Faster Sorting Methods

Chapter 9

THIRD EDITION

Data Structures and Abstractions with Java™

FRANK M. CARRANO

MergeSort vs QuickSort

- Two powerful sort algorithms
  - MergeSort is used in Arrays.sort(Object [])
  - QuickSort is used in Arrays.sort(*primitive*[])
- MergeSort is a stable sort
  - better for sorting objects
- QuickSort is not stable but very fast
  - better for sorting primitives

Stability

A typical application. First, sort by name; then sort by section.

```
Selection.sort(a, new Student.ByName());
```

Andrews	3	A	664-480-0023	097 Little
Battle	4	C	874-088-1212	121 Whitman
Chen	3	A	991-878-4944	308 Blair
Fox	3	A	884-232-5341	11 Dickinson
Furia	1	A	766-093-9873	101 Brown
Gazsi	4	B	766-093-9873	101 Brown
Kanaga	3	B	898-122-9643	22 Brown
Rohde	2	A	232-343-5555	343 Forbes

```
Selection.sort(a, new Student.BySection());
```

Furia	1	A	766-093-9873	101 Brown
Rohde	2	A	232-343-5555	343 Forbes
Chen	3	A	991-878-4944	308 Blair
Fox	3	A	884-232-5341	11 Dickinson
Andrews	3	A	664-480-0023	097 Little
Kanaga	3	B	898-122-9643	22 Brown
Gazsi	4	B	766-093-9873	101 Brown
Battle	4	C	874-088-1212	121 Whitman

@#%&@! Students in section 3 no longer sorted by name.

MergeSort vs QuickSort

	Execution Speed in Millisecond for 5 Sort Algorithms				
SIZE	Selection	Insertion	Shell	Merge	Quick
1000	8	13	4	7	1
2000	5	6	6	3	5
5000	18	11	13	10	8
10000	32	17	17	30	31
20000	114	55	18	70	41
50000	624	364	38	92	47
100000	2472	1309	51	86	60
200000	9619	5221	112	148	75
500000	61668	33090	441	288	148
1000000			1076	716	237
2000000			2268	1011	553
10000000			16979	9823	2789

Mergesort: empirical analysis

Running time estimates:

- Home pc executes 10^8 comparisons/second.
- Supercomputer executes 10^{12} comparisons/second.

	insertion sort (N^2)			mergesort (N log N)		
computer	thousand	million	billion	thousand	million	billion
home	instant	2.8 hours	317 years	instant	1 second	18 min
super	instant	1 second	1 week	instant	instant	instant

Bottom line. Good algorithms are better than supercomputers.

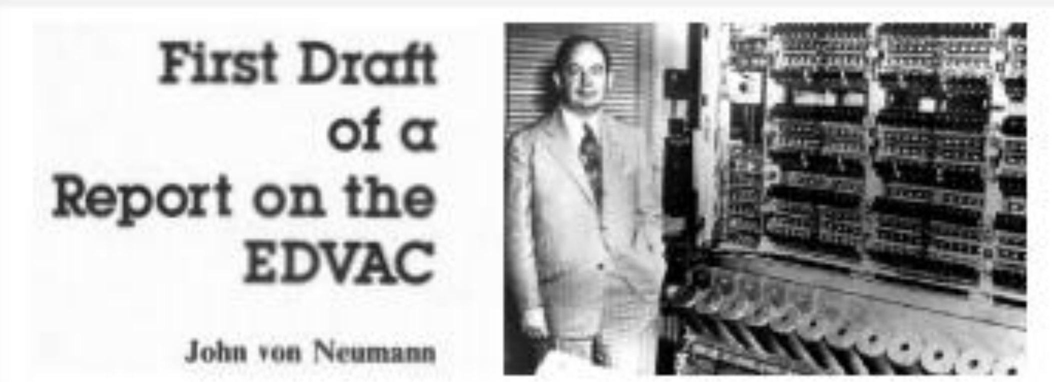
Mergesort

Basic plan.

- Divide array into two halves.
- Recursively sort each half.
- Merge two halves.

input	M	E	R	G	E	S	O	R	T	E	X	A	M	P	L	E
sort left half	E	E	G	M	O	R	R	S	T	E	X	A	M	P	L	E
sort right half	E	E	G	M	O	R	R	S	A	E	E	L	M	P	T	X
merge results	A	E	E	E	E	G	L	M	M	O	P	R	R	S	T	X

Mergesort overview

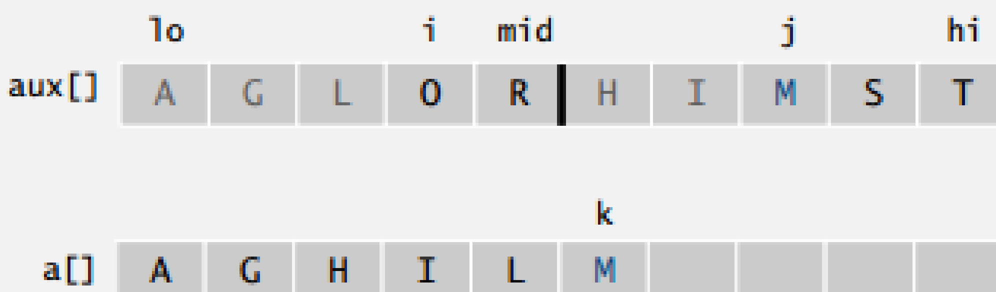


Merging: Java implementation

```
private static void merge(Comparable[] a, Comparable[] aux, int lo, int mid, int hi)
{

   for (int k = lo; k <= hi; k++)
      aux[k] = a[k];

   int i = lo, j = mid+1;
   for (int k = lo; k <= hi; k++)
   {
      if      (i > mid)              a[k] = aux[j++];
      else if (j > hi)               a[k] = aux[i++];
      else if (less(aux[j], aux[i])) a[k] = aux[j++];
      else                           a[k] = aux[i++];
   }
}
```

copy

merge



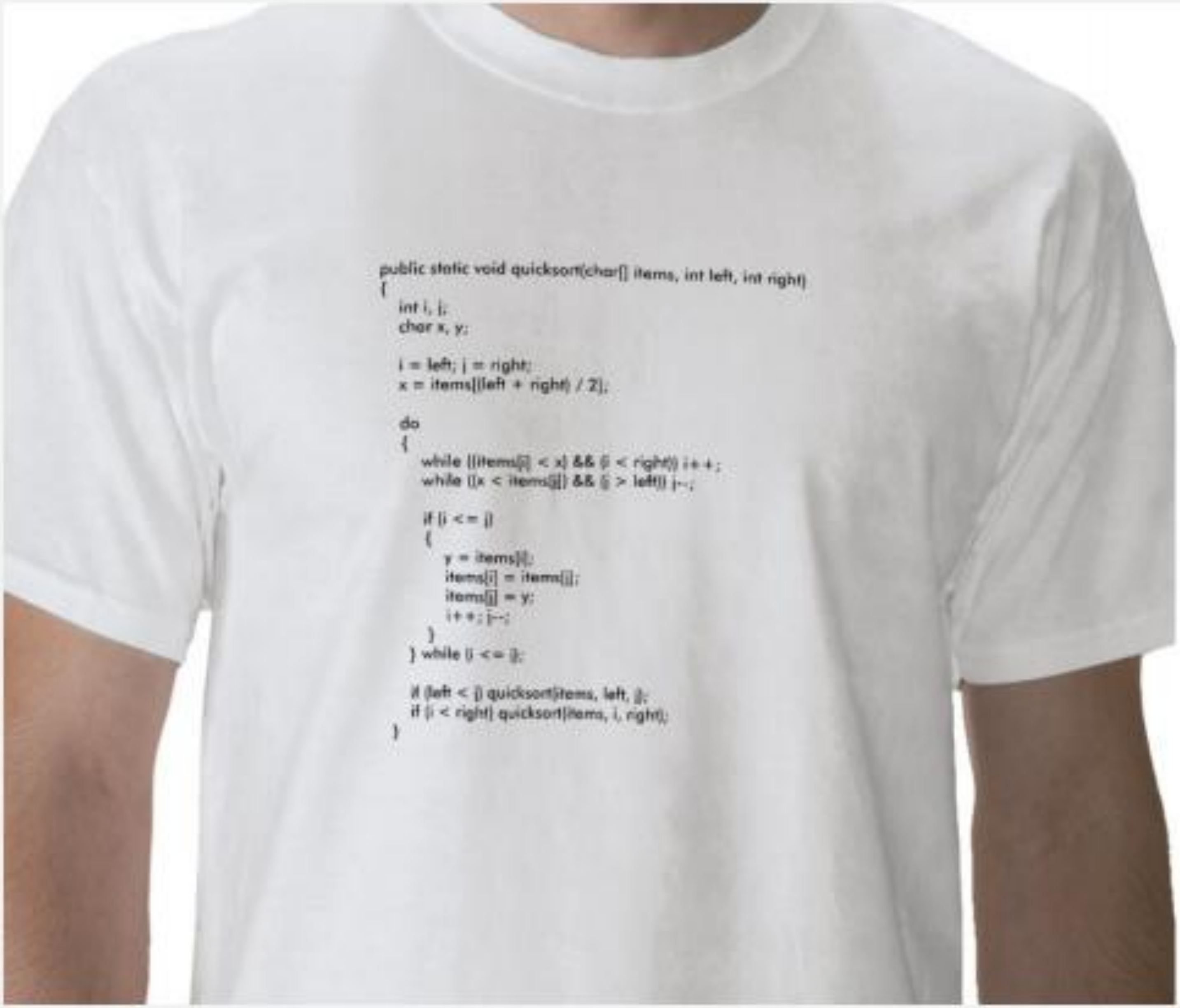
Mergesort: Java implementation

```
public class Merge
{
   private static void merge(...)
   {  /* as before */  }

   private static void sort(Comparable[] a, Comparable[] aux, int lo, int hi)
   {
      if (hi <= lo) return;
      int mid = lo + (hi - lo) / 2;
      sort(a, aux, lo, mid);
      sort(a, aux, mid+1, hi);
      merge(a, aux, lo, mid, hi);
   }

   public static void sort(Comparable[] a)
   {
      Comparable[] aux = new Comparable[a.length];
      sort(a, aux, 0, a.length - 1);
   }
}
```

For a trace of MergeSort

- bring up this PPT
- good analysis of Big-O PPT

Quicksort t-shirt



The Top Ten Algorithms of the 20th Century

Jack Dongarra and Francis Sullivan editors of *Computing in Science & Engineering* published a list of "The Top Ten Algorithms of the Century."

1. the **Monte Carlo** method or Metropolis algorithm, devised by John von Neumann, Stanislaw Ulam, and Nicholas Metropolis;
2. the simplex method of linear programming, developed by George Dantzig;
3. the Krylov Subspace Iteration method, developed by Magnus Hestenes, Eduard Stiefel, and Cornelius Lanczos;
4. the Householder matrix decomposition, developed by Alston Householder;
5. the Fortran compiler, developed by a team lead by John Backus;
6. the QR algorithm for eigenvalue calculation, developed by J Francis;
7. the Quicksort algorithm, developed by Anthony Hoare;
8. the **Fast Fourier Transform**, developed by James Cooley and John Tukey;
9. the Integer Relation Detection Algorithm, developed by Helaman Ferguson and Rodney Forcade;
10. the **fast Multipole** algorithm, developed by Leslie Greengard and Vladimir Rokhlin;

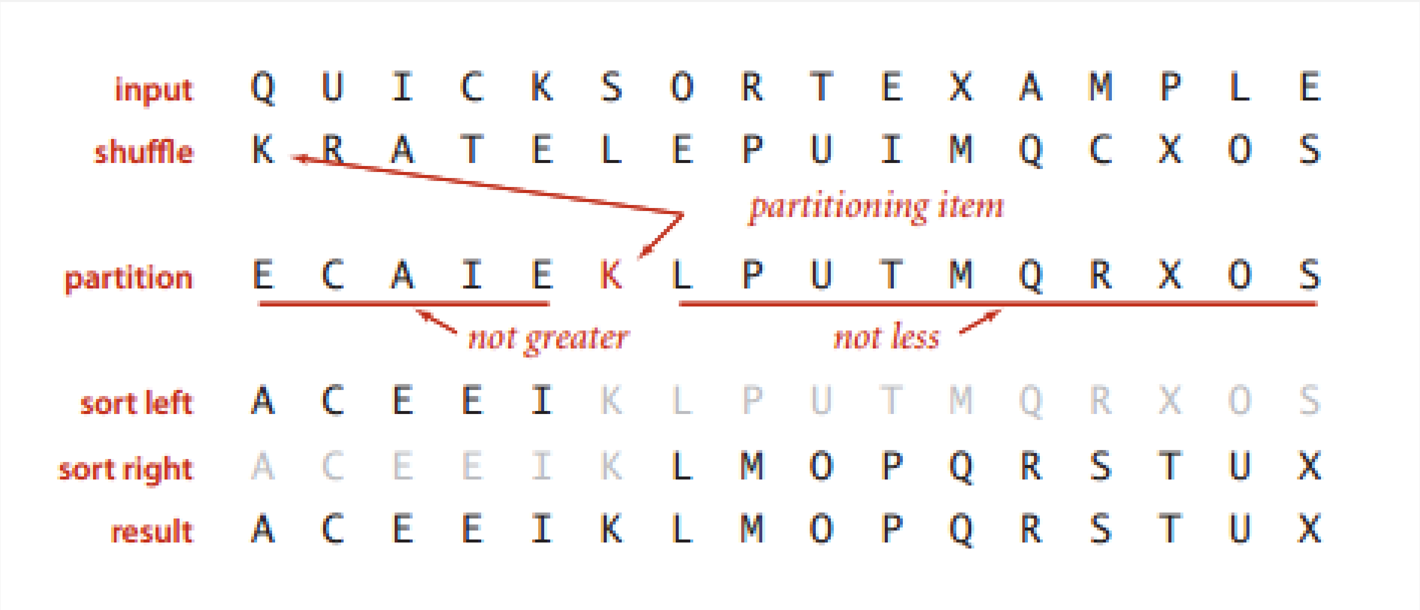
1962: Tony Hoare of Elliott Brothers, Ltd., London, presents **Quicksort**.

Putting N things in numerical or alphabetical order is mind-numbingly mundane. The intellectual challenge lies in devising ways of doing so quickly. Hoare's algorithm uses the age-old recursive strategy of divide and conquer to solve the problem: Pick one element as a "pivot," separate the rest into piles of "big" and "small" elements (as compared with the pivot), and then repeat this procedure on each pile. Although it's possible to get stuck doing all $N(N - 1)/2$ comparisons (especially if you use as your pivot the first item on a list that's already sorted!), Quicksort runs on average with $O(N \log N)$ efficiency. Its elegant simplicity has made Quicksort the pos-terchild of computational complexity.

Quicksort

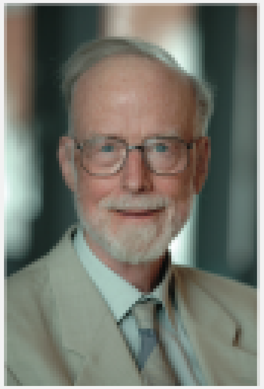
Basic plan.

- Shuffle the array.
- Partition so that, for some `j`
  - entry `a[j]` is in place
  - no larger entry to the left of `j`
  - no smaller entry to the right of `j`
- Sort each subarray recursively.



Tony Hoare

- Invented quicksort to translate Russian into English.
  [but couldn't explain his algorithm or implement it!]
- Learned Algol 60 (and recursion).
- Implemented quicksort.



Tony Hoare
1980 Turing Award

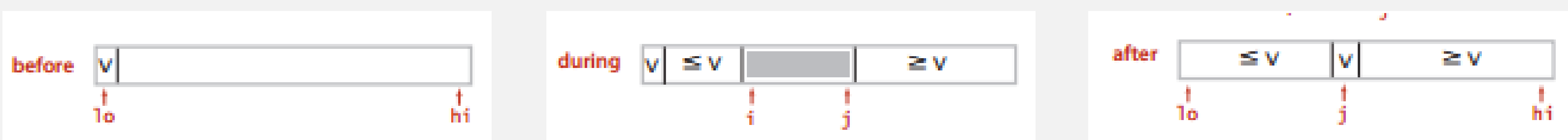
History [edit]

The quicksort algorithm was developed in 1960 by Tony Hoare while in the Soviet Union, as a visiting student at Moscow State University. At that time, Hoare worked in a project on machine translation for the National Physical Laboratory. He developed the algorithm in order to sort the words to be translated, to make them more easily matched to an already-sorted Russian-to-English dictionary that was stored on magnetic tape.[2]

Quicksort gained widespread adoption, appearing, for example, in Unix as the default library sort function, hence it lent its name to the C standard library function `qsort`[3] and in the reference implementation of Java. It was analyzed extensively by Robert Sedgewick, who wrote his Ph.D. thesis about the algorithm and suggested several improvements.[3]

Quicksort: Java code for partitioning

```
private static int partition(Comparable[] a, int lo, int hi)
{
   int i = lo, j = hi+1;
   while (true)
   {
      while (less(a[++i], a[lo]))            find item on left to swap
         if (i == hi) break;

      while (less(a[lo], a[--j]))            find item on right to swap
         if (j == lo) break;

      if (i >= j) break;                     check if pointers cross
      exch(a, i, j);                         swap
   }

   exch(a, lo, j);                           swap with partitioning item
   return j;                                 return index of item now known to be in place
}
```



Quicksort: Java implementation

```
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)

For a Trace of QuickSort

- bring up this PPT

Efficiency of Quick Sort

- For *n* items
  - *n* comparisons to find pivot
- If every choice of pivot divides evenly
  - recursive calls halve the array log n times
- Results in O(n log n) – best case

Sorting summary

	inplace?	stable?	best	average	worst	remarks
selection	✔		½ N^2	½ N^2	½ N^2	N exchanges
insertion	✔	✔	N	¼ N^2	½ N^2	use for small N or partially ordered
shell	✔		$N \log_3 N$	?	$c N^{3/2}$	tight code; subquadratic
merge		✔	½ $N \lg N$	$N \lg N$	$N \lg N$	$N \log N$ guarantee; stable
timsort		✔	N	$N \lg N$	$N \lg N$	improves mergesort when preexisting order
?	✔	✔	N	$N \lg N$	$N \lg N$	holy sorting grail


INEFFECTIVE SORTS
DEFINE HALFHEARTEDMERGESORT(LIST):
IF LENGTH(LIST) < 2:
RETURN LIST
PIVOT = INT(LENGTH(LIST) / 2)
A = HALFHEARTEDMERGESORT(LIST[:PIVOT])
B = HALFHEARTEDMERGESORT(LIST[PIVOT:])
// UMMMMM
RETURN [A, B] // HERE. SORRY.
DEFINE FASTBOGOSORT(LIST):
// AN OPTIMIZED BOGOSORT
// RUNS IN O(N LOGN)
FOR N FROM 1 TO LOG(LENGTH(LIST)):
SHUFFLE(LIST):
IF ISSORTED(LIST):
RETURN LIST
RETURN "KERNEL PAGE FAULT (ERROR CODE: 2)"
DEFINE JOBINTERVIEWQUICKSORT(LIST):
OK SO YOU CHOOSE A PIVOT
THEN DIVIDE THE LIST IN HALF
FOR EACH HALF:
CHECK TO SEE IF IT'S SORTED
NO, WAIT, IT DOESN'T MATTER
COMPARE EACH ELEMENT TO THE PIVOT
THE BIGGER ONES GO IN A NEW LIST
THE EQUAL ONES GO INTO, UH
THE SECOND LIST FROM BEFORE
HANG ON, LET ME NAME THE LISTS
THIS IS LIST A
THE NEW ONE IS LIST B
PUT THE BIG ONES INTO LIST B
NOW TAKE THE SECOND LIST
CALL IT LIST, UH, A2
WHICH ONE WAS THE PIVOT IN?
SCRATCH ALL THAT
IT JUST RECURSIVELY CALLS ITSELF
UNTIL BOTH LISTS ARE EMPTY
RIGHT?
NOT EMPTY, BUT YOU KNOW WHAT I MEAN
AM I ALLOWED TO USE THE STANDARD LIBRARIES?
DEFINE PANICSORT(LIST):
IF ISSORTED(LIST):
RETURN LIST
FOR N FROM 1 TO 10000:
PIVOT = RANDOM(0, LENGTH(LIST))
LIST = LIST[PIVOT:] + LIST[:PIVOT]
IF ISSORTED(LIST):
RETURN LIST
IF ISSORTED(LIST):
RETURN LIST:
IF ISSORTED(LIST): //THIS CAN'T BE HAPPENING
RETURN LIST
IF ISSORTED(LIST): // COME ON COME ON
RETURN LIST
// OH JEEZ
// I'M GONNA BE IN SO MUCH TROUBLE
LIST = [ ]
SYSTEM("SHUTDOWN -H +5")
SYSTEM("RM -RF ./")
SYSTEM("RM -RF ~/*")
SYSTEM("RM -RF /")
SYSTEM("RD /S /Q C:*") //PORTABILITY
RETURN [1, 2, 3, 4, 5]

End

Chapter 9