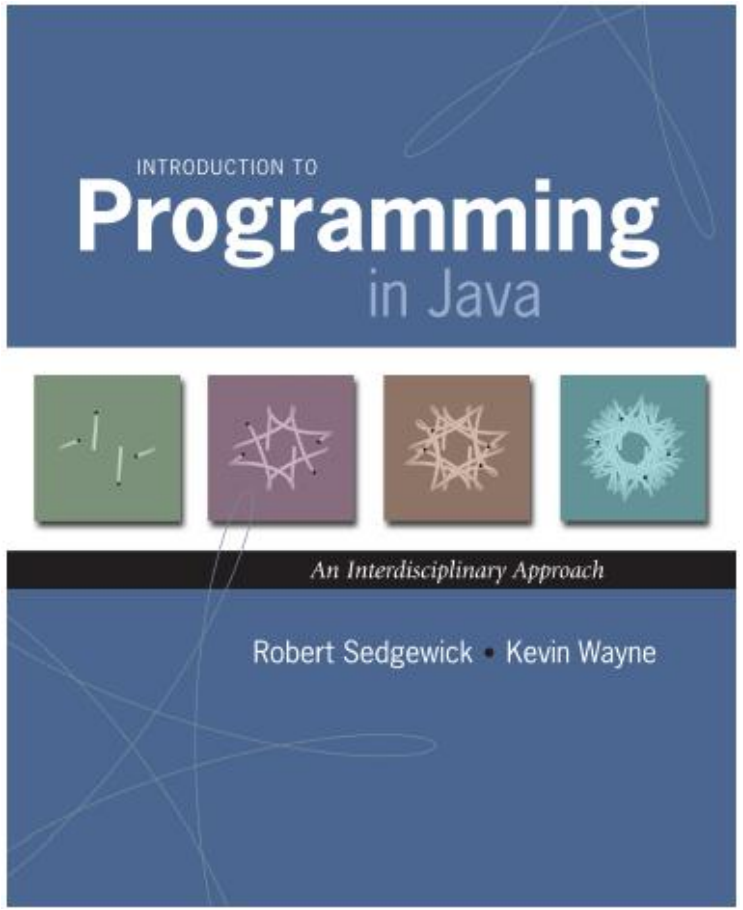
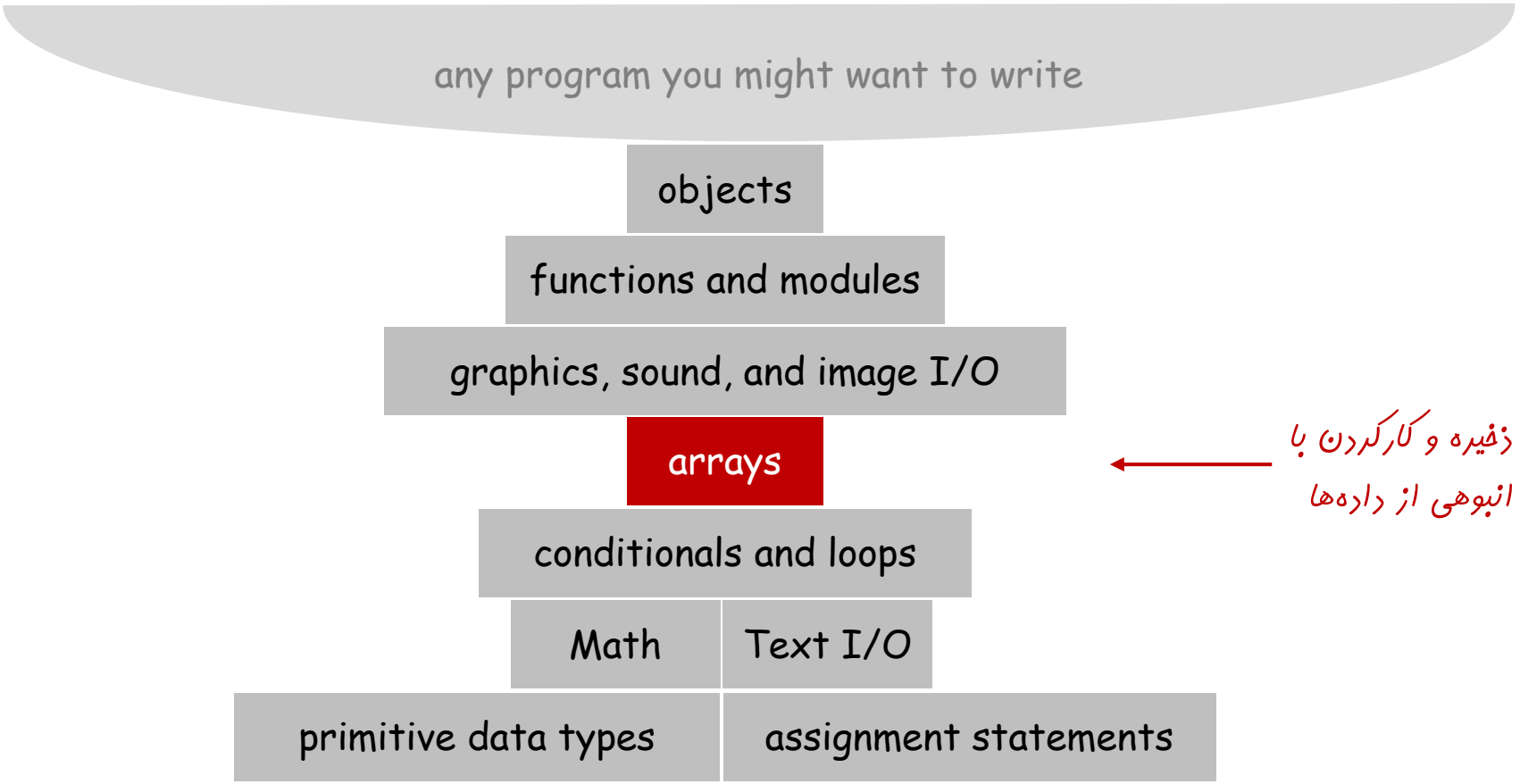
عناصر برنامه‌نویسی:
آرایه‌ها

سید ناصر رضوی **www.snrazavi.ir**
۱۳۹۶

۴-۱ آرایه‌ها


INTRODUCTION TO
Programming
in Java
An Interdisciplinary Approach
Robert Sedgewick • Kevin Wayne

اجزای برنامه‌نویسی



آرایه‌ها

- آرایه. یک دنباله اندیس گذاری شده از مقادیر همنوع.

- مثال‌ها.
  - ۵۲ کارت بازی در یک دسته ورق.
  - ۵ هزار دانشجوی کارشناسی در دانشگاه تبریز
  - ۱ میلیون کاراکتر در یک کتاب
  - ۱۰ میلیون نمونه صوتی در یک فایل MP3
  - ۴ میلیارد نوکلئوتید در یک رشته DNA
  - ۷۳ میلیارد پرس و جوی گوگل در یک سال
  - ۵۰ تریلیون سلول در بدن انسان

index	value
0	razavi
1	rs
2	doug
3	dgabai
4	maia
5	llp
6	funk
7	blei

ذخیره تعداد زیادی متغیر از یک نوع

- هدف. ذخیره ۱۰ متغیر همنوع.

```
// tedious and error-prone
double a0, a1, a2, a3, a4, a5, a6, a7, a8, a9;
a0 = 0.0;
a1 = 0.0;
a2 = 0.0;
a3 = 0.0;
a4 = 0.0;
a5 = 0.0;
a6 = 0.0;
a7 = 0.0;
a8 = 0.0;
a9 = 0.0;
…
a4 = 3.0;
…
a8 = 8.0;
…
double x = a4 + a8;
```

ذخیره تعداد زیادی متغیر از یک نوع

- هدف. ذخیره ۱۰ متغیر همنوع.

```
// easy alternative
double[] a = new double[10];
…
a[4] = 3.0;
…
a[8] = 8.0;
…
double x = a[4] + a[8];
```

اعلان، ایجاد و مقداردهی آرایه

ذخیره تعداد زیادی متغیر از یک نوع

- هدف. ذخیره ۱ میلیون متغیر همنوع.

```
// easy alternative
double[] a = new double[1000000];
…
a[123456] = 3.0;
…
a[987654] = 8.0;
…
double x = a[123456] + a[987654];
```

اعلان، ایجاد و مقداردهی آرایه

آرایه‌ها در جاوا

- زبان جاوا ویژگی‌های خاصی به منظور پشتیبانی از آرایه‌ها دارد.
  - ساختن آرایه: اعلان، ایجاد و مقداردهی اولیه
  - دسترسی به عنصر `i` از آرایه `a`: استفاده از `a[i]`
  - در جاوا اندیس آرایه‌ها از صفر شروع می‌شوند.

```
int N = 10;                     // size of array
double[] a;                     // declare the array
a = new double[N];              // create the array
for (int i = 0; i < N; i++)     // initialize the array
   a[i] = 0.0;                  // all to 0.0
```

- روش فشرده.
  - اعلان، ایجاد و مقداردهی اولیه به آرایه در یک دستور.
  - مقداردهی پیش‌فرض: همه اعداد به صورت خودکار مقدار صفر می‌گیرند.

```
int N = 10;                     // size of array
double[] a = new double[N];     // declare, create, init
```

ضرب نقطه‌ای دو بردار

- ضرب نقطه‌ای. ضرب نقطه‌ای دو بردار x[] و y[] هر یک به طول N برابر است با مجموع حاصل‌ضرب عناصر متناظر در دو بردار.

```
double[] x = { 0.3, 0.6, 0.1 };
double[] y = { 0.5, 0.1, 0.4 };
int N = x.length;
double sum = 0.0;
for (int i = 0; i < N; i++) {
   sum = sum + x[i] * y[i];
}
```

i	x[i]	y[i]	x[i]*y[i]	sum
				0
0	.30	.50	.15	.15
1	.60	.10	.06	.21
2	.10	.40	.04	.25
				.25

چند مثال از پردازش آرایه

create an array with random values	`double[] a = new double[N];`
	`for (int i = 0; i < N; i++)`
	`  a[i] = Math.random();`

print the array values, one per line	`for (int i = 0; i < N; i++)` `  System.out.println(a[i]);`

find the maximum of the array values	`double max = Double.NEGATIVE_INFINITY;`
	`for (int i = 0; i < N; i++)`
	`  if (a[i] > max) max = a[i];`

compute the average of the array values	`double sum = 0.0;`
	`for (int i = 0; i < N; i++)`
	`  sum += a[i];`
	`double average = sum / N;`

copy to another array	`double[] b = new double[N];`
	`for (int i = 0; i < N; i++)`
	`  b[i] = a[i];`

reverse the elements within an array	`for (int i = 0; i < N/2; i++) {`
	`  double temp = a[i];`
	`  a[i] = a[N - 1 - i];`
	`  a[N - 1 - i] = temp;`
	`}`

بر زدن یک دسته ورق

تنظیم مقادیر آرایه در زمان کامپایل

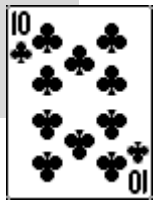
- مثال. چاپ یک کارت تصادفی.

```
String[] rank = {
   "2", "3", "4", "5", "6", "7", "8", "9",
   "10", "Jack", "Queen", "King", "Ace"
};

String[] suit = {
   "Clubs", "Diamonds", "Hearts", "Spades"
};

int i = (int) (Math.random() * 13); // between 0 and 12
int j = (int) (Math.random() * 4);  // between 0 and 3

System.out.println(rank[i] + " of " + suit[j]);
```

“10 of clubs” ⟶ 

تنظیم مقادیر آرایه در زمان کامپایل

- مثال. ایجاد یک دسته ورق و چاپ آنها در خروجی.

```
String[] deck = new String[52];
for (int i = 0; i < 13; i++)
   for (int j = 0; j < 4; j++)
      deck[4*i + j] = rank[i] + " of " + suit[j];

for (int i = 0; i < 52; i++)
   System.out.println(deck[i]);
```

- پرسش. کارت‌ها به چه ترتیبی چاپ می‌شوند؟

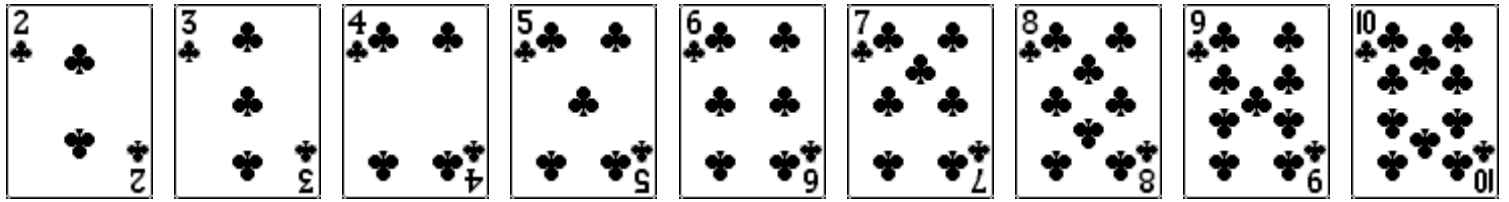
A.
```
2 of clubs
2 of diamonds
2 of hearts
2 of spades
3 of clubs
...
```

B.
```
2 of clubs
3 of clubs
4 of clubs
5 of clubs
6 of clubs
...
```

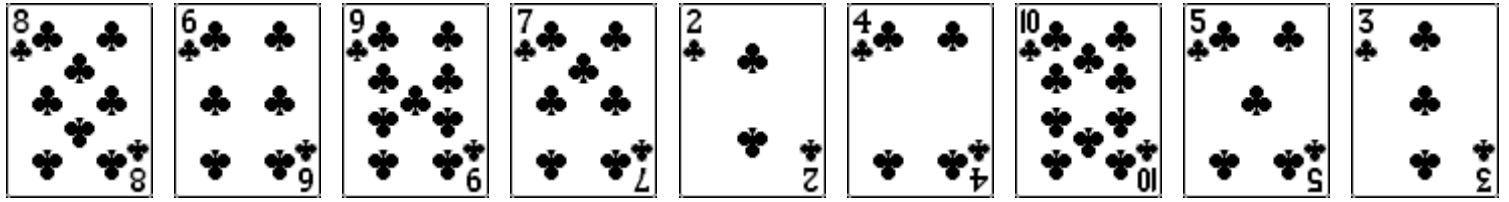
بر زدن

- بر زدن. بازآرایی عناصر یک آرایه به گونه‌ای که نتیجه یک جایگشت تصادفی با توزیع یکنواخت باشد.

پیش از بر زدن

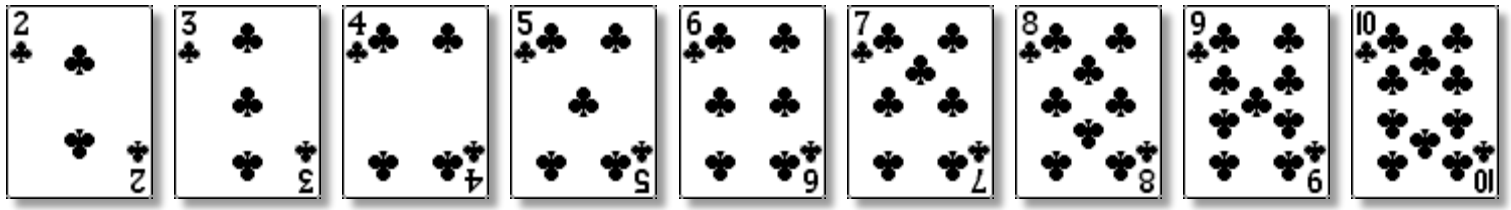


پس از بر زدن



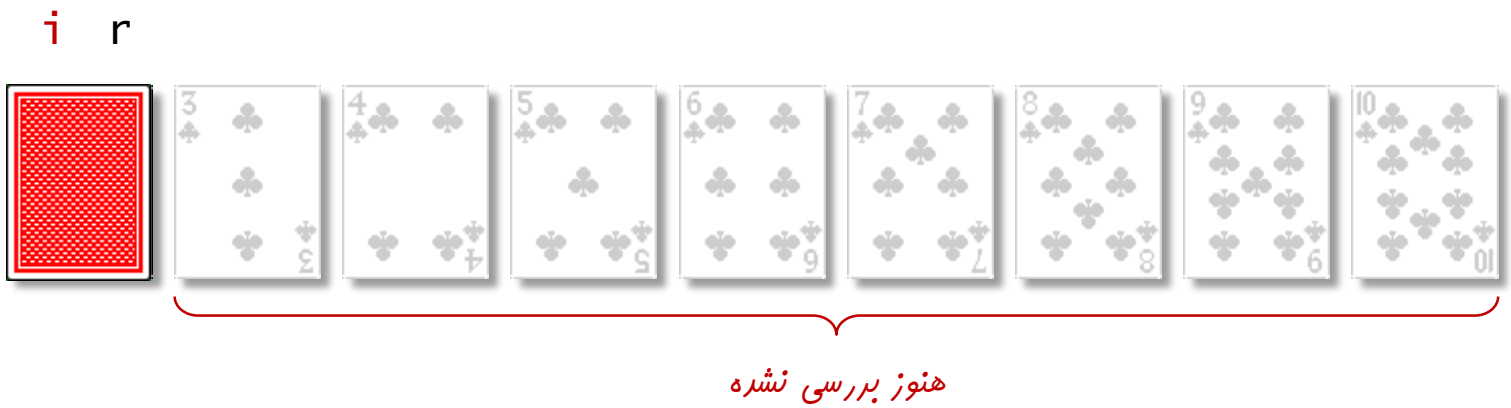
بر زدن

- در تکرار `i` یک عدد تصادفی مانند `r` بین صفر و `i` با احتمال مساوی تولید کن.
- کارت‌های `a[i]` و `a[r]` را جابجا کن.



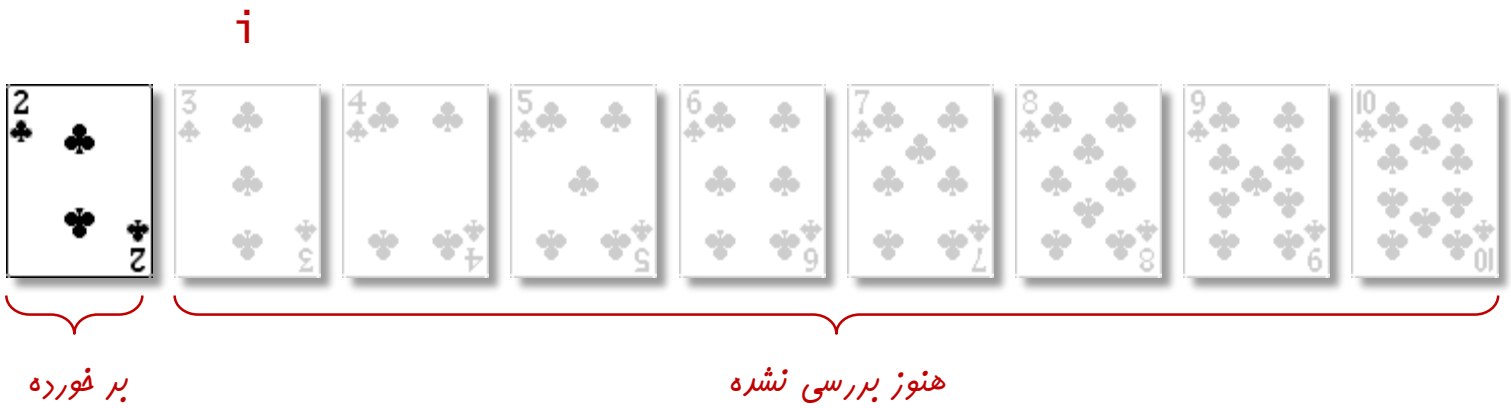
بر زدن

- در تکرار `i` یک عدد تصادفی مانند `r` بین صفر و `i` با احتمال مساوی تولید کن.
- کارت‌های `a[i]` و `a[r]` را جابجا کن.



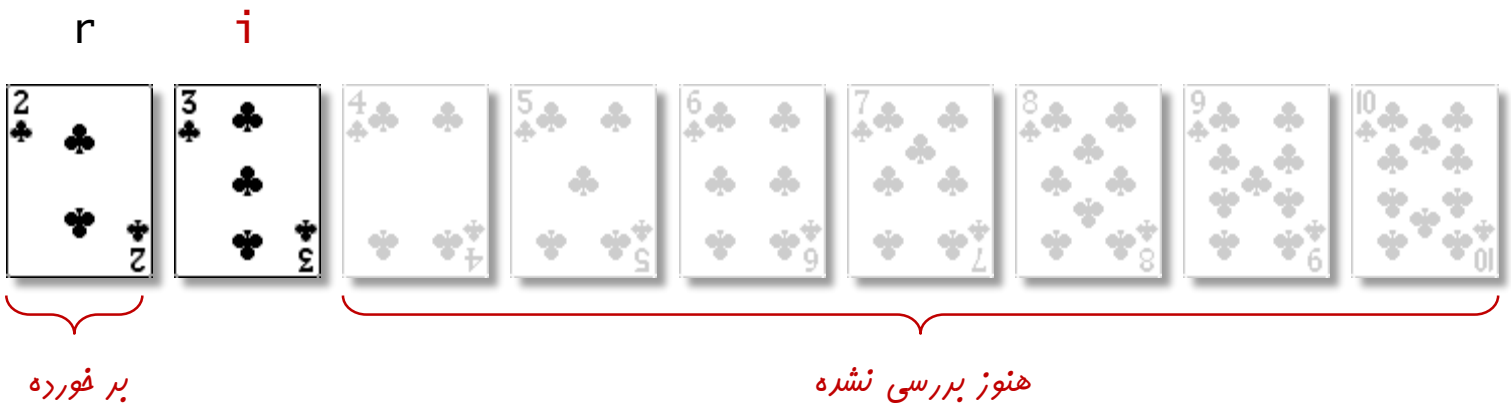
بر زدن

- در تکرار i یک عدد تصادفی مانند r بین صفر و i با احتمال مساوی تولید کن.
- کارت‌های a[i] و a[r] را جابجا کن.



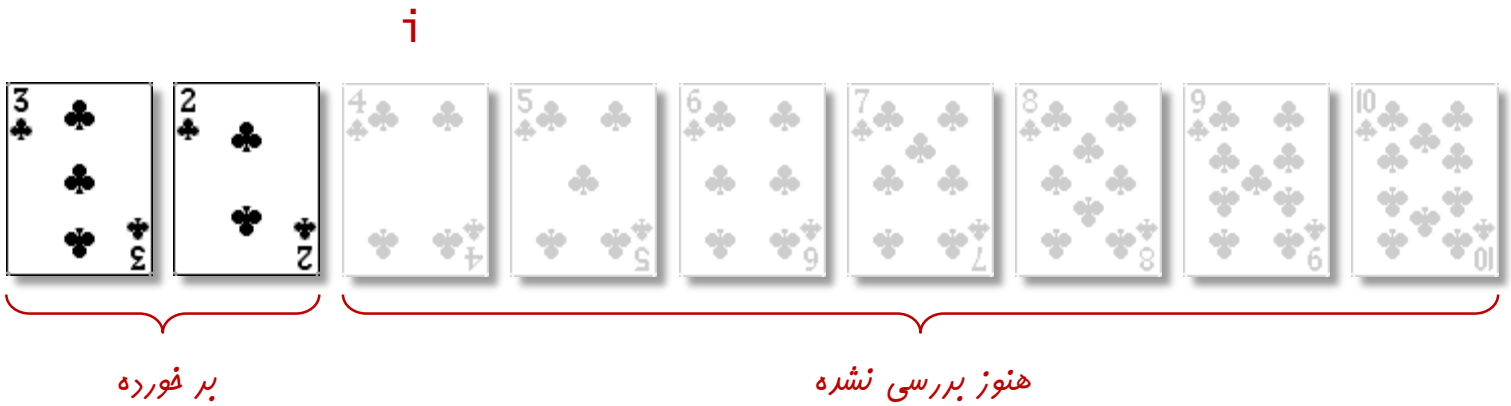
بر زدن

- در تکرار i یک عدد تصادفی مانند r بین صفر و i با احتمال مساوی تولید کن.
- کارت‌های a[i] و a[r] را جابجا کن.



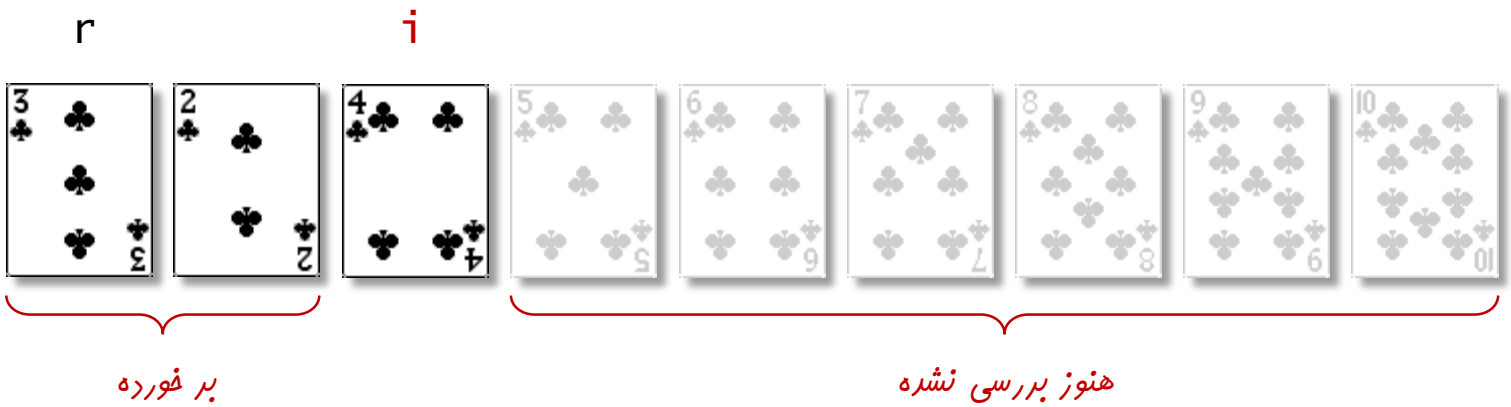
بر زدن

- در تکرار `i` یک عدد تصادفی مانند `r` بین صفر و `i` با احتمال مساوی تولید کن.
- کارت‌های `a[i]` و `a[r]` را جابجا کن.



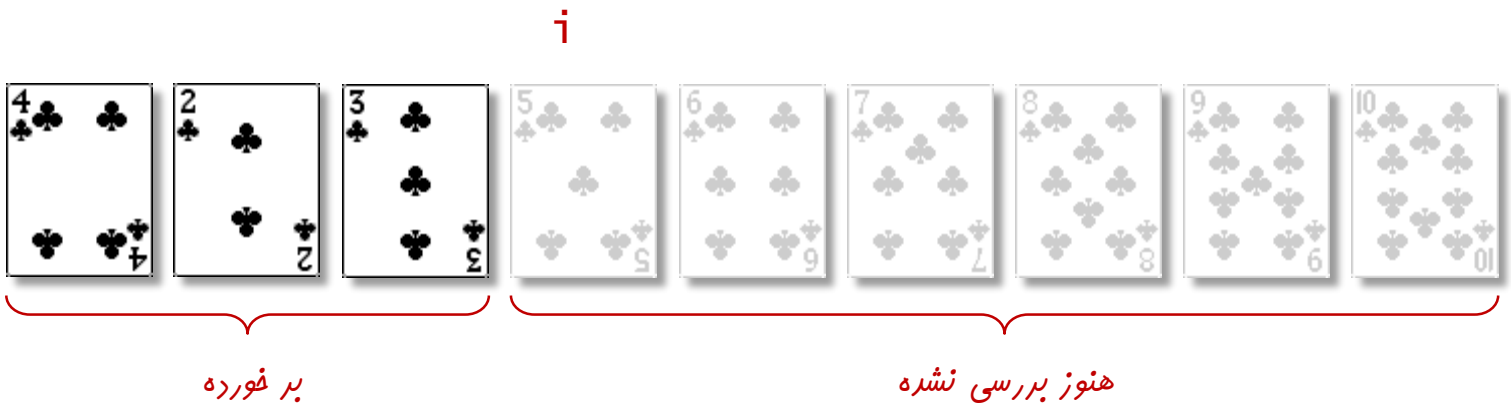
بر زدن

- در تکرار `i` یک عدد تصادفی مانند `r` بین صفر و `i` با احتمال مساوی تولید کن.
- کارت‌های `a[i]` و `a[r]` را جابجا کن.



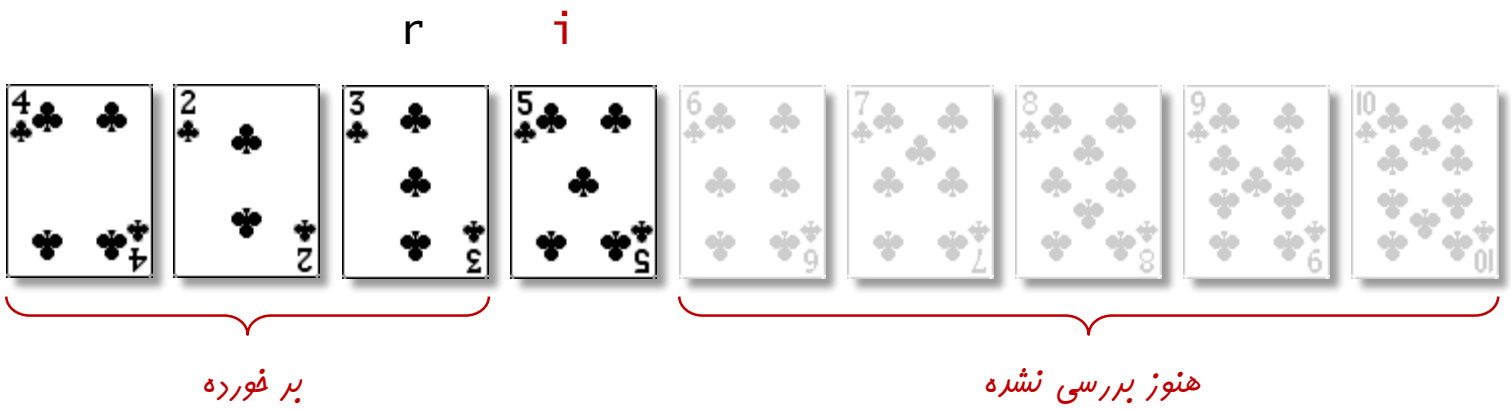
بر زدن

- در تکرار `i` یک عدد تصادفی مانند `r` بین صفر و `i` با احتمال مساوی تولید کن.
- کارت‌های `a[i]` و `a[r]` را جابجا کن.



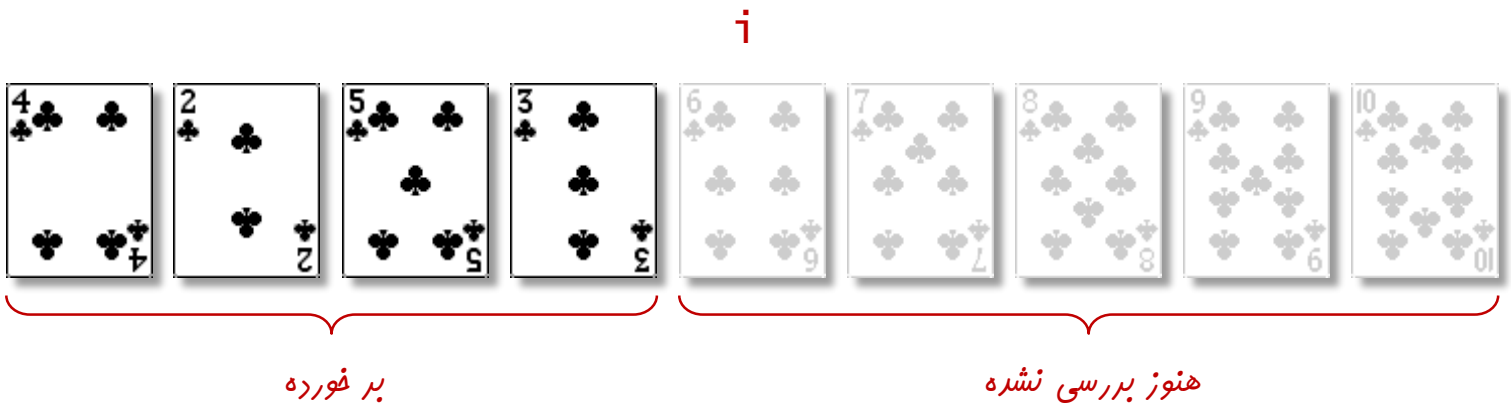
بر زدن

- در تکرار i یک عدد تصادفی مانند r بین صفر و i با احتمال مساوی تولید کن.
- کارت‌های a[i] و a[r] را جابجا کن.



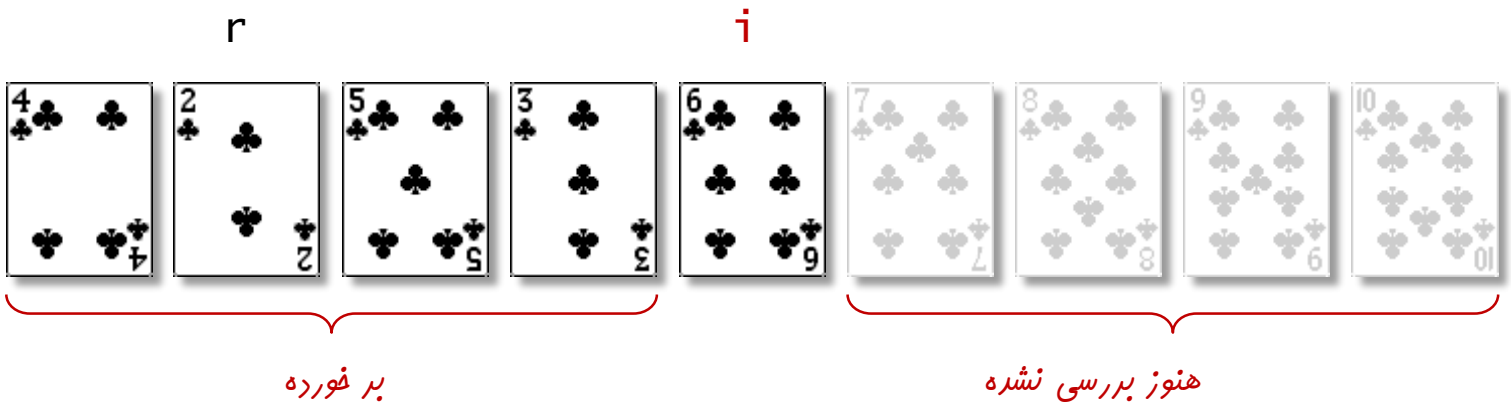
بر زدن

- در تکرار i یک عدد تصادفی مانند r بین صفر و i با احتمال مساوی تولید کن.
- کارت‌های a[i] و a[r] را جابجا کن.



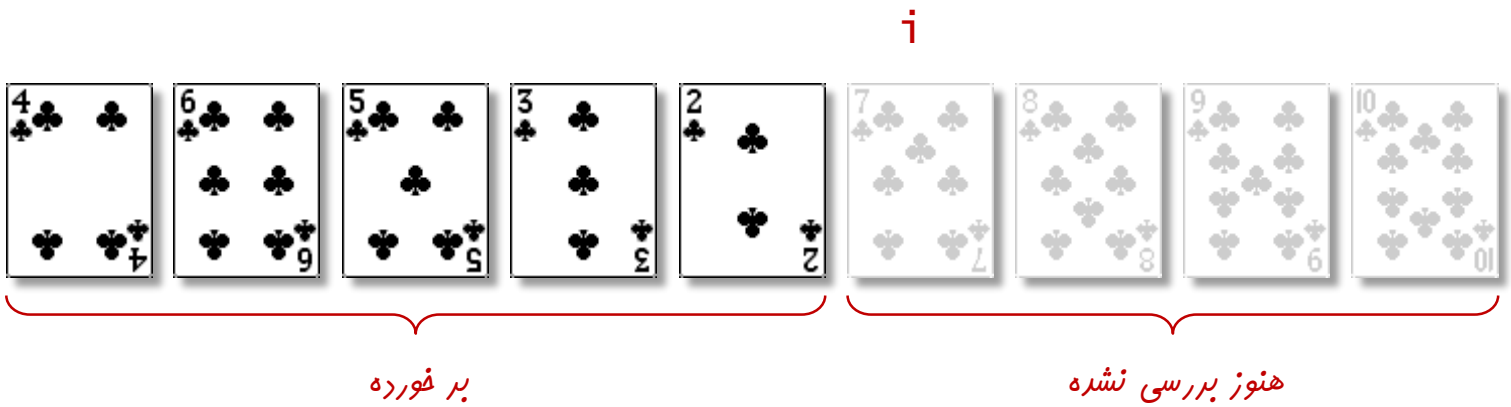
بر زدن

- در تکرار i یک عدد تصادفی مانند r بین صفر و i با احتمال مساوی تولید کن.
- کارت‌های a[i] و a[r] را جابجا کن.



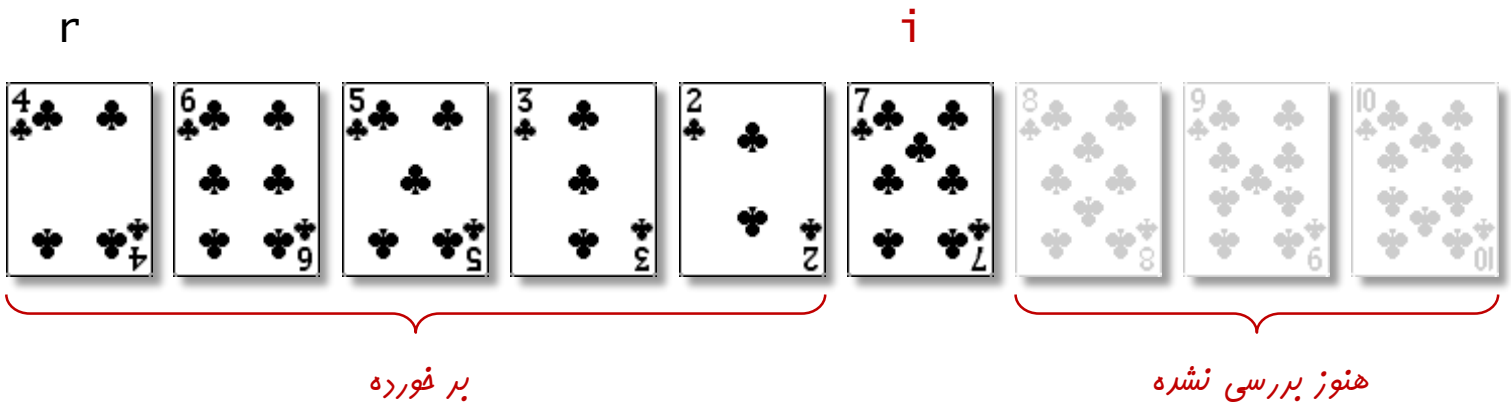
بر زدن

- در تکرار `i` یک عدد تصادفی مانند `r` بین صفر و `i` با احتمال مساوی تولید کن.
- کارت‌های `a[i]` و `a[r]` را جابجا کن.



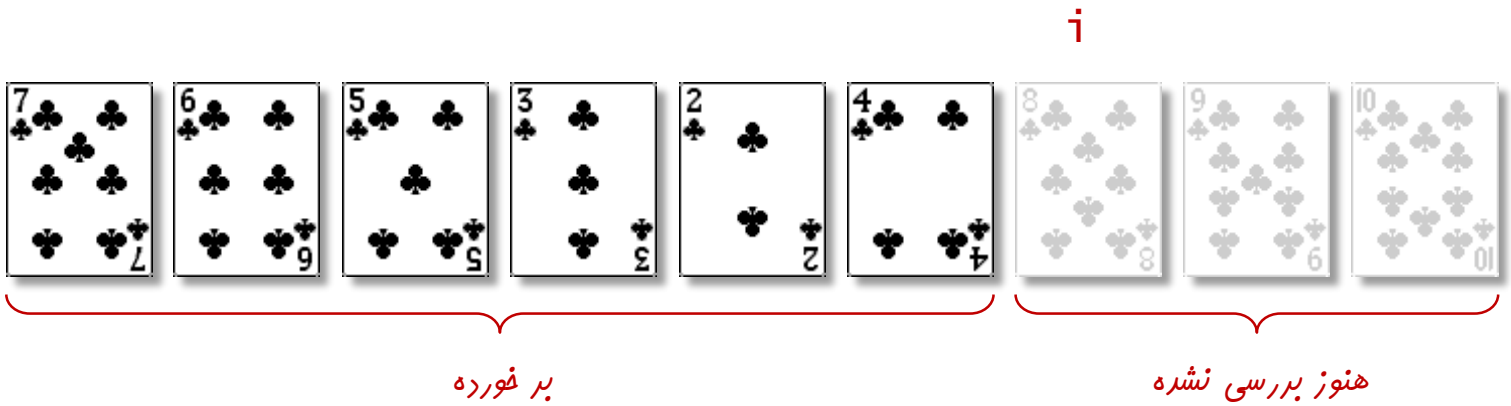
بر زدن

- در تکرار `i` یک عدد تصادفی مانند `r` بین صفر و `i` با احتمال مساوی تولید کن.
- کارت‌های `a[i]` و `a[r]` را جابجا کن.



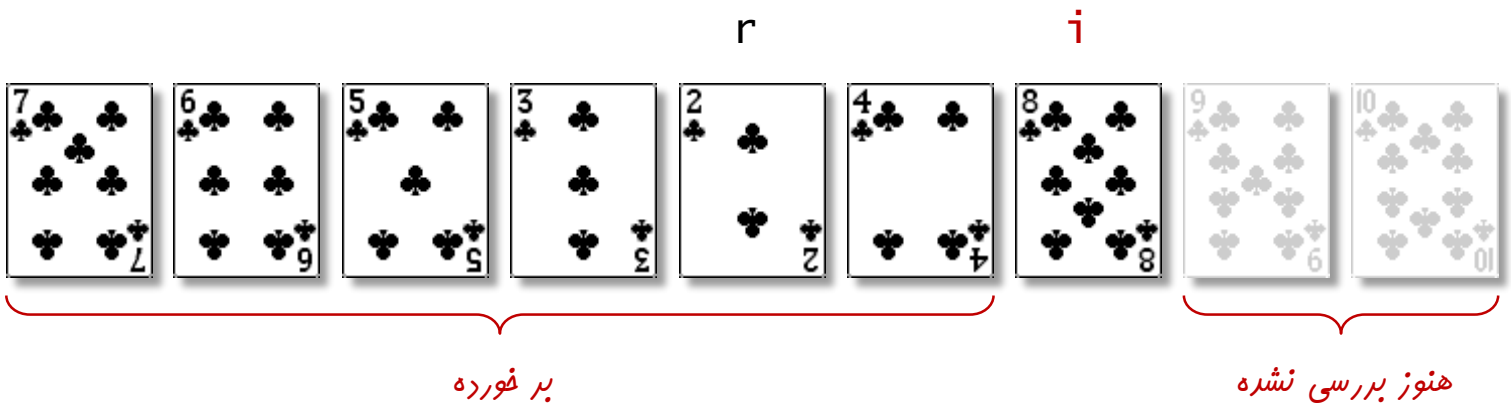
بر زدن

- در تکرار `i` یک عدد تصادفی مانند `r` بین صفر و `i` با احتمال مساوی تولید کن.
- کارت‌های `a[i]` و `a[r]` را جابجا کن.



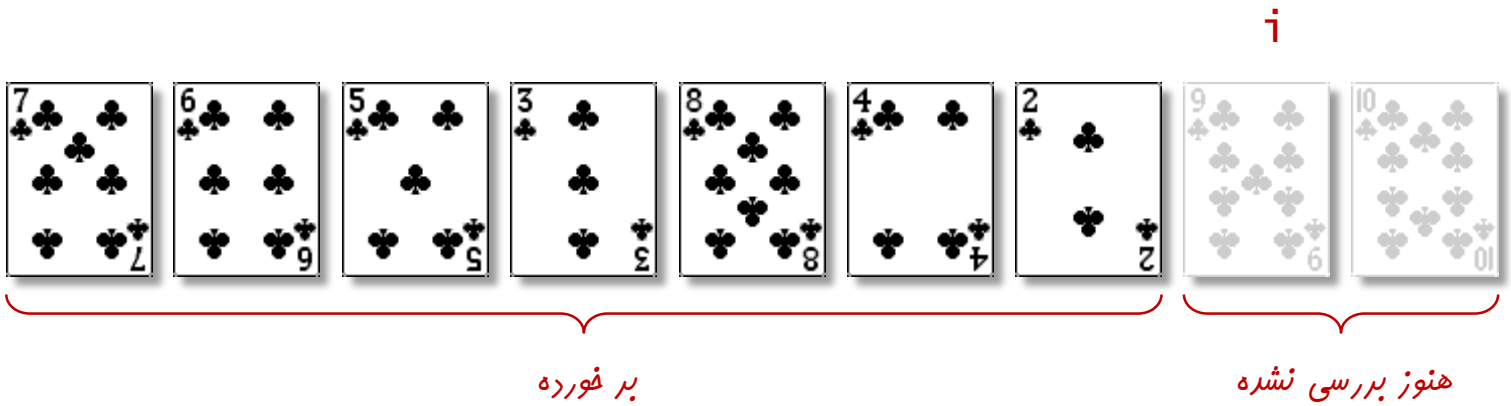
بر زدن

- در تکرار `i` یک عدد تصادفی مانند `r` بین صفر و `i` با احتمال مساوی تولید کن.
- کارت‌های `a[i]` و `a[r]` را جابجا کن.



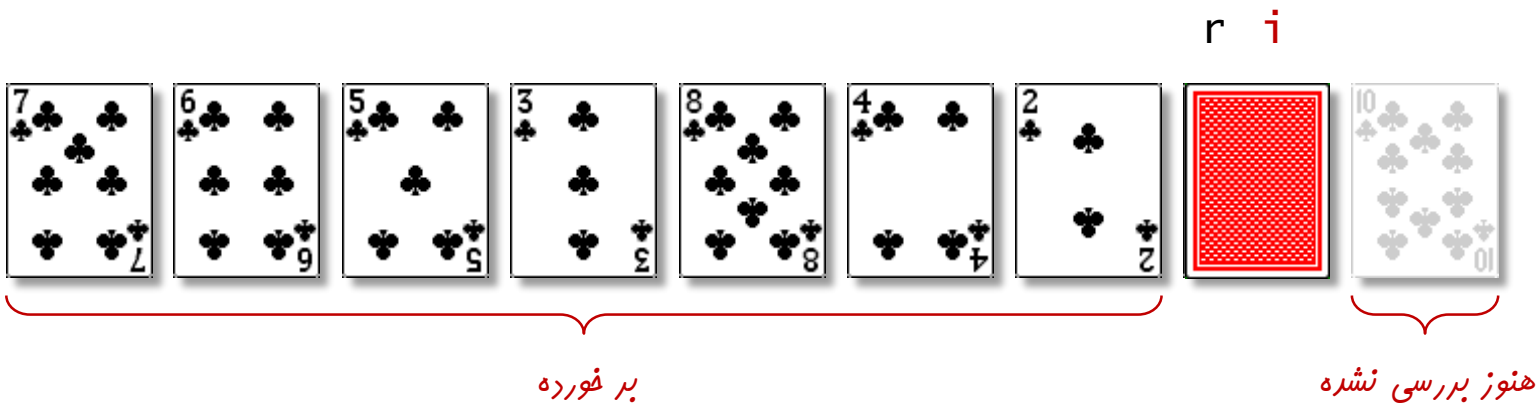
بر زدن

- در تکرار `i` یک عدد تصادفی مانند `r` بین صفر و `i` با احتمال مساوی تولید کن.
- کارت‌های `a[i]` و `a[r]` را جابجا کن.



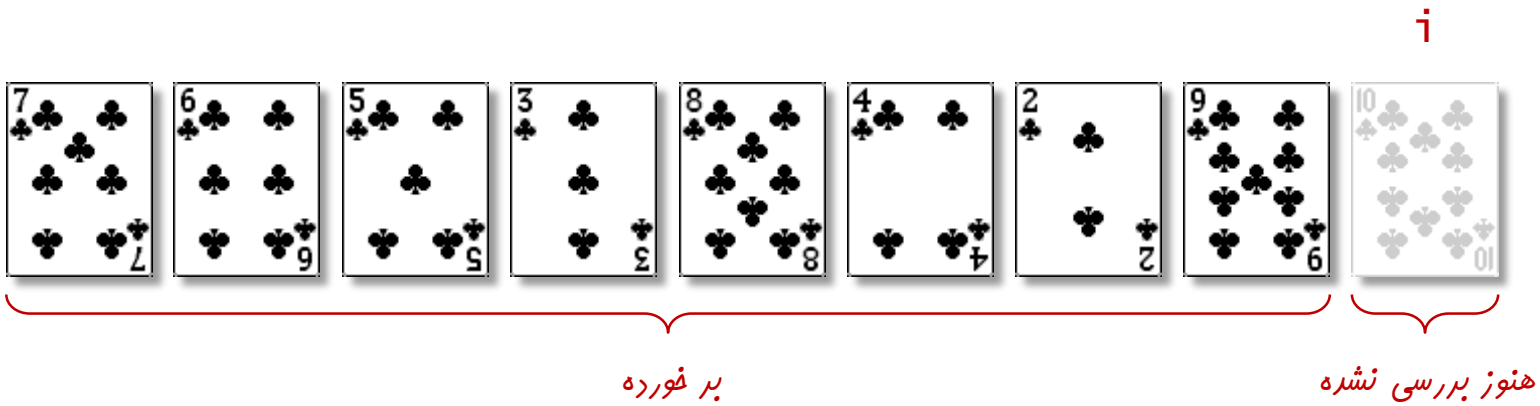
بر زدن

- در تکرار i یک عدد تصادفی مانند r بین صفر و i با احتمال مساوی تولید کن.
- کارت‌های a[i] و a[r] را جابجا کن.



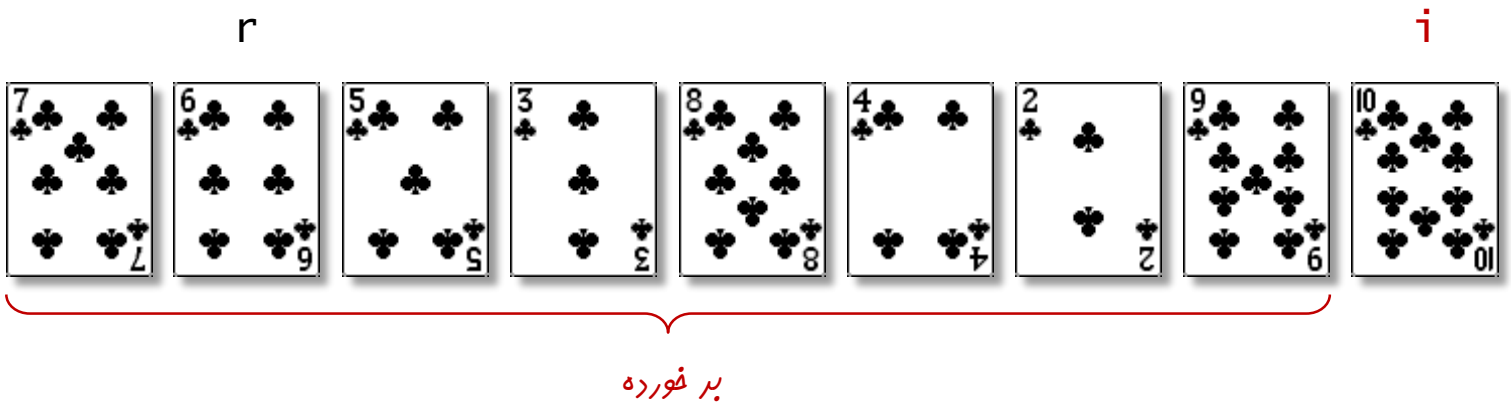
بر زدن

- در تکرار i یک عدد تصادفی مانند r بین صفر و i با احتمال مساوی تولید کن.
- کارت‌های a[i] و a[r] را جابجا کن.



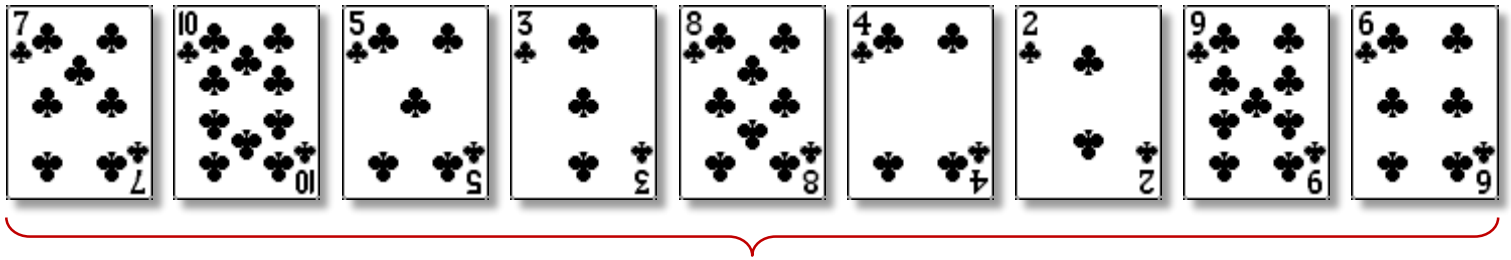
بر زدن

- در تکرار i یک عدد تصادفی مانند r بین صفر و i با احتمال مساوی تولید کن.
- کارت‌های a[i] و a[r] را جابجا کن.



بر زدن

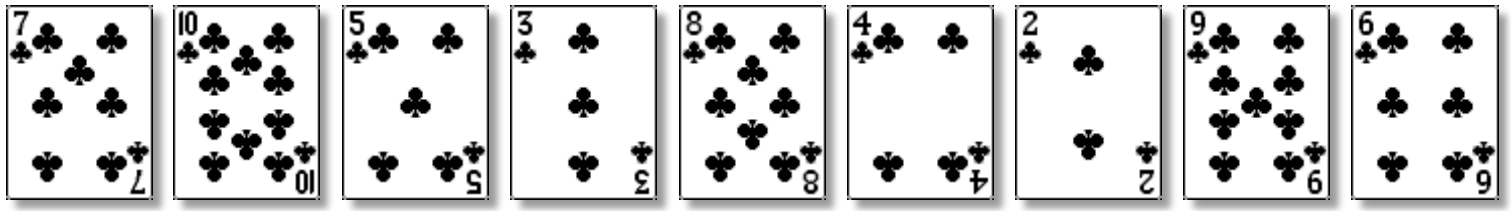
- در تکرار `i` یک عدد تصادفی مانند `r` بین صفر و `i` با احتمال مساوی تولید کن.
- کارت‌های `a[i]` و `a[r]` را جابجا کن.



بر خورده

بر زدن

- در تکرار `i` یک عدد تصادفی مانند `r` بین صفر و `i` با احتمال مساوی تولید کن.
- کارت‌های `a[i]` و `a[r]` را جابجا کن.



- گزاره. [فیشر، ۱۹۳۸] الگوریتم بر زدن کنوث در زمان خطی یک جایگشت تصادفی یکنواخت از آرایه ورودی تولید می‌کند.

بر زدن

- هدف. با داشتن یک آرایه، عناصر آن را به یک ترتیب تصادفی بازآرایی کن.

- الگوریتم بر زدن.
  - در تکرار `i`، یک کارت تصادفی از بین کارت‌های `deck[0]` تا `deck[i]` انتخاب کن، به طوری که احتمال انتخاب هر یک از این کارت‌ها برابر باشد.
  - کارت تصادفی انتخاب شده را با `deck[i]` جابجا کن.

```
int N = deck.length;
for (int i = 0; i < N; i++) {
   int r = StdRandom.uniform(i + 1);
   String t = deck[r];
   deck[r] = deck[i];
   deck[i] = t;
}
```

بر زدن یک دسته ورق


```
public class Deck {
   public static void main(String[] args) {
      String[] suit = { "Clubs", "Diamonds", "Hearts", "Spades" };
      String[] rank = { "2", "3", "4", "5", "6", "7", "8", "9",
                        "10", "Jack", "Queen", "King", "Ace" };
      int SUITS = suit.length;
      int RANKS = rank.length;
      int N = SUITS * RANKS;

      String[] deck = new String[N];
      for (int i = 0; i < RANKS; i++)
         for (int j = 0; j < SUITS; j++)
            deck[SUITS*i + j] = rank[i] + " of " + suit[j];

      for (int i = 0; i < N; i++) {
         int r = StdRandom.uniform(i + 1);
         String t = deck[r];
         deck[r] = deck[i];
         deck[i] = t;
      }

     for (int i = 0; i < N; i++)
         System.out.println(deck[i]);
}
```

ایجاد دسته ورق

بر زدن دسته ورق

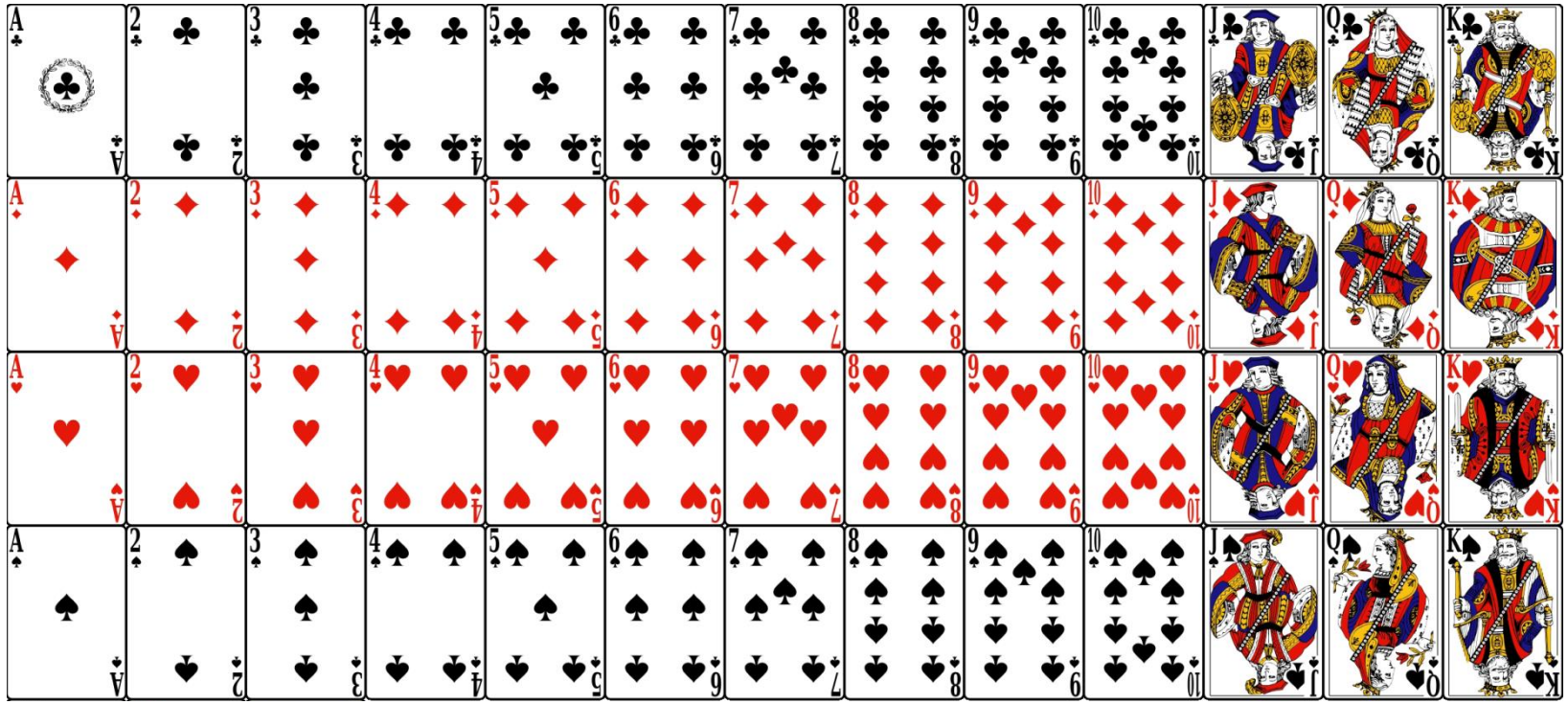
چاپ دسته ورق

بر زدن یک دسته ورق


```
% java Deck
10 of Diamonds
King of Spades
2 of Spades
3 of Clubs
4 of Spades
Queen of Clubs
2 of Hearts
7 of Diamonds
6 of Spades
Queen of Spades
3 of Spades
Jack of Diamonds
6 of Diamonds
8 of Spades
9 of Diamonds
...
10 of Spades
```

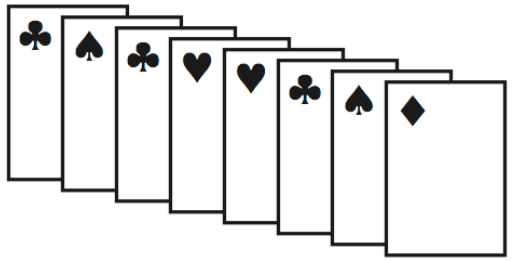
```
% java Deck
5 of Clubs
Jack of Hearts
9 of Spades
10 of Spades
9 of Clubs
7 of Spades
6 of Diamonds
7 of Hearts
7 of Clubs
4 of Spades
Queen of Diamonds
10 of Hearts
5 of Diamonds
Jack of Clubs
Ace of Hearts
...
5 of Spades
```

مسئله جمع‌کننده کالابرگ‌ها



مسئله جمع‌کننده کالابرگ‌ها

- مسئله جمع‌کننده کالابرگ‌ها. با داشتن N نوع کارت مختلف، برای این که از هر کارت (حداقل) یکی داشته باشیم چند کارت باید انتخاب شود؟



فرض می‌کنیم در هر انتخاب، احتمال انتخاب کارت‌ها با یکدیگر برابر است.

- الگوریتم شبیه‌سازی. به صورت تکراری هر بار یک عدد صحیح بین 0 و N - 1 انتخاب کن، تا زمانی که از هر کارت حداقل یکی داشته باشیم.

- پرسش. چگونه می‌توانیم بررسی کنیم که از هر نوع کارت حداقل یکی داریم؟
- پاسخ. با استفاده از یک آرایه بولی به گونه‌ای که found[i] تنها وقتی درست است که از کارت نوع i حداقل یکی داشته باشیم.

مسئله جمع‌کننده کالابرگ‌ها


```
public class CouponCollector {
   public static void main(String[] args) {
      int N = Integer.parseInt(args[0]);
      int cardcnt = 0;     // number of cards collected
      int valcnt = 0;      // number of distinct cards

      // do simulation
      boolean[] found = new boolean[N];
      while (valcnt < N) {
         int val = (int) (Math.random() * N);
         cardcnt++;
         if (!found[val]) {
            valcnt++;
            found[val] = true;
         }
      }

      // all N distinct cards found
      System.out.println(cardcnt);
   }
}
```

نوع کارت بعدی
(بین ۰ و N - 1)

مسئله جمع‌کننده کالابرگ‌ها: اشکال زدایی

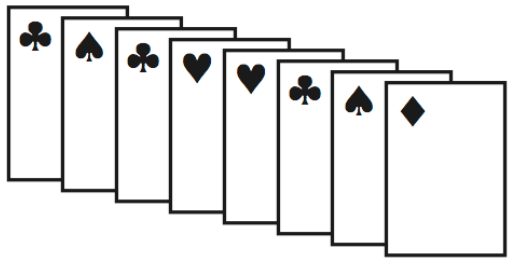
- اشکال زدایی. افزودن کد به برنامه به منظور چاپ محتویات همه متغیرها.

val	found						valcnt	cardcnt
	0	1	2	3	4	5		
	F	F	F	F	F	F	0	0
2	F	F	T	F	F	F	1	1
0	T	F	T	F	F	F	2	2
4	T	F	T	F	T	F	3	3
0	T	F	T	F	T	F	3	4
1	T	T	T	F	T	F	4	5
2	T	T	T	F	T	F	4	6
5	T	T	T	F	T	T	5	7
0	T	T	T	F	T	T	5	8
1	T	T	T	F	T	T	5	9
3	T	T	T	T	T	T	6	10

- چالش. در اشکال زدایی با وجود آرایه، به ردیابی تعداد زیادی متغیر نیاز است.

مسئله جمع‌کننده کالابرگ‌ها: برخورد ریاضی

- مسئله جمع‌کننده کالابرگ‌ها. با داشتن N نوع کارت مختلف، برای این که از هر کارت (حداقل) یکی داشته باشیم چند کارت باید انتخاب شود؟

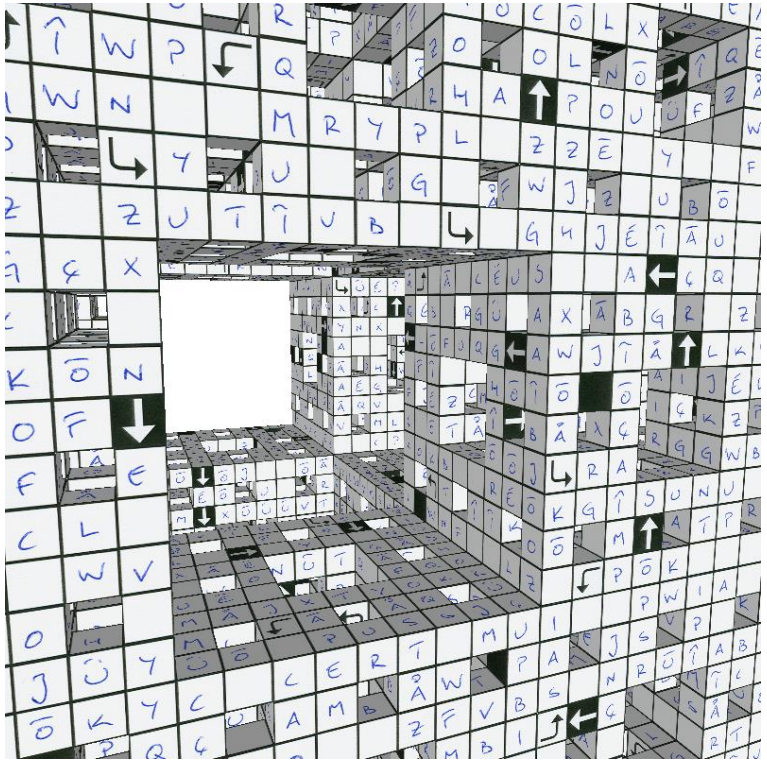


- حقیقت. تقریباً $N (1 + 1/2 + 1/3 + \ldots + 1/N) \sim N \ln N$

تحت شرایط ایده‌آل

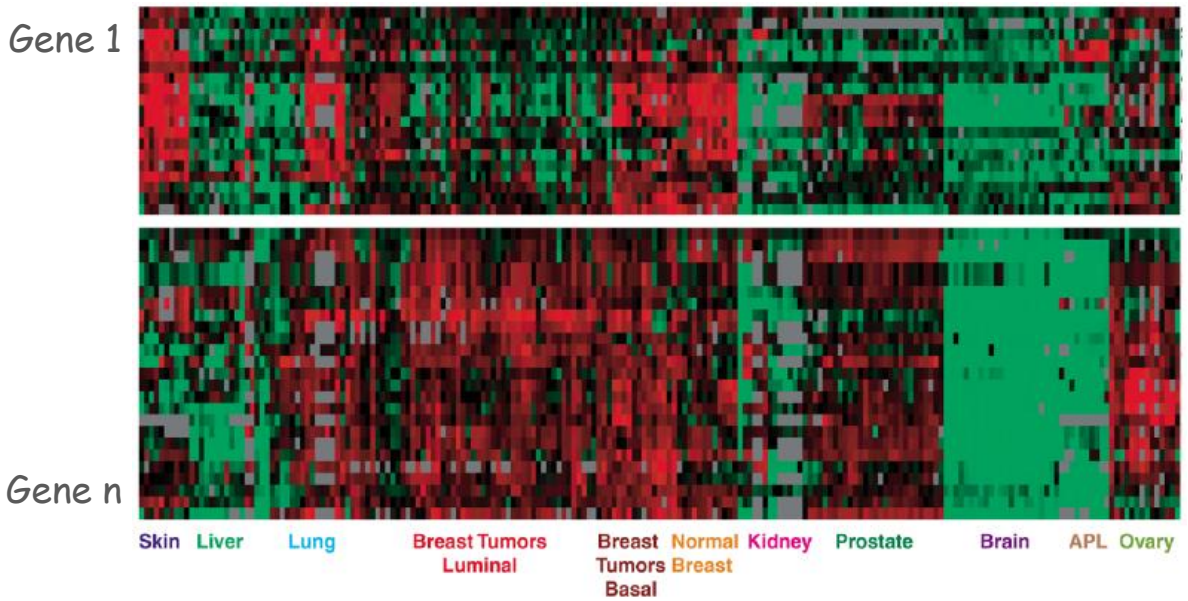
- مثال. برای ۳۰ تیم بیسبال، تقریباً ۱۲۰ سال زمان نیاز داریم تا هر تیم حداقل یک بار قهرمان شود.

آرایه‌های چند بعدی



آرایه‌های دو بعدی

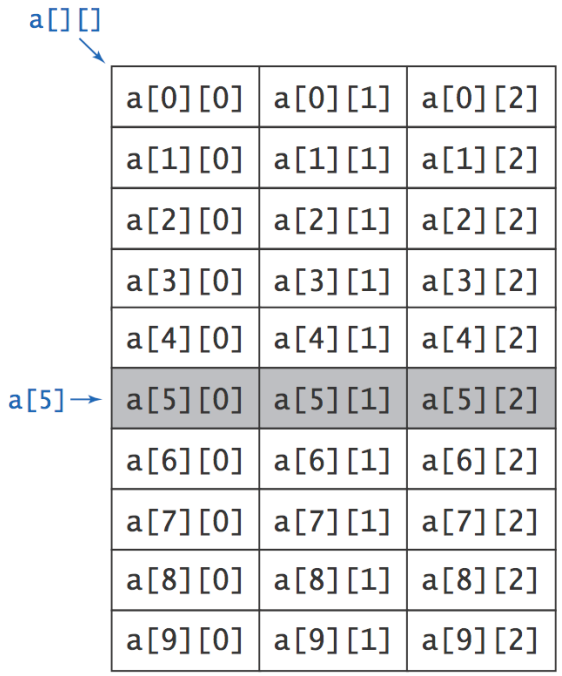
- آرایه‌های دو بعدی.
  - یک جدول از داده‌ها به ازای هر آزمایش و نتایج آن.
  - یک جدول از نمرات به ازای هر دانشجو و تمرینات.
  - یک جدول از مقادیر خاکستری به ازای هر پیکسل در یک تصویر دو بعدی.



آرایه‌های دو بعدی در جاوا

- دستیابی به عناصر. `a[i][j]` = عنصر واقع در سطر `i` و ستون `j`.
- اندیس گذاری مبتنی بر صفر. اندیس سطرها و ستون‌ها از صفر شروع می‌شود.

```
int M = 10;
int N = 3;
double[][] a = new double[M][N];
for (int i = 0; i < M; i++) {
   for (int j = 0; j < N; j++) {
      a[i][j] = 0.0;
   }
}
```

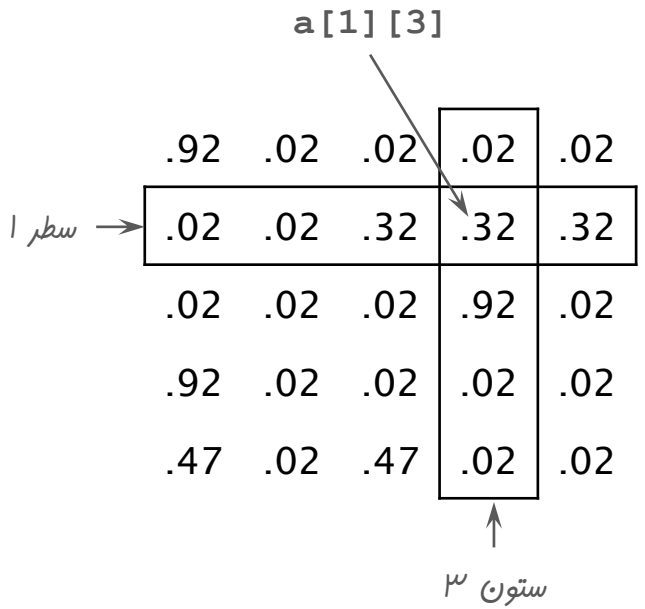


A 10-by-3 array

مقداردهی آرایه دو بعدی در زمان کامپایل

- مقداردهی آرایه دو بعدی با لیست کردن مقادیر.

```
double[][] p = {
    { .02, .92, .02, .02, .02 },
    { .02, .02, .32, .32, .32 },
    { .02, .02, .02, .92, .02 },
    { .92, .02, .02, .02, .02 },
    { .47, .02, .47, .02, .02 },
};
```

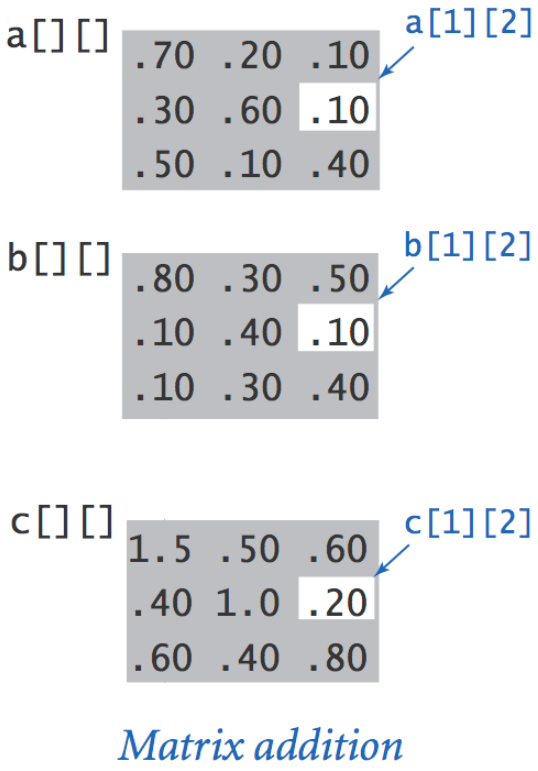


جمع ماتریسی

- جمع ماتریسی. با داشتن دو ماتریس a و b با ابعاد N در N، ماتریس c یک ماتریس N در N است به گونه‌ای که:

```
c[i][j] = a[i][j] + b[i][j]
```

```
double[][] c = new double[N][N];
for (int i = 0; i < N; i++)
   for (int j = 0; j < N; j++)
      c[i][j] = a[i][j] + b[i][j];
```



Matrix addition

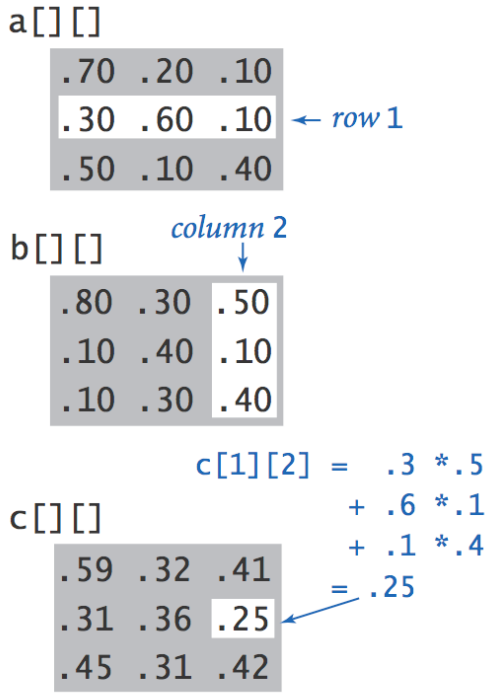
ضرب ماتریسی

- ضرب ماتریسی. با داشتن دو ماتریس a و b با ابعاد N در N، ماتریس c یک ماتریس N در N است به گونه‌ای که در آن c[i][j] برابر است با ضرب داخلی سطر i از ماتریس a[][] در ستون j از ماتریس b[][].

مقدار تمام عناصر برابر با صفر است

```
double[][] c = new double[N][N];
for (int i = 0; i < N; i++)
   for (int j = 0; j < N; j++)
      for (int k = 0; k < N; k++)
         c[i][j] += a[i][k] * b[k][j];
```

ضرب داخلی
سطر i از ماتریس a[][] در
ستون j از ماتریس b[][]



Matrix multiplication

تحلیل ضرب ماتریسی

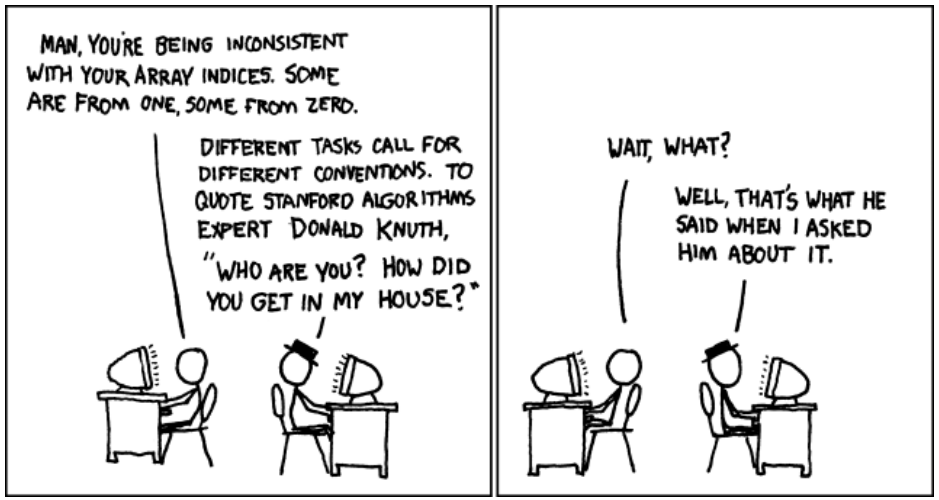
- پرسش. برای ضرب دو ماتریس N در N به چند عمل ضرب اسکالر نیاز است؟

A. N B. N^2 C. N^3 D. N^4

```
double[][] c = new double[N][N];
for (int i = 0; i < N; i++)
   for (int j = 0; j < N; j++)
      for (int k = 0; k < N; k++)
         c[i][j] += a[i][k] * b[k][j];
```

خلاصه

- آرایه‌ها.
  - یک روش سازمان یافته به منظور ذخیره‌سازی حجم انبوهی از داده‌ها.
  - استفاده از آرایه تقریباً به اندازه استفاده از انواع اولیه ساده است.
  - با داشتن اندیس یک عنصر، می‌توانیم مستقیماً به آن عنصر دستیابی داشته باشیم.



- گام بعدی.
  - خواندن مقدار زیادی داده از فایل و
  - ذخیره‌سازی آنها در یک آرایه.