر تعداد المان‌ها در سطل B_i

تحلیل متوسط زمان اجرای bucket sort

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

ثابت خواهیم کرد:

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

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

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

تحلیل متوسط زمان اجرای bucket sort

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

ثابت خواهیم کرد:

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

$$E\left[n_i^2\right] = \sum_{j=1}^{n} E\left[X_{ij}^2\right] + \sum_{1 \le j \le n} \sum_{\substack{1 \le k \le n \\ k \ne j}} E\left[X_{ij} X_{ik}\right]$$

امیدریاضی و برخی خواص آن

تعریف امیدریاضی برای متغیر تصادفی x_i

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

برای متغیر تصادفی مستقل X و Y داریم:

$$E(XY) = E(X)E(Y)$$

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

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

$$E\left[X_{ij} X_{ik}\right] = E\left[X_{ij}\right] E\left[X_{ik}\right] = \frac{1}{n} \cdot \frac{1}{n} = \frac{1}{n^2} .$$

تحلیل متوسط زمان اجرای bucket sort

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

ثابت خواهیم کرد:

$$\boxed{\mathrm{E}\left[n_i^2\right] = 2 - 1/n} \quad \text{for } i = 0, 1, \ldots, n-1$$

$$\mathrm{E}\left[n_i^2\right] = \sum_{j=1}^{n} \mathrm{E}\left[X_{ij}^2\right] + \sum_{1 \le j \le n} \sum_{\substack{1 \le k \le n \\ k \ne j}} \mathrm{E}\left[X_{ij} X_{ik}\right]$$

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

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