

Перетасовка Кнута

- На итерации i выбрать r между 0 и i при нормальном распределении
- Поменять $a[i]$ и $a[r]$

```
public class StdRandom
{
    ...
    public static void shuffle(Object[] a)
    {
        int N = a.length;
        for (int i = 0; i < N; i++)
        {
            int r = StdRandom.uniform(i + 1);
            exch(a, i, r);
        }
    }
}
```

← between 0 and i

War story (online poker)

Texas hold'em poker. Software must shuffle electronic cards.



How We Learned to Cheat at Online Poker: A Study in Software Security

War story (online poker)

Shuffling algorithm in FAQ at www.planetpoker.com

```
for i := 1 to 52 do begin
  r := random(51) + 1;
  swap := card[r];
  card[r] := card[i];
  card[i] := swap;
end;
```

← between 1 and 51



- Bug 1.** Random number r never 52 \Rightarrow 52nd card can't end up in 52nd place.
- Bug 2.** Shuffle not uniform (should be between 1 and i).
- Bug 3.** `random()` uses 32-bit seed \Rightarrow 2^{32} possible shuffles.
- Bug 4.** Seed = milliseconds since midnight \Rightarrow 86.4 million shuffles.

War story (Microsoft)

Microsoft antitrust probe by EU. Microsoft agreed to provide a randomized ballot screen for users to select browser in Windows 7.

<http://www.browserchoice.eu>

Select your web browser(s)



A fast new browser from Google. Try it now!



Safari for Windows from Apple, the world's most innovative browser.



Your online security is Firefox's top priority. Firefox is free, and made to help you get the most out of the



The fastest browser on Earth. Secure, powerful and easy to use, with excellent privacy protection.



Designed to help you take control of your privacy and browse with confidence. Free from Microsoft.

↑
appeared last
50% of the time

Быстрая сортировка (Quicksort)

Два классических алгоритма сортировки

- Критические компоненты в мировой вычислительной инфраструктуре
 - Понимание научных основ этих алгоритмов даст нам возможность разрабатывать прикладные системы для решения реальных задач
 - Быстрая сортировка входит в десятку самых полезных алгоритмов, разработанных в 20-м веке


```
public static void quicksort(char[] items, int left, int right)
{
    int i, j;
    char x, y;

    i = left; j = right;
    x = items[(left + right) / 2];

    do
    {
        while ((items[i] < x) && (i < right)) i++;
        while ((x < items[j]) && (j > left)) j--;

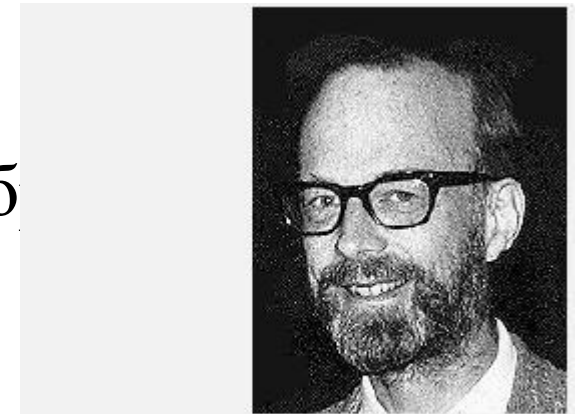
        if (i <= j)
        {
            y = items[i];
            items[i] = items[j];
            items[j] = y;
            i++; j--;
        }
    } while (i <= j);

    if (left < j) quicksort(items, left, j);
    if (i < right) quicksort(items, i, right);
}
```

Быстрая сортировка

- Основной план

- Перемешать элементы случайным образом
- Разбиение для элемента j
 - $a[j]$ оставить на месте
 - Нет элементов меньше чем $a[j]$ с правой стороны
 - Нет элементов больше чем $a[j]$ с левой стороны



Sir Charles Antony Richard Hoare
1980 Turing Award

input	Q	U	I	C	K	S	O	R	T	E	X	A	M	P	L	E
shuffle	K	R	A	T	E	L	E	P	U	I	M	Q	C	X	O	S
partition	E	C	A	I	E	K	L	P	U	T	M	Q	R	X	O	S
sort left	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
sort right	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
result	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X

Быстрая сортировка

- Повторять до тех пор, пока i и j не пересекутся
 - Проверять i -ые элементы до тех пор пока $a[i] < a[lo]$
 - Проверять j -ые элементы до тех пор пока $a[j] > a[lo]$
 - Поменять местами $a[i]$ и $a[j]$
- Видео 1


Быстрая сортировка: реализация на Java

```
public class Quick
{
    private static int partition(Comparable[] a, int lo, int hi)
    { /* see previous slide */ }

    public static void sort(Comparable[] a)
    {
        StdRandom.shuffle(a);
        sort(a, 0, a.length - 1);
    }

    private static void sort(Comparable[] a, int lo, int hi)
    {
        if (hi <= lo) return;
        int j = partition(a, lo, hi);
        sort(a, lo, j-1);
        sort(a, j+1, hi);
    }
}
```

shuffle needed for
performance guarantee
(stay tuned)



Быстрая сортировка

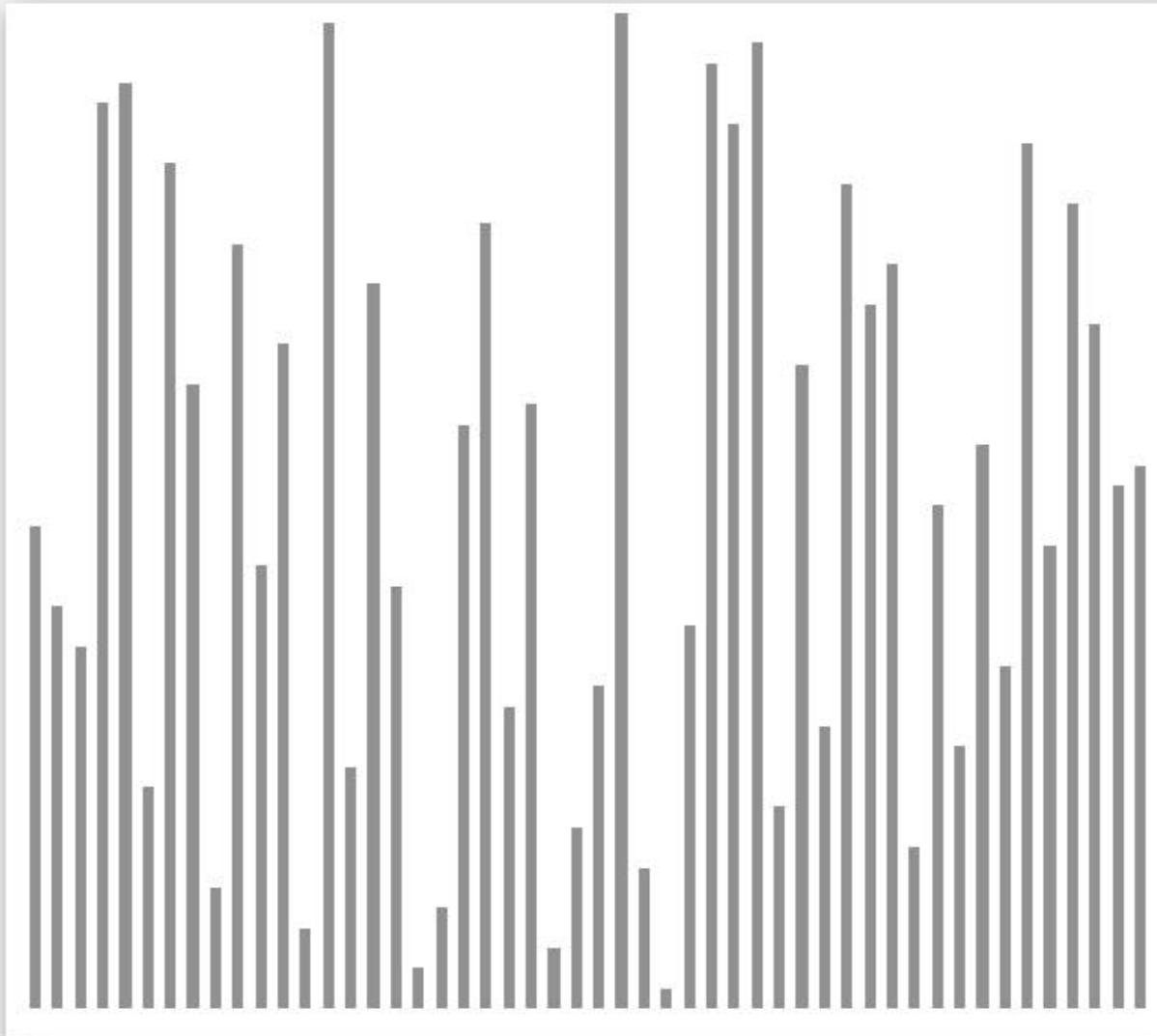
	lo	j	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
initial values				Q	U	I	C	K	S	O	R	T	E	X	A	M	P	L	E
random shuffle				K	R	A	T	E	L	E	P	U	I	M	Q	C	X	O	S
	0	5	15	E	C	A	I	E	K	L	P	U	T	M	Q	R	X	O	S
	0	3	4	E	C	A	E	I	K	L	P	U	T	M	Q	R	X	O	S
	0	2	2	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	0	0	1	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	1		1	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	4		4	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	6	6	15	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	7	9	15	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
	7	7	8	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
	8		8	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
	10	13	15	A	C	E	E	I	K	L	M	O	P	S	Q	R	T	U	X
	10	12	12	A	C	E	E	I	K	L	M	O	P	R	Q	S	T	U	X
	10	11	11	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
	10		10	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
	14	14	15	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
	15		15	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
result				A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X





no partition
for subarrays
of size 1

Quicksort trace (array contents after each partition)

Быстрая сортировка

50 random items



-  algorithm position
-  in order
-  current subarray
-  not in order

<http://www.sorting-algorithms.com/quick-sort>

Быстрая сортировка: реализация

- Не требует дополнительной памяти
- Выход из циклов. Обращайте особое внимание на условия выхода из циклов
- Ограничения. Проверка $j == l_0$ излишняя, но $i == h_i$ нет
- Рандомизация. Перетасовка нужна чтобы обеспечить гарантии производительности
- Равные ключи. Если присутствуют дубликаты, то можно использовать другой вариант алгоритма

Быстрая сортировка: лучший случай

- Лучший случай. Количество сравнений $\sim N \log_2 N$

			a[]														
lo	j	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
initial values			H	A	C	B	F	E	G	D	L	I	K	J	N	M	O
random shuffle			H	A	C	B	F	E	G	D	L	I	K	J	N	M	O
0	7	14	D	A	C	B	F	E	G	H	L	I	K	J	N	M	O
0	3	6	B	A	C	D	F	E	G	H	L	I	K	J	N	M	O
0	1	2	A	B	C	D	F	E	G	H	L	I	K	J	N	M	O
0		0	A	B	C	D	F	E	G	H	L	I	K	J	N	M	O
2		2	A	B	C	D	F	E	G	H	L	I	K	J	N	M	O
4	5	6	A	B	C	D	E	F	G	H	L	I	K	J	N	M	O
4		4	A	B	C	D	E	F	G	H	L	I	K	J	N	M	O
6		6	A	B	C	D	E	F	G	H	L	I	K	J	N	M	O
8	11	14	A	B	C	D	E	F	G	H	J	I	K	L	N	M	O
8	9	10	A	B	C	D	E	F	G	H	I	J	K	L	N	M	O
8		8	A	B	C	D	E	F	G	H	I	J	K	L	N	M	O
10		10	A	B	C	D	E	F	G	H	I	J	K	L	N	M	O
12	13	14	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
12		12	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
14		14	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
			A	B	C	D	E	F	G	H	I	J	K	L	M	N	O

Быстрая сортировка: худший случай

- Худший случай. Количество сравнений $\sim \frac{1}{2} N^2$

			a[]														
lo	j	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
initial values			A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
random shuffle			A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0	0	14	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	1	14	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
2	2	14	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
3	3	14	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
4	4	14	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	5	14	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
6	6	14	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
7	7	14	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
8	8	14	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
9	9	14	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
10	10	14	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
11	11	14	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
12	12	14	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
13	13	14	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
14		14	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
			A	B	C	D	E	F	G	H	I	J	K	L	M	N	O

Быстрая сортировка: свойства

- **Худший случай.** Квадратичное количество сравнений
 - $N + (N-1) + (N-2) + \dots + 1 \sim \frac{1}{2} N^2$
- **Средний случай.** Количество сравнений $\sim 1,39 N \log_2 N$
 - На 39% сравнений больше, чем у сортировки слиянием
 - Но, на практике, быстрее сортировки слиянием, потому что меньше перемещаются данные
- **Перетасовка**
 - Гарантирует отсутствия худшего случая
- **Предупреждение.** Часть реализаций быстрой