Quicksort

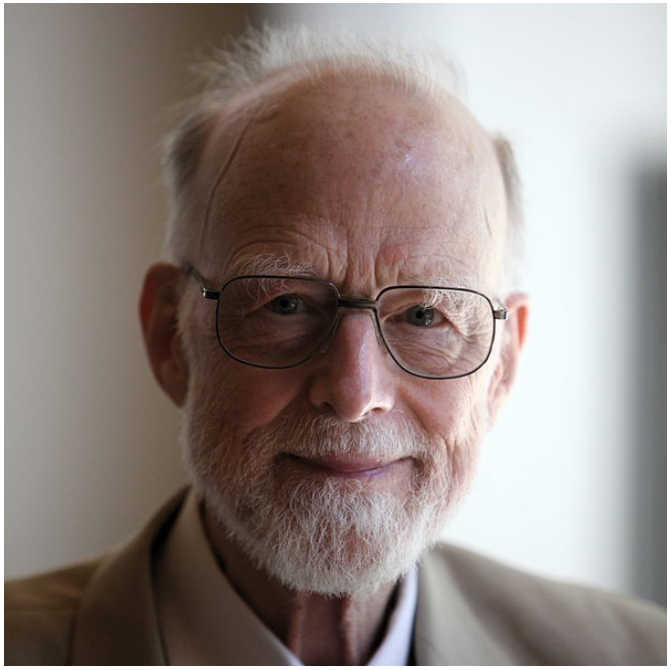
Tölvunarfræði 2, vor 2012

Hallgrímur H. Gunnarsson

Háskóli Íslands

2012-02-29

Tony Hoare



- Quicksort (1959)
- Rökstudd forritun
- Hoare logic
- CSP
- Hönnun forritunarmála

Hoare um hugbúnaðargerð

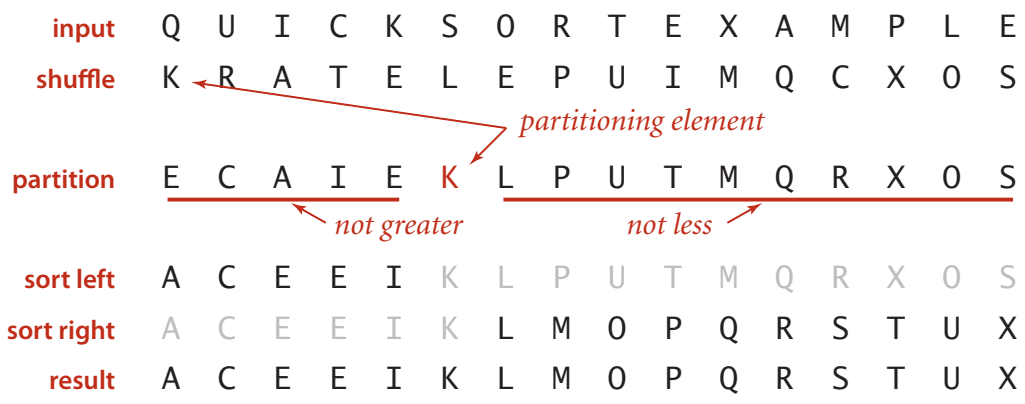
There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult.

– *Tony Hoare (from his 1980 Turing Award lecture)*

Quicksort

Hugmyndin:

- partition: Skiptum fylkinu þ.a. fyrir eitthvað k þá er
  1. f[k] er í réttu sæti
  2. ekkert stærra stak er fyrir neðan f[k]
  3. ekkert minna stak er fyrir ofan f[k]
- sort: Endurkvæmt röðum sitthvorum hlutanum



Quicksort overview

Quicksort

```
// Notkun: qsort(f, lo, hi);
// Fyrir:  f[lo..hi] er löglegt svæði
// Eftir:  f[lo..hi] er í vaxandi röð
static void qsort(Comparable[] a, int lo, int hi)
{
    if (hi <= lo) return;
    int k = partition(a, lo, hi);
    qsort(a, lo, k-1);
    qsort(a, k+1, hi);
}
```

Skipting (e. partitioning)

```
// Notkun: k = partition(a, lo, hi);
// Fyrir:  a[lo..hi] er löglegt svæði
// Eftir:  Búið er að víxla gildunum í f[lo..hi] þannig
//         að f[lo..k-1] <= f[k] <= f[k+1..hi].
static int partition(String[] a, int lo, int hi)
{
    // ..
}
```

Það eru til mörg afbrigði af skiptingu:

- Velur notandi fallsins vendistakið eða velur fallið það sjálft?
- Hvernig er vendistakið valið?
- Hvernig er skiptingin forrituð?
- Hvernig er unnið með jöfn gildi?

Dæmi um útfærslu á skiptingu

```
// Notkun: k = partition(f,i,j,k);
// Fyrir:  f[i..j-1] er löglegt svæði í f,
//         i <= k < j
// Eftir:  Búið er að víxla gildum í svæðinu þ.a.
//         f[i..k-1] <= f[k] <= f[k+1..j-1]
static int partition(int[] f, int i, int j, int k) {

    int p = f[k];
    int n = i+1;

    swap(f, i, k);

    // [p| ??? ]
    //  i n     j
```

Dæmi um útfærslu á skiptingu

```
    for (int t = i+1; t != j; t++) {
        // [p| < p | >= p | ??? ]
        //  i       n      t     j
        if (f[t] < p) {
            swap(f, t, n);
            n++;
        }
    }
    // [p| < p | >= p ]
    //  i       n      j
    n--;
    swap(f, i, n);
    // [ < p |p| >=p ]
    //  i     n       j
    return n;
}
```

Dæmi um þrjár aðrar aðferðir

http://algs4.cs.princeton.edu/lectures/23DemoPartitioning.pdf

Quicksort animation

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

Quicksort Wikipedia animation

http://en.wikipedia.org/wiki/Quicksort

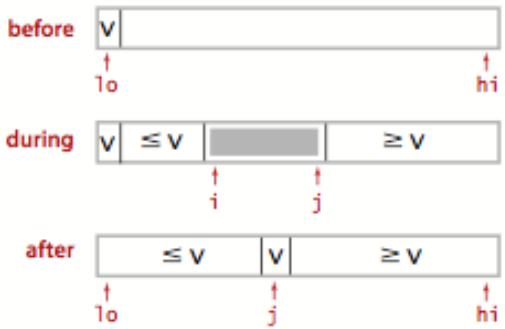
Sedgewick 2-way partitioning

```
// partition the subarray a[lo .. hi] by returning an index j
// so that a[lo .. j-1] <= a[j] <= a[j+1 .. hi]
private static int partition(Comparable[] a, int lo, int hi) {
    int i = lo;
    int j = hi + 1;
    Comparable v = a[lo];
    while (true) {
        while (less(a[++i], v))
            if (i == hi) break;

        while (less(v, a[--j]))
            if (j == lo) break;

        if (i >= j) break;
        swap(a, i, j);
    }
    swap(a, lo, j);
    return j;
}
```

Partioning trace

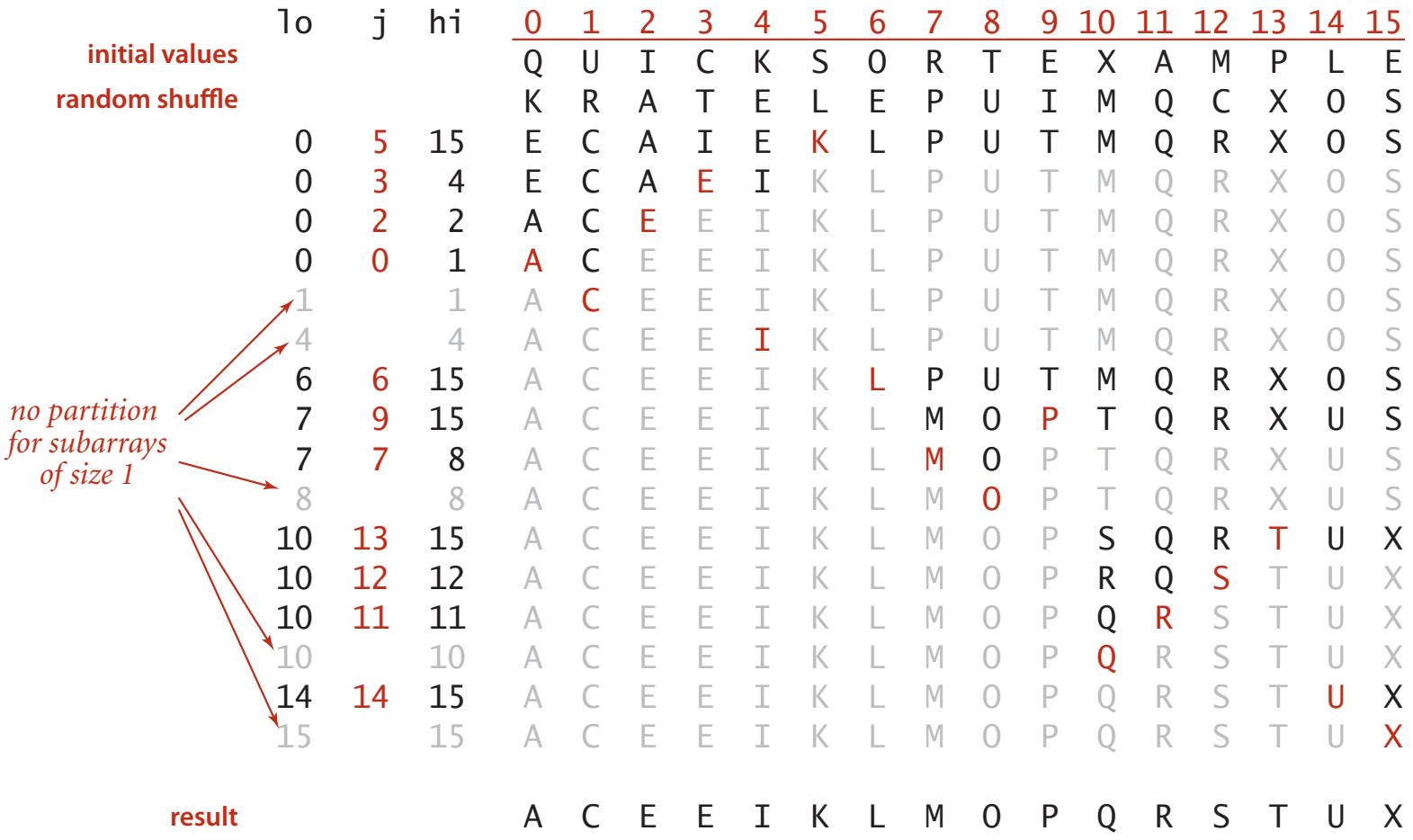


Quicksort partitioning overview

	i	j	a[i] 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
			v															
initial values	0	16	K	R	A	T	E	L	E	P	U	I	M	Q	C	X	O	S
scan left, scan right	1	12	K	R	A	T	E	L	E	P	U	I	M	Q	C	X	O	S
exchange	1	12	K	C	A	T	E	L	E	P	U	I	M	Q	R	X	O	S
scan left, scan right	3	9	K	C	A	T	E	L	E	P	U	I	M	Q	R	X	O	S
exchange	3	9	K	C	A	I	E	L	E	P	U	T	M	Q	R	X	O	S
scan left, scan right	5	6	K	C	A	I	E	L	E	P	U	T	M	Q	R	X	O	S
exchange	5	6	K	C	A	I	E	E	L	P	U	T	M	Q	R	X	O	S
scan left, scan right	6	5	K	C	A	I	E	E	L	P	U	T	M	Q	R	X	O	S
final exchange	6	5	E	C	A	I	E	K	L	P	U	T	M	Q	R	X	O	S
result	6	5	E	C	A	I	E	K	L	P	U	T	M	Q	R	X	O	S

Partitioning trace (array contents before and after each exchange)

Quicksort trace



Quicksort trace (array contents after each partition)

Hegðun quicksort

Grf. skiptingu sem skiptir alltaf eftir fyrsta gildi í svæðinu

Hvernig hegðar quicksort sér fyrir mismunandi tegundir af inntaki?

Í besta tilfelli?

En í versta tilfelli?

Í besta tilfelli, jöfn skipting – $O(n \log n)$

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

Í versta tilfelli, línuleg endurkvæmni – $O(n^2)$

			a[ ]													
lo	j	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	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
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

Endurbætur

Hvað er þá til ráða?

Hvernig er hægt að gera quicksort betra?

Endurbætur

Hvað er þá til ráða?

Hvernig er hægt að gera quicksort betra?

Svar: Veljum betra vendistak

Vendistak

En hvernig getum við valið betra vendistak?

- Það er best að skipta eftir miðgildinu
- en við vitum ekki miðgildið

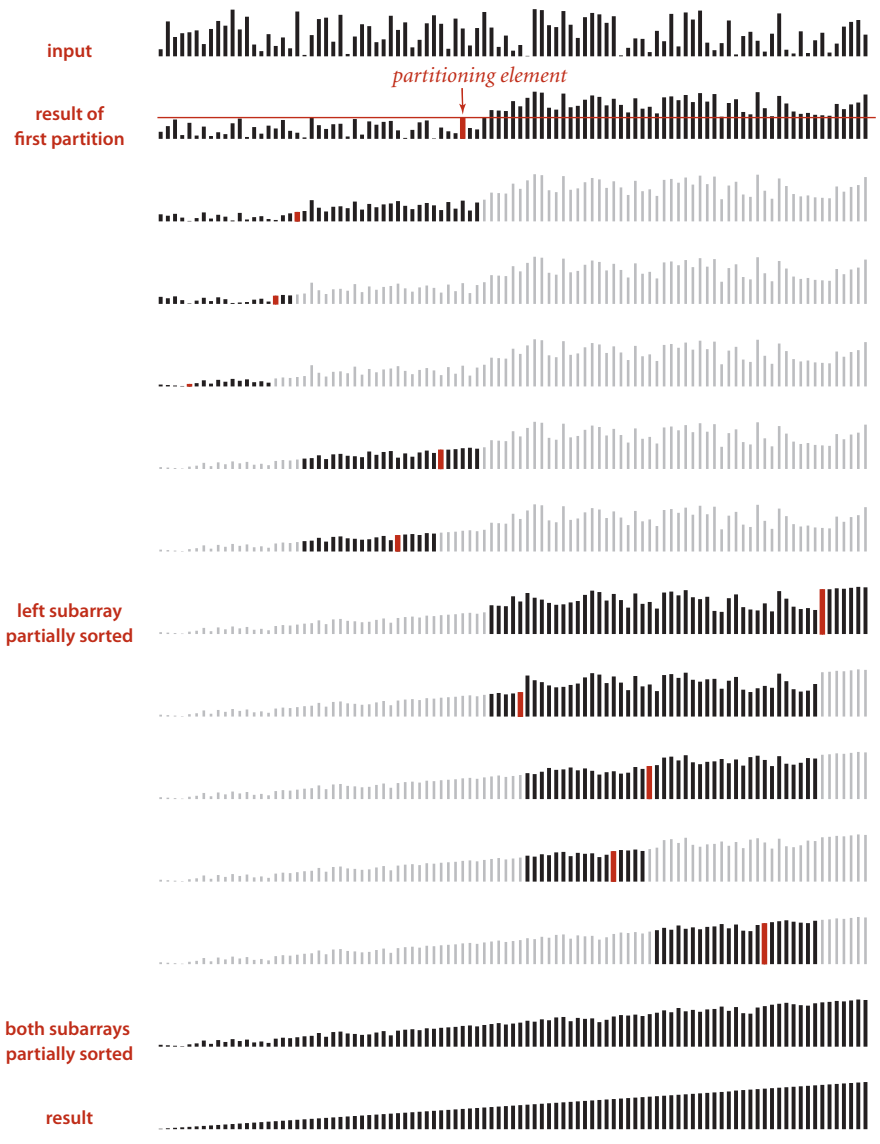
Quickselect:

- Hægt að finna miðgildið á $O(n)$ tíma með quickselect
- en það er mjög dýrt og almennt ekki gert í praxís

Millivegur:

- Compromise: median of three
- Tökum þrjú sýni úr svæðinu og notum miðgildi þeirra sem vendistak. Reynist almennt nokkuð vel.

Visual trace of quicksort w/ median of 3 and cutoff



Quicksort with median-of-3 partitioning and cutoff for small subarrays

Annar möguleiki

Vendistak valið af handahófi:

- Veljum vendistak af handahófi með slembigjafa
- eða stokkum öllu fylkinu í upphafi áður en við röðum

Línuleg stokkun (Knuth shuffle):

```
public static void shuffle(int[] f, Random g)
{
    for (int i = 0; i < f.length; i++)
    {
        int r = g.nextInt(i + 1);
        swap(f, i, r);
    }
}
```

Jöfn gildi

Quicksort með tveggja svæða skiptingu (2-way partitioning) hegðar sér ekki vel þegar fylkið inniheldur mjög mikið af jöfnum gildum

Ef fylkið inniheldur bara jöfn gildi þá er tímaflækjan $O(n^2)$ – fáum n skiptingar og hvert skipting kostar $O(n)$

Hvað er þá til ráða?

Jöfn gildi

Quicksort með tveggja svæða skiptingu (2-way partitioning) hegðar sér ekki vel þegar fylkið inniheldur mjög mikið af jöfnum gildum

Ef fylkið inniheldur bara jöfn gildi þá er tímaflækjan $O(n^2)$ – fáum n skiptingar og sérhver skipting kostar $O(n)$

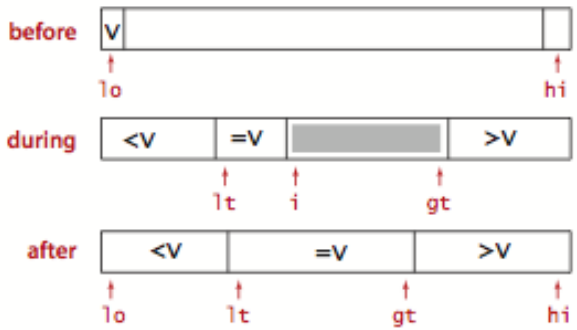
Hvað er þá til ráða?

Skiptum í þrjú svæði með eitt svæði sem inniheldur jöfn gildi

Þriggja svæða skipting (3-way partitioning):

- Dijkstra 3-way
- Bentley-McIlroy 3-way

Dijkstra 3-way partioning trace



3-way partitioning overview

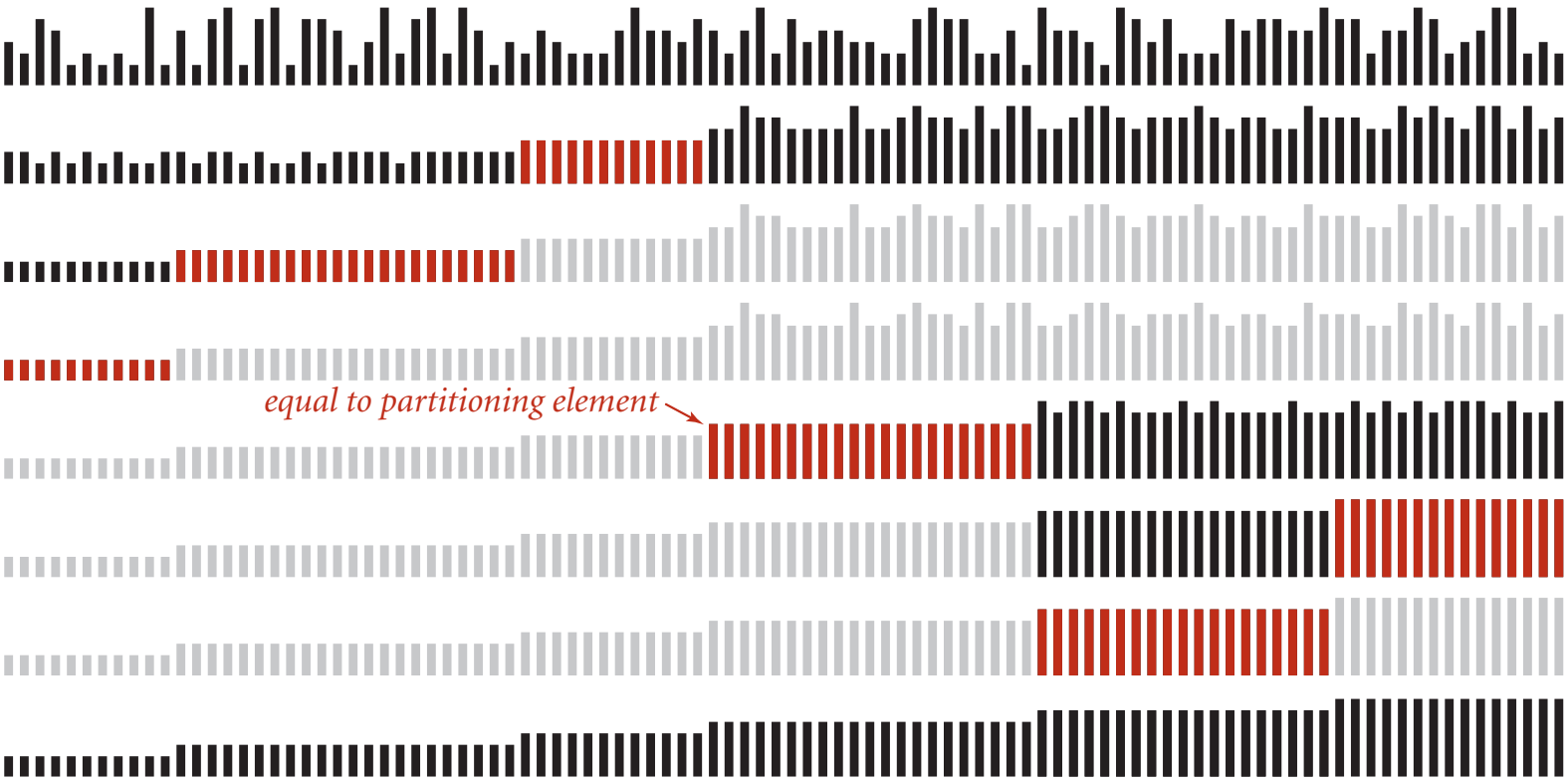
			v	a[]										
lt	i	gt		0	1	2	3	4	5	6	7	8	9	10	11
0	0	11		R	B	W	W	R	W	B	R	R	W	B	R
0	1	11		R	B	W	W	R	W	B	R	R	W	B	R
1	2	11		B	R	W	W	R	W	B	R	R	W	B	R
1	2	10		B	R	R	W	R	W	B	R	R	W	B	W
1	3	10		B	R	R	W	R	W	B	R	R	W	B	W
1	3	9		B	R	R	B	R	W	B	R	R	W	W	W
2	4	9		B	B	R	R	R	W	B	R	R	W	W	W
2	5	9		B	B	R	R	R	W	B	R	R	W	W	W
2	5	8		B	B	R	R	R	W	B	R	R	W	W	W
2	5	7		B	B	R	R	R	R	B	R	W	W	W	W
2	6	7		B	B	R	R	R	R	B	R	W	W	W	W
3	7	7		B	B	B	R	R	R	R	R	W	W	W	W
3	8	7		B	B	B	R	R	R	R	R	W	W	W	W
3	8	7		B	B	B	R	R	R	R	R	W	W	W	W

3-way partitioning trace (array contents after each loop iteration)

Dijkstra 3-way quicksort

```
// quicksort the subarray a[lo .. hi] using 3-way partitioning
static void qsort(Comparable[] a, int lo, int hi)
{
    if (hi <= lo) return;
    int lt = lo, gt = hi;
    Comparable v = a[lo];
    int i = lo;
    while (i <= gt)
    {
        int cmp = a[i].compareTo(v);
        if      (cmp < 0) swap(a, lt++, i++);
        else if (cmp > 0) swap(a, i, gt--);
        else              i++;
    }
    // a[lo..lt-1] < v = a[lt..gt] < a[gt+1..hi].
    qsort(a, lo, lt-1);
    qsort(a, gt+1, hi);
}
```

Visual trace of quicksort w/ Dijkstra 3-way



Visual trace of quicksort with 3-way partitioning

Quicksort 3-way animation

http://www.sorting-algorithms.com/quick-sort-3-way