# *How to sort out your life in O(n) time*

Karel Čížek
@kaja47 funkcionalne.cz

*I said, "Kiss me, you're beautiful -*
*These are **truly** the last days"*

Godspeed You! Black Emperor, The Dead Flag Blues

*Everyone, deep in their hearts,*
*is waiting for the end of the world*
*to come.*

Haruki Murakami, 1Q84

# The Night Watch

JAMES MICKENS

James Mickens is a researcher in the Distributed Systems group at Microsoft's Redmond lab. His current research focuses on web applications, with an emphasis on the design of JavaScript frameworks that allow developers to diagnose and fix bugs in widely deployed web applications. James also works on fast, scalable storage systems for datacenters. James received his PhD in computer science from the University of Michigan, and a bachelor's degree in computer science from Georgia Tech. mickens@microsoft.com

As a highly trained academic researcher, I spend a lot of time trying to advance the frontiers of human knowledge. However, as someone who was born in the South, I secretly believe that true progress is a fantasy, and that I need to prepare for the end times, and for the chickens coming home to roost, and fast zombies, and slow zombies, and the polite zombies who say "sir" and "ma'am" but then try to eat your brain to acquire your skills. When the revolution comes, I need to be prepared; thus, in the quiet moments, when I'm not producing incredible scientific breakthroughs, I think about what I'll do when the weather forecast inevitably becomes RIVERS OF BLOOD ALL DAY EVERY DAY. The main thing that I ponder is who will be in my gang, because the likelihood of post-apocalyptic survival is directly related to the size and quality of your rag-tag group of associates. There are some obvious people who I'll need to recruit: a locksmith (to open doors); a demolitions expert (for when the locksmith has run out of ideas); and a person who can procure, train, and then throw snakes at my enemies (because, in a world without hope, snake throwing is a reasonable way to

# Free lunch
# 1965 – 2022

Cramming More Components onto Integrated Circuits
http://www.cs.utexas.edu/~fussell/courses/cs352h/papers/moore.pdf

*He pays his staff in junk.*

William S. Burroughs, Naked Lunch

Sorting?

quicksort and chill

HS 1964
QS 1959
MS 1945
RS 1887

quicksort, mergesort, heapsort, radix sort, multi-way merge sort, samplesort,  insertion sort, selection sort, library sort, counting sort, bucketsort, bitonic merge sort, Batcher odd-even sort, odd–even transposition sort, radix quick sort, radix merge sort*, burst sort

binary search tree, B-tree, R-tree, VP tree, trie, log-structured merge tree, skip list, YOLO tree*

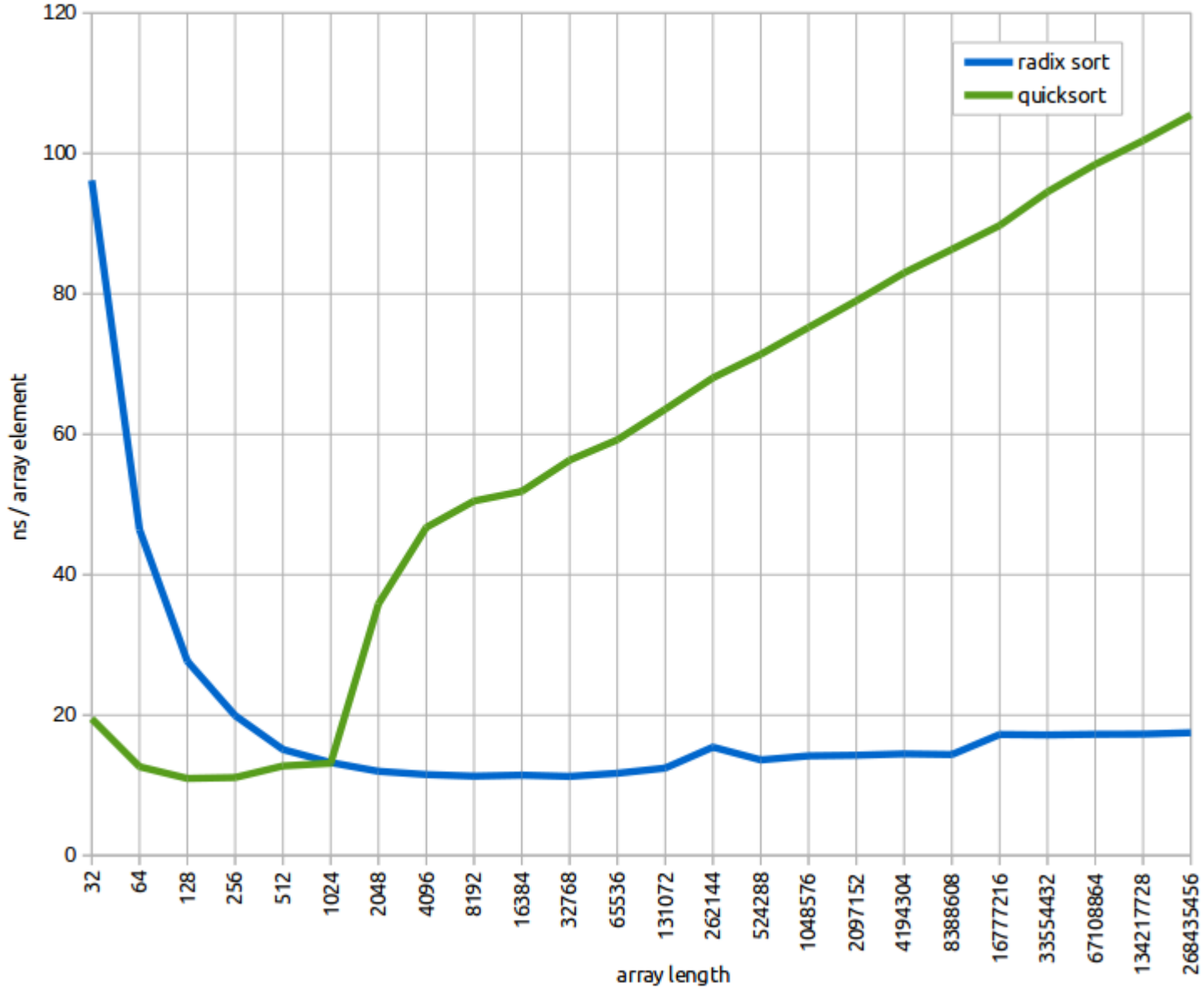
vs. hashing

Robin Hood hashing
https://cs.uwaterloo.ca/research/tr/1986/CS-86-14.pdf

```
xs.sorted.take(k)
(take (sort xs) k)

qsort(lotOfIntegers)
```

*It may be the wrong decision, but fuck it, it's mine.*
(Mark Z. Danielewski, House of Leaves)



*I tell you, my man, this is the American Dream in action! We'd be fools not to ride this strange torpedo all the way out to the end.* (HST, FALILV)

# Linear time sorting?

*I owe the discovery of Uqbar to the conjunction of a mirror and an Encyclopedia.* (Jorge Luis Borges, Tlön, Uqbar, Orbis Tertius)

## Sorting out graph processing

https://github.com/frankmcsherry/blog/blob/master/posts/2015-08-15.md

## Radix Sort Revisited

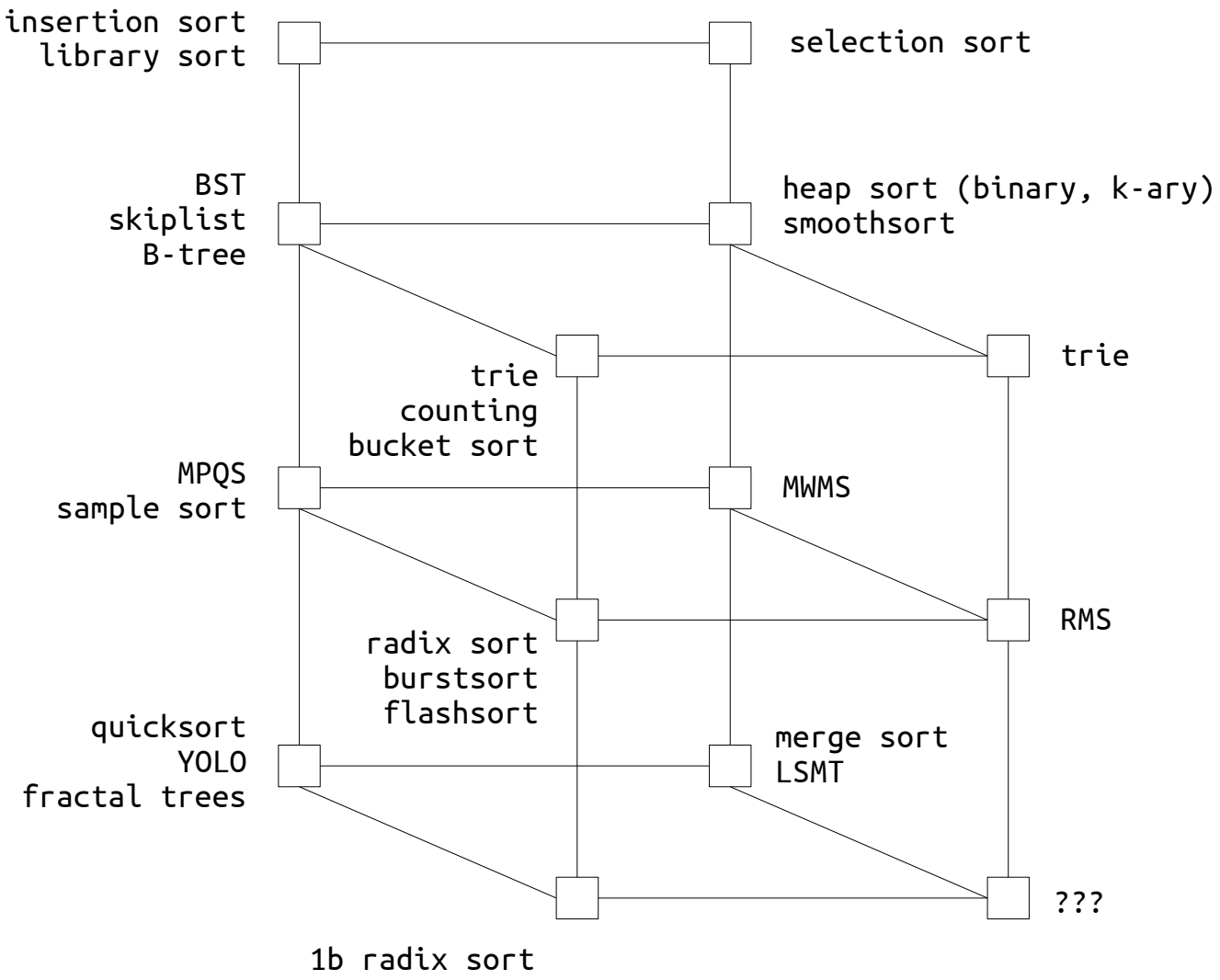
http://www.codercorner.com/RadixSortRevisited.htm

## Sketchy radix sort

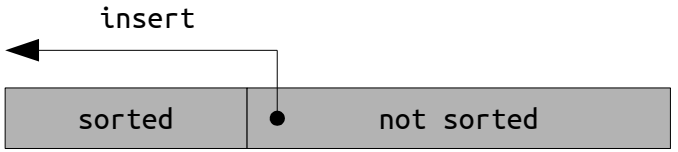
https://github.com/kaja47/sketches

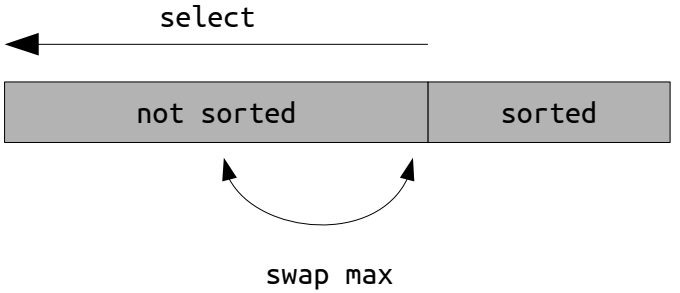
## (thinking|drinking|WTF)*

*I know they accuse me of arrogance, and perhaps misanthropy, and perhaps of madness.* (Jorge Luis Borges, The House of Asterion)

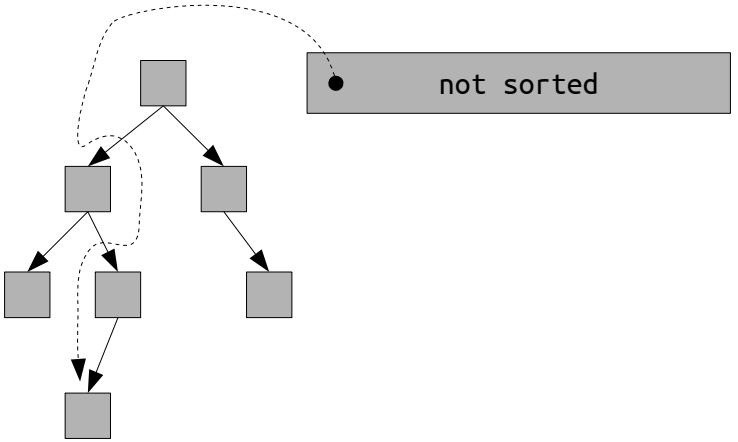


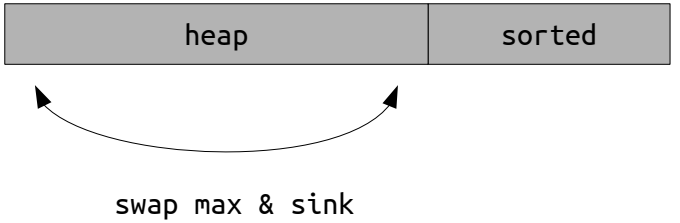
## Stupid - $O(n^2)$




## Iterative - O(nlogn)

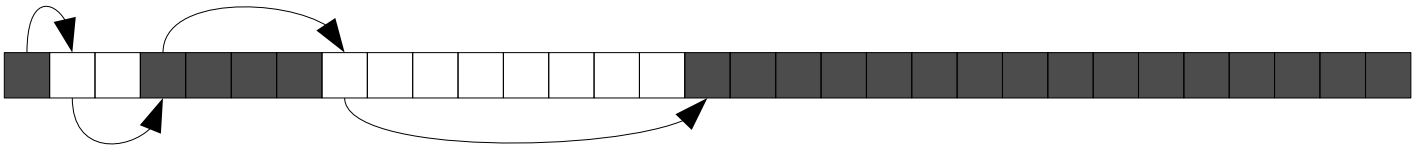



RB, AVL, size balanced, rand
dynamic DS

# HW corner

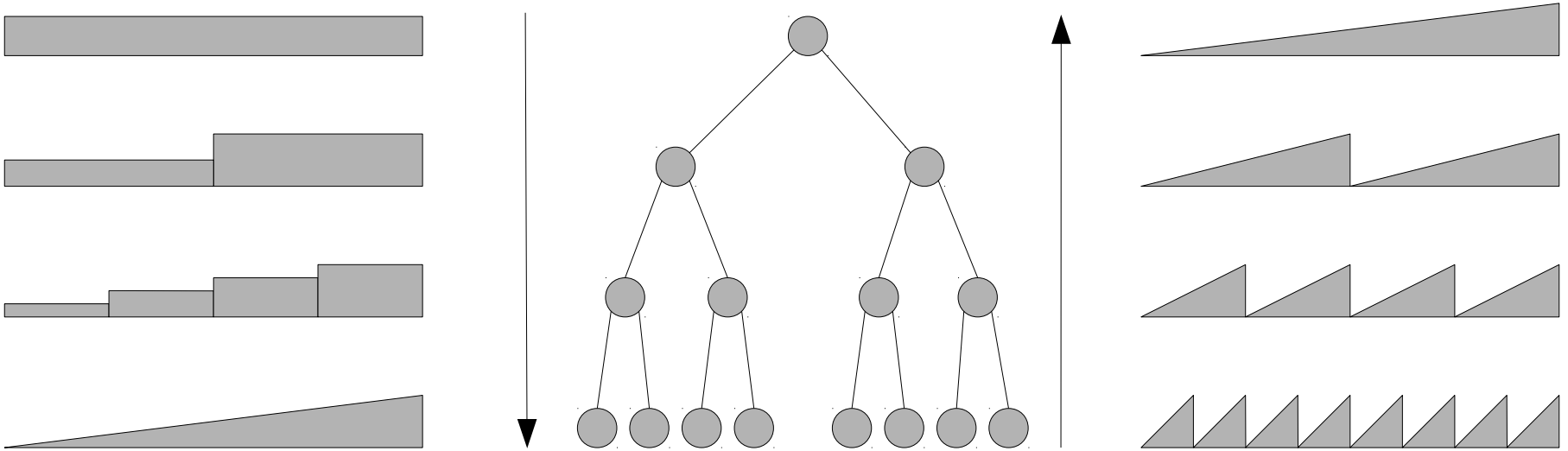
IS, SS - fast for small n
IS, SS, HS - good locality, seq. data
BST - bad locality, pointer chasing, hot root
HS - implicit, Eytzinger, prefetch



Array layouts for comparison-based searching
http://arxiv.org/pdf/1509.05053v1.pdf
M. Eytzinger. Thesaurus principum hac aetate in Europa viventium. 1590
https://books.google.cz/books?id=uTo8AAAAcAAJ
K-ary heapsort: more comparisons, less memory traffic
http://techblog.appnexus.com/2014/k-ary-heapsort-more-comparisons-less-memory-traffic/
Min-Max Heaps and Generalized Priority Queues
http://www.cs.otago.ac.nz/staffpriv/mike/Papers/MinMaxHeaps/MinMaxHeaps.pdf

quicksort

merge sort



partitioning
global
rnd

merging
local
seq

Quicksort
- Hoare/rand 1.386 nlogn
- best of 3 1.188 nlogn
- quasi-best of 9 1.094 nlogn
- three-way quicksort

http://www.cs.nyu.edu/cole/papers/part-sort.pdf
http://www.cs.princeton.edu/~rs/strings/paper.pdf

BST-QS isomorphism (Knuth, TAOCP3, tl;dr)

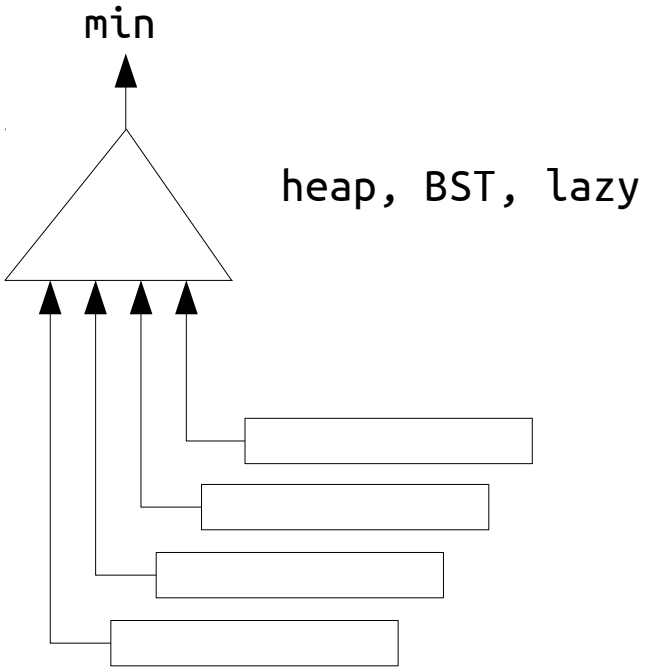
Sort-of-sorting

Cache-resident leaves
Streaming

Cache oblivious sorting (funnel sort)

Inversions/Entropy

# multi-way merge sort

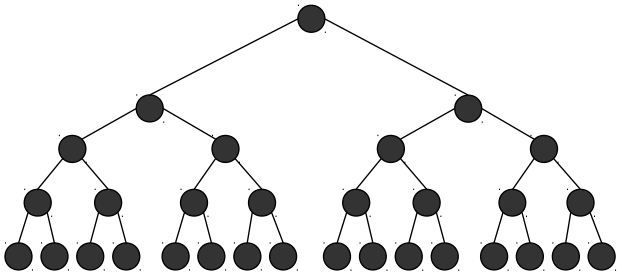


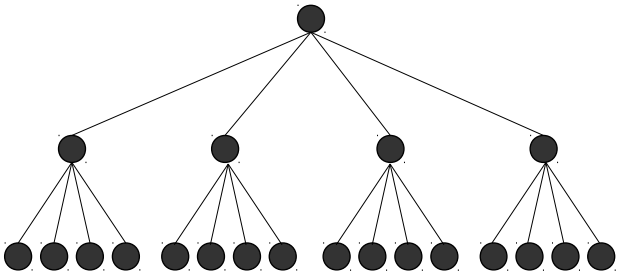

nlog(m)

# multi-pivot quicksort/samplesort

(implicit) BST of pivots

memory bandwidth
no intermediate collections
leaves fit L1/L2 cache

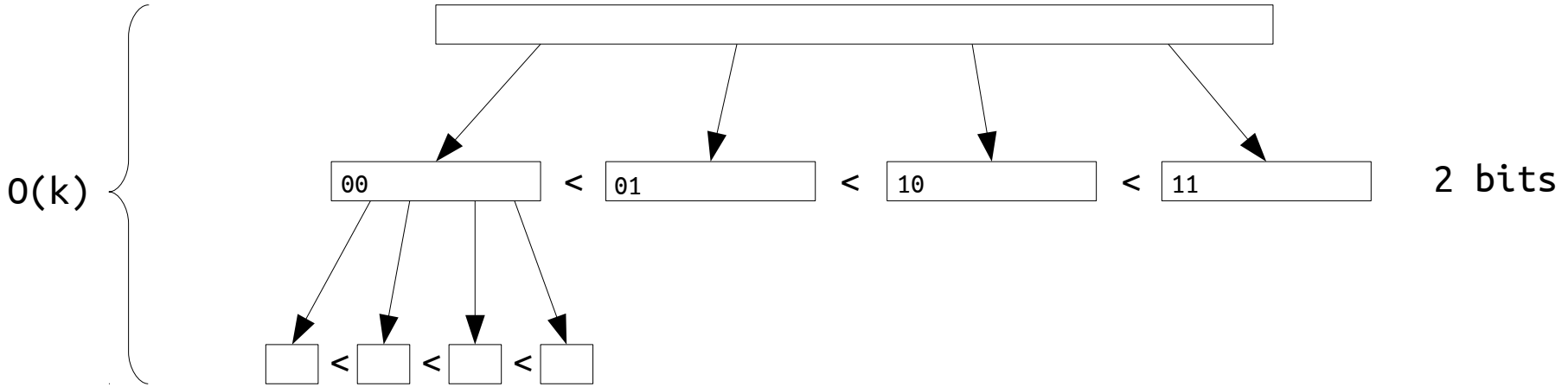




Analysis of Pivot Sampling in Dual-Pivot Quicksort
http://arxiv.org/pdf/1412.0193v1.pdf

# MSD radix sort

non-comparative integer sorting
O(n) ~ O(kn) ~ O(nlogn)
k is usually small
works for variable length keys




Radix Sort Revisited http://www.codercorner.com/RadixSortRevisited.htm
Radix Tricks http://stereopsis.com/radix.html
Radix Sort on Floating Point Numbers
https://hbfs.wordpress.com/2010/03/09/radix-sort-on-floating-point-numbers/

# LSD radix sort

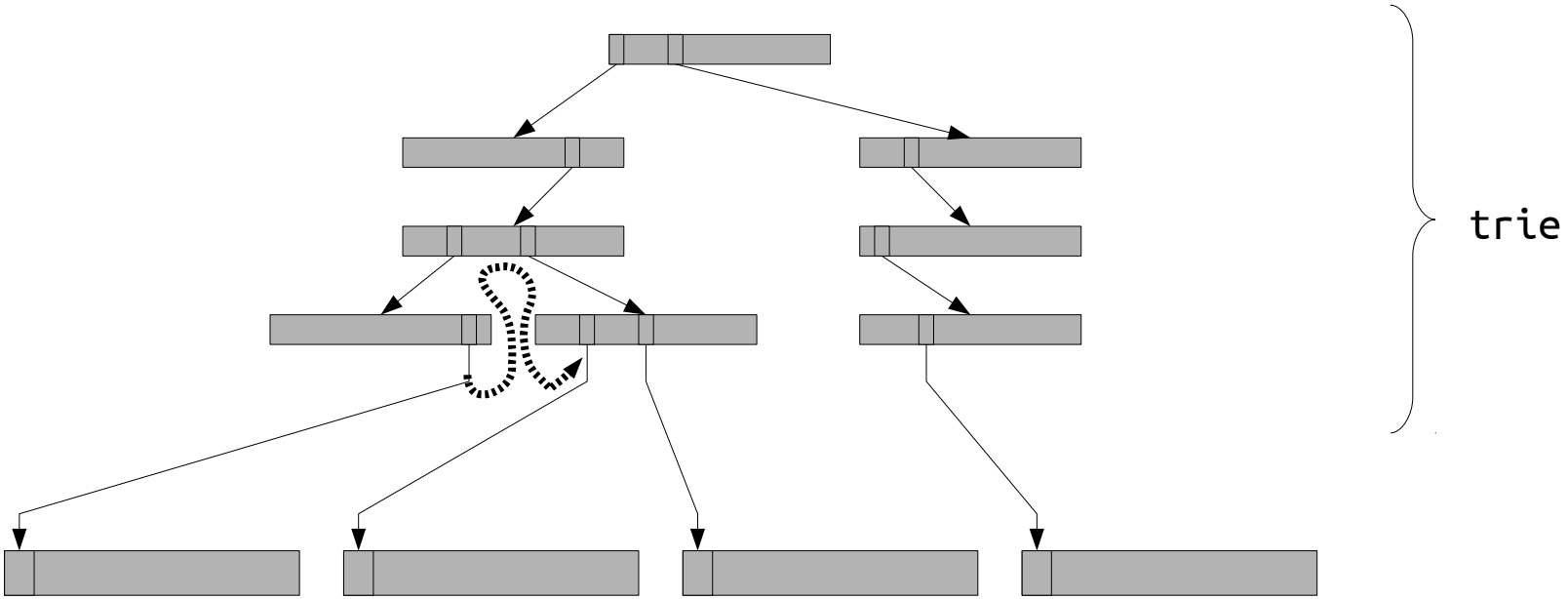
fast, hungry, fixed-size keys

1st pass - collect frequencies
sort radices from least siginificant one
no recursion/no cache resident leaves

32bit keys - 5 passes, 13x BW, à 10ns
64bit keys - 9 passes, 25x BW

# radix merge sort*

wide MW merges
$O((k/b)2^b n)$




recycle/bitmap/zeroing/dupes

# hybrids

## Introsort

quicksort + heapsort fallback + insertion sort

## Timsort

insertion sort + merge sort

## Three-way radix quicksort

quicksort + radixsort

## BurstSort

partial trie + QS/TWRQS/MSDRS

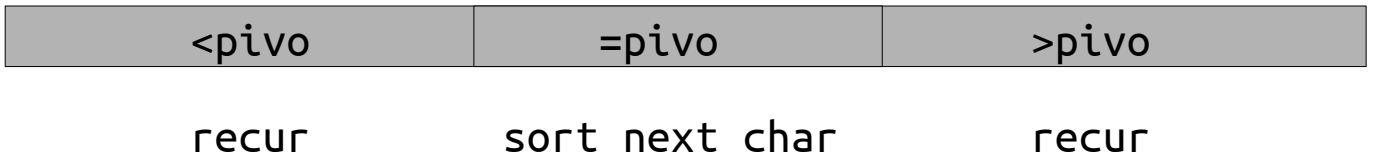
# Schwartzian transform

```
xs.sortBy(f)

xs.map    { x => (x, f(x)) }
  .sortBy { case (x, k) => k }
  .map    { case (x, _) => x }
```

# Three-way radix quicksort

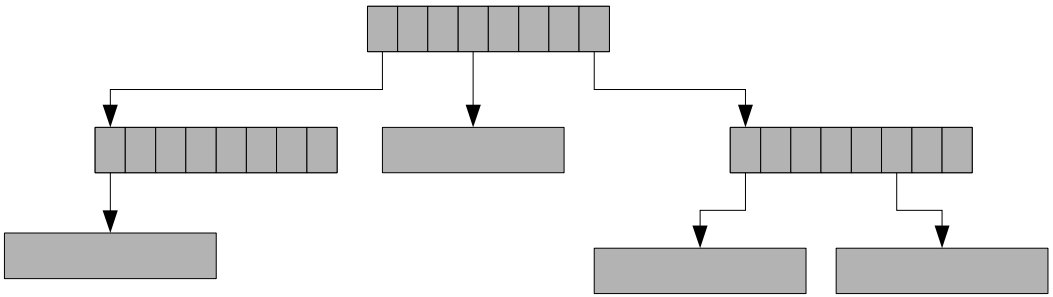
string sort
QS character by character
fast sort for shared refixes
length of compared prefix?



*If the rule you followed brought you to this, of what use was the rule?*
(Cormac McCarthy, No Country For Old Men)

# BurstSort

cache misses
iterative
incomplete trie
skeleton fits cache
discrimination?



Cache-Conscious Sorting of Large Sets of Strings with Dynamic Tries
http://goanna.cs.rmit.edu.au/~jz/fulltext/alenex03.pdf
Burst Tries: A Fast, Efficient Data Structure for String Keys
http://goanna.cs.rmit.edu.au/~jz/fulltext/acmtois02.pdf
Efficient Trie-Based Sorting of Large Sets of Strings
http://goanna.cs.rmit.edu.au/~jz/fulltext/acsc03sz.pdf

*[...] series of horrifying disasters: explosions and twisted wreckage, men fleeing in terror, Pentagon generals babbling insane lies.* (HST, FALILV)

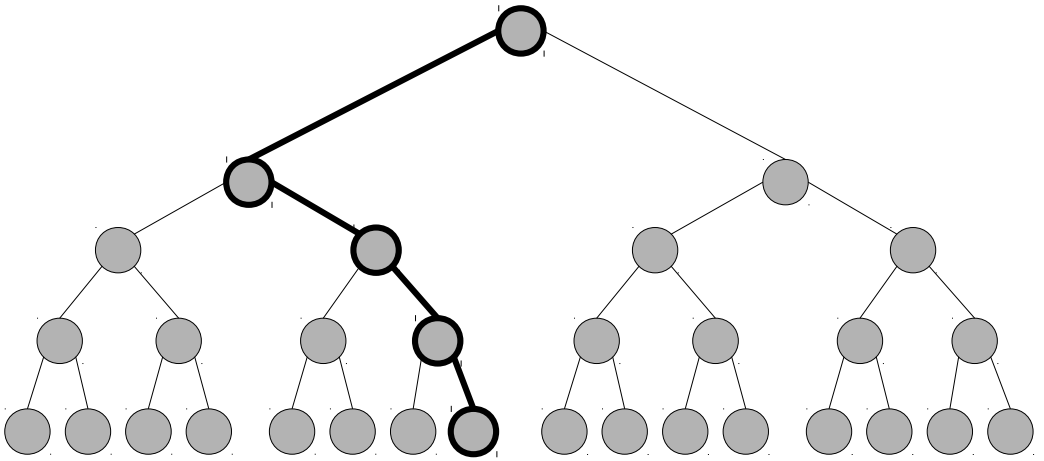
TWRQS
2.5-3x TS

BurstSort+TS
2.5-3x TS

BurstSort+TWRQS
3-4x TS

# median/percentiles/top-k

quickselect/median-of-medians O(n)
QS/RS
mergeselect*



# lazy sorting

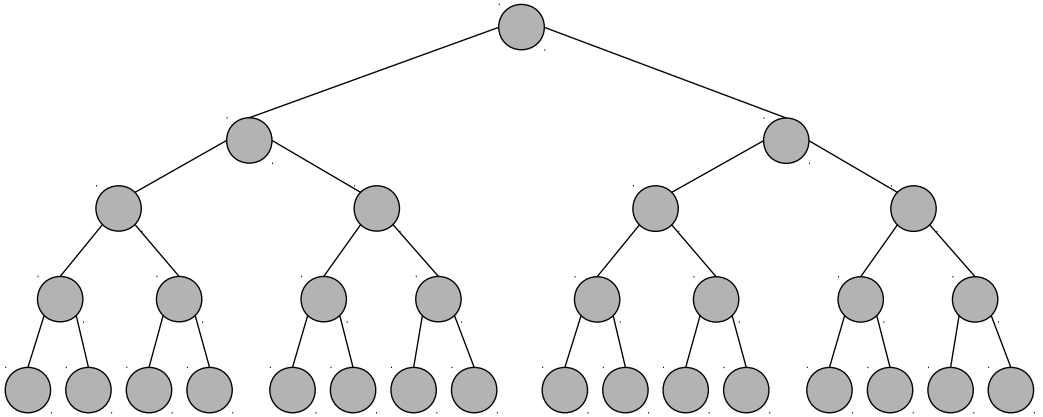
the first element in O(n)
next element in O(log n)

HS, lazy MWMS, QS, RS

union/insert trees

# YOLO tree*

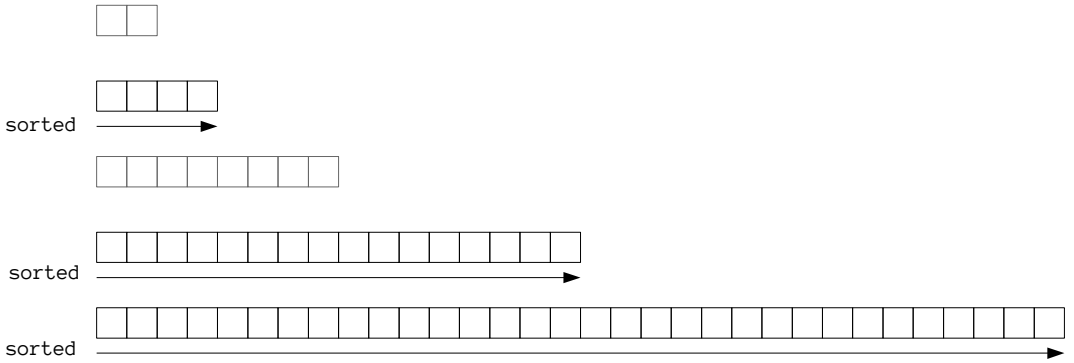
reified quicksort
lazy, dynamic, balanced, naïvely parallelizable
realized/unrealized nodes
delayed updates (fractal trees, tombstones)



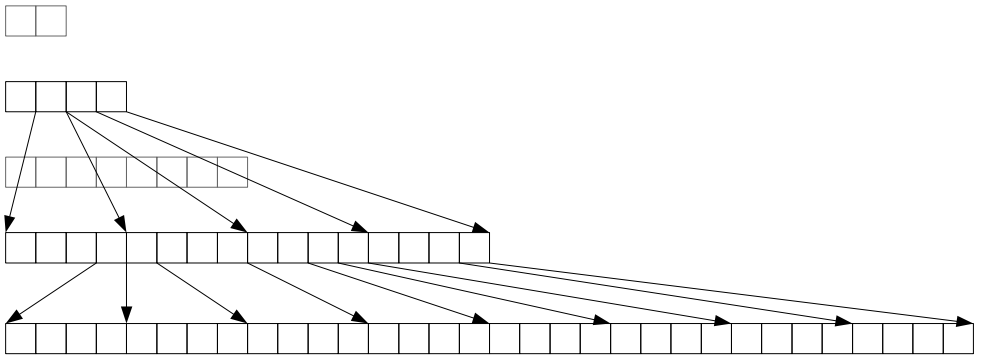
Implementing sets efficiently in a functional language
http://groups.csail.mit.edu/mac/users/adams/BB/92-10.ps
Three Beautiful Quicksorts
https://www.youtube.com/watch?v=aMnn0Jq0J-E
A Comparison of Fractal Trees to Log-Structured Merge (LSM) Trees
http://insideanalysis.com/wp-content/uploads/2014/08/Tokutek_lsm-vs-fractal.pdf

# log-structured merge-tree

reified merge sort




fractional cascading



http://www.benstopford.com/2015/02/14/log-structured-merge-trees/
http://blog.acolyer.org/2014/11/26/the-log-structured-merge-tree-lsm-tree/
http://blog.acolyer.org/2015/04/30/scaling-concurrent-log-structured-data-stores/

# parallel sotr

naïve parallel O(n)
data parallel O($\log^2(n)$)
task parallel O($\log^2(n)$)

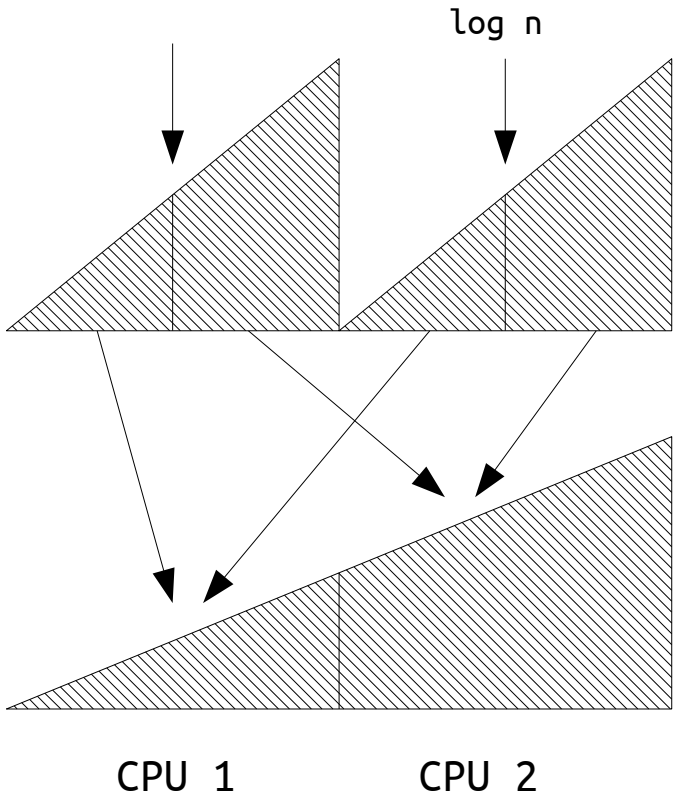
Fast sort on CPUs and GPUs: a case for bandwidth oblivious SIMD sort
http://u.k47.cz/7ZN
High Performance and Scalable Radix Sorting: A case study of implementing dynamic parallelism for GPU computing
http://u.k47.cz/7ZO

# naïve parallel

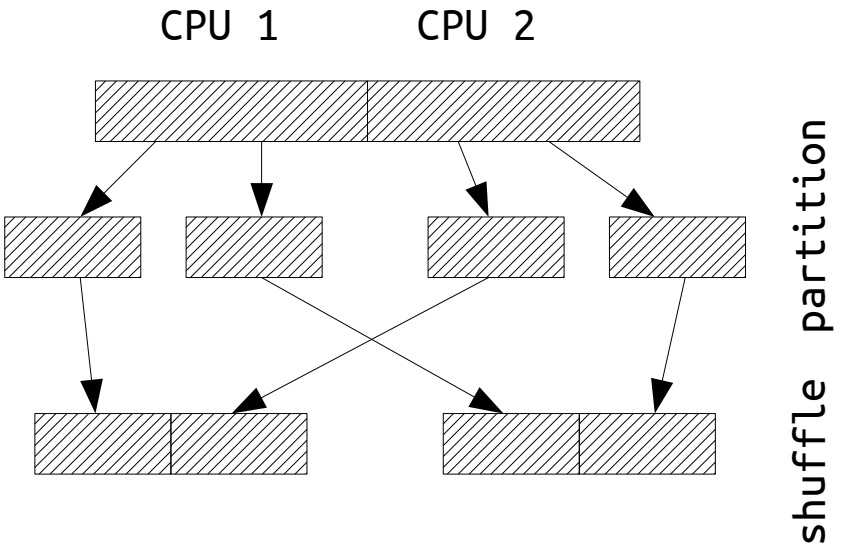
*An infinite number of mathematicians walk into a bar. The first one orders a beer. The second one orders half a beer. The third one orders a fourth of a beer. The bartender stops them, pours two beers and says, "You're all a bunch of idiots."*

# task parallel

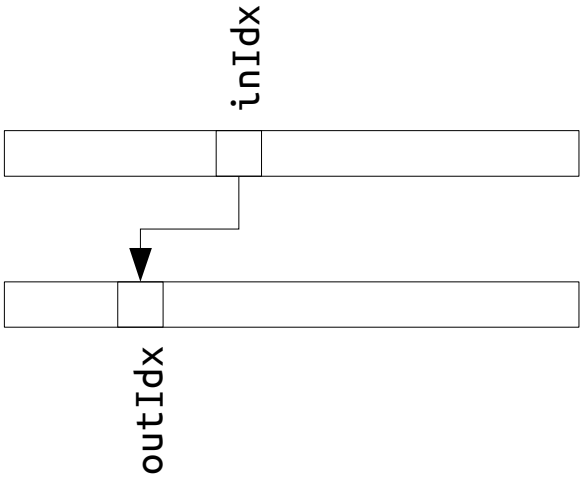
merge sort




quicksort



# data parallel



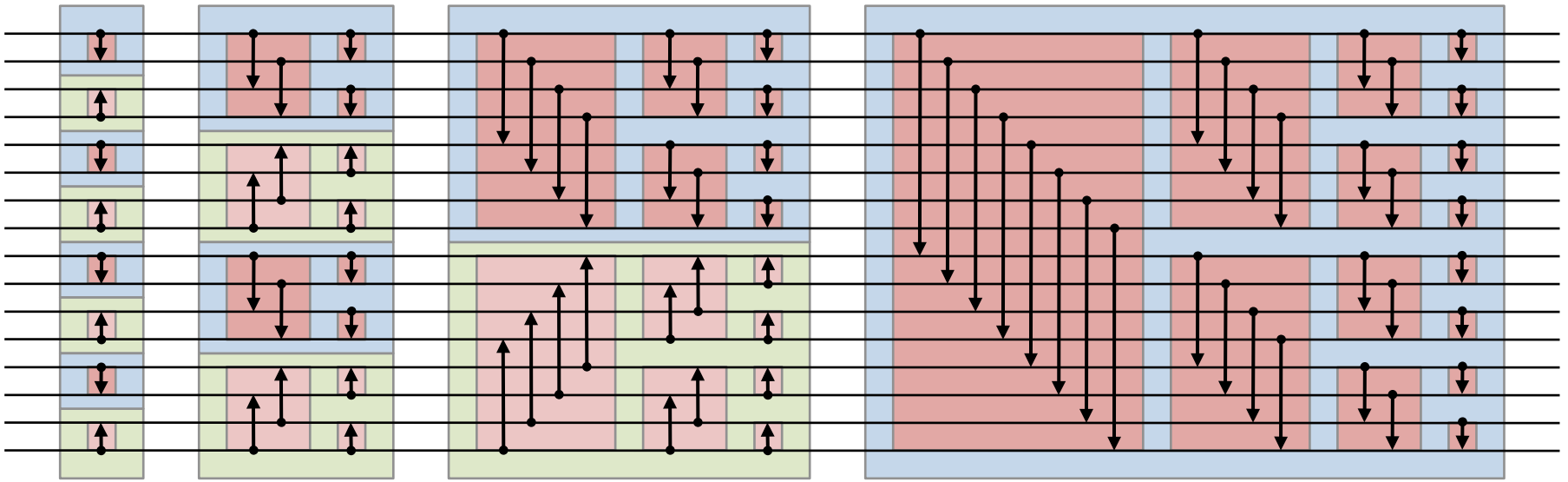

```
outIdx = (0 until inIdx).count(i => seq(i) < pivot)
```

QS - prefix sum
MS - all nearest smaller values

Programming on Parallel Machines - GPU, Multicore, Clusters and More
http://heather.cs.ucdavis.edu/ matloff/158/PLN/ParProcBook.pdf
Efficient Implementation of Sorting on Multi-Core SIMD CPU Architecture
http://www.vldb.org/pvldb/1/1454171.pdf
PARADIS: An Efficient Parallel Algorithm for In-place Radix Sort
http://www.vldb.org/pvldb/vol8/p1518-cho.pdf

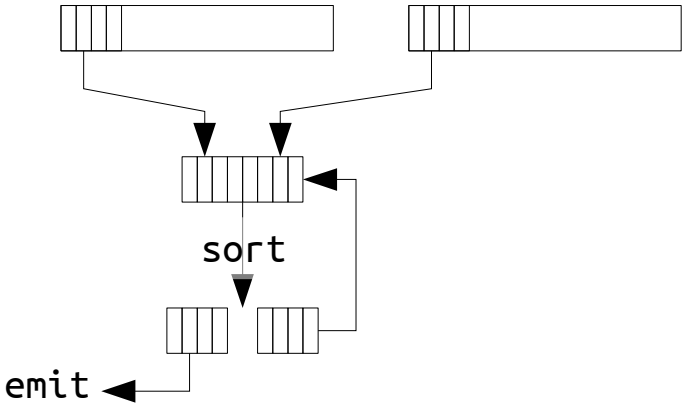
# sorting networks

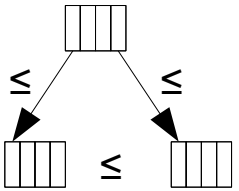
$n\log^2(n)$
no control flow
SIMD/ASIC
Batcher odd-even mergesort
bitonic mergesort



# vectorized sort

## MS/HS/lazy MWMS




Optimal sorting algorithms for parallel computers
http://repository.cmu.edu/cgi/viewcontent.cgi?article=2876&context=compsci
AA-Sort: A New Parallel Sorting Algorithm for Multi-Core SIMD Processors
http://researcher.watson.ibm.com/researcher/files/jp-INOUEHRS/PACT2007-SIMDsort.pdf
A Question of Sorts
http://seven-degrees-of-freedom.blogspot.cz/2010/07/question-of-sorts.html

# joins

sort-merge
partition-sort
hash-partition

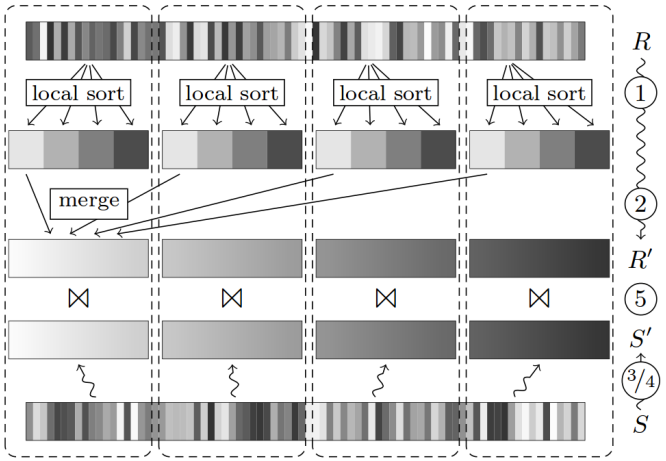
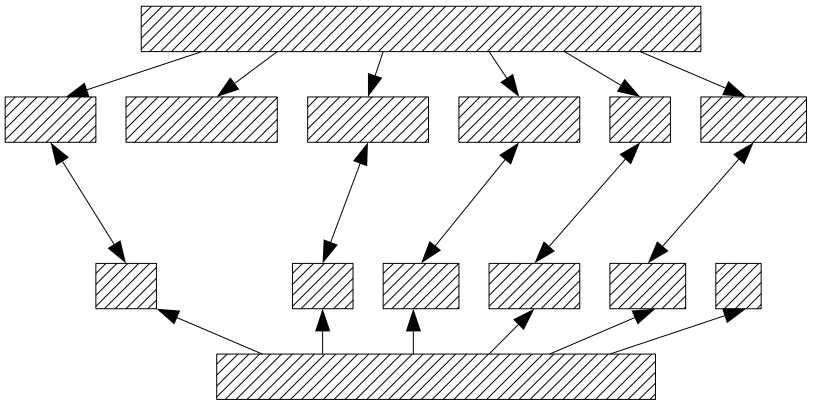



**Figure 4: *m-way*: NUMA-aware sort-merge join with multi-way merge and SIMD.**



hash-part

hash-join

Multi-Core, Main-Memory Joins: Sort vs. Hash Revisited
http://www.vldb.org/pvldb/vol7/p85-balkesen.pdf

# FP

Generic Top-down Discrimination for Sorting and Partitioning in Linear Time
http://www.diku.dk/hjemmesider/ansatte/henglein/papers/henglein2011a.pdf

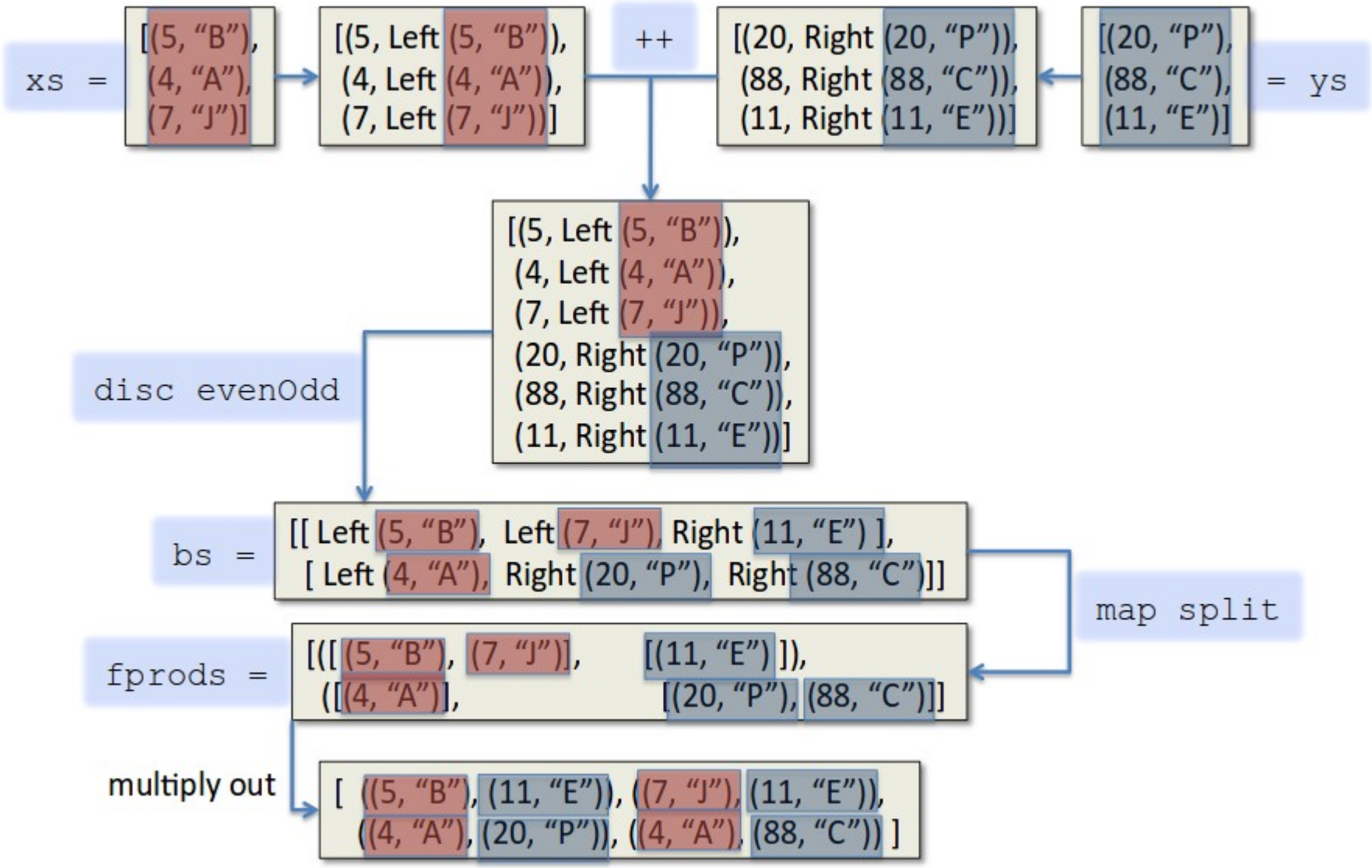
Generic Multiset Programming with Discrimination-based Joins and Symbolic Cartesian Products
http://www.diku.dk/hjemmesider/ansatte/henglein/papers/henglein2011c.pdf

Generic sorting and partitioning in linear time and fully abstractly
https://www.youtube.com/watch?v=sz9ZlZIRDAg

Discrimination is Wrong: Improving Productivity
https://www.youtube.com/watch?v=cB8DapKQz-I

```
data Order t where
     NatO     :: Int → Order Int
     TrivO    :: Order t
     SumL     :: Order t1 → Order t2 → Order (Either t1 t2)
     ProdL    :: Order t1 → Order t2 → Order (t1, t2)
     MapO     :: (t1 → t2) → Order t2 → Order t1
     ListL    :: Order t → Order [t]
     BagO     :: Order t → Order [t]
     SetO     :: Order t → Order [t]
     Inv      :: Order t → Order t
```

```
data Order t where
     NatO     :: Int → Order Int
     TrivO    :: Order t
     SumL     :: Order t1 → Order t2 → Order (Either t1 t2)
     ProdL    :: Order t1 → Order t2 → Order (t1, t2)
     MapO     :: (t1 → t2) → Order t2 → Order t1
     ListL    :: Order t → Order [t]
     BagO     :: Order t → Order [t]
     SetO     :: Order t → Order [t]
     Inv      :: Order t → Order t
```


xs =
[(5, "B"),
(4, "A"),
(7, "J")]
[(5, Left (5, "B")),
(4, Left (4, "A")),
(7, Left (7, "J"))]
++
[(20, Right (20, "P")),
(88, Right (88, "C")),
(11, Right (11, "E"))]
[(20, "P"),
(88, "C"),
(11, "E")]
= ys
[(5, Left (5, "B")),
(4, Left (4, "A")),
(7, Left (7, "J")),
(20, Right (20, "P")),
(88, Right (88, "C")),
(11, Right (11, "E"))]
disc evenOdd
bs =
[[ Left (5, "B"), Left (7, "J"), Right (11, "E") ],
[ Left (4, "A"), Right (20, "P"), Right (88, "C")]]
map split
fprods =
[([(5, "B"), (7, "J")], [(11, "E") ]),
([(4, "A")], [(20, "P"), (88, "C")])]
multiply out
[ ((5, "B"), (11, "E")), ((7, "J"), (11, "E")),
((4, "A"), (20, "P")), ((4, "A"), (88, "C")) ]

*'Cause we could be immortals, immortals*
*Just not for long, for long*
(Fall Out Boy, Immortals)

## Counting sort/pigeonhole sort

small domain

## Bucket sort/proxmap sort

known distribution

## Radixsort

Constant size elements

## Integer sorting

O(n sqrt(loglogK))

*"What's it going to be then, eh?"*

Anthony Burgess, A Clockwork Orange

*Hey young blood*
*doesn't it feel*
*like our time is running out?*

Fall Out Boy, The Phoenix


funkcionalne.cz