Quicksort Showdown - **quicksort_Magnetica_partitioning** vs **quicksort_Bentley_McIlroy_3way_partitioning**

A short review by sanmayce@sanmayce.com, 2021-Nov15

Legend:
FEW = 2,233,861,800 keys, of them distinct keys = 10
MANY = 2,482,300,900 keys, of them distinct keys = 2,847,531
~ALL = 2,009,333,753 keys, of them distinct keys = 1,912,608,132

Laptop Intel i5-7200U 3.1GHz max turbo, 36GB DDR4 2133MHz, running Fedora 33:

Performer/Keys	FEW distinct	MANY distinct	~ALL distinct
Compiler	GCC 10.2.1	GCC 10.2.1	GCC 10.2.1
Magnetica	**39** seconds user 112,449,514,935 instructions 24,092,559,780 branches 3,384,288,209 branch-misses 13,329,400,797 L1-dcache-loads 1,655,811,975 L1-dcache-load-misses 70,845,055 LLC-loads 62,316,430 LLC-load-misses	**222** seconds user 801,299,477,504 instructions 152,341,819,141 branches 19,420,019,786 branch-misses 105,227,165,554 L1-dcache-loads 9,758,775,155 L1-dcache-load-misses 331,205,631 LLC-loads 260,265,720 LLC-load-misses	**292** seconds user 1,077,947,210,852 instructions 202,031,197,397 branches 25,788,554,132 branch-misses 143,089,373,092 L1-dcache-loads 7,361,011,433 L1-dcache-load-misses 240,367,987 LLC-loads 189,560,929 LLC-load-misses

Laptop Intel i5-7200U 3.1GHz max turbo, 36GB DDR4 2133MHz, running Fedora 33:

Performer/Keys	FEW distinct	MANY distinct	~ALL distinct
Compiler	GCC 10.2.1	GCC 10.2.1	GCC 10.2.1
Bentley-McIlroy	**45** seconds user 141,773,758,594 instructions 29,021,682,740 branches 2,964,754,548 branch-misses 20,759,616,908 L1-dcache-loads 2,253,507,095 L1-dcache-load-misses 240,693,428 LLC-loads 119,509,033 LLC-load-misses	**226** seconds user 656,081,987,764 instructions 163,692,577,479 branches 18,927,109,180 branch-misses 85,146,879,545 L1-dcache-loads 8,623,541,343 L1-dcache-load-misses 286,332,303 LLC-loads 227,470,519 LLC-load-misses	**306** seconds user 896,389,731,533 instructions 226,893,780,871 branches 26,011,308,958 branch-misses 119,844,660,142 L1-dcache-loads 7,940,603,646 L1-dcache-load-misses 258,555,901 LLC-loads 217,482,356 LLC-load-misses

My threads with the Magnetica's source and binaries (Linux and Windows):

https://www.overclock.net/threads/benchmark-quicksort-says-sorting-2-billion-qwords.1794855/

https://www.qb64.org/forum/index.php?topic=3518.msg137645#msg137645

https://www.linuxquestions.org/questions/programming-9/qsort-vs-%27magnetica%27-quicksort-4175703333/#post6300860

```
// Written by Sanmayce, 2021-Nov-13
void Quicksort_QB64_v4(uint64_t QWORDS[], int64_t Left, int64_t Right) {
    int InsertionsortTHRESHOLD = 0;
    int64_t Indx, Jndx, PL, PR;
    int64_t Stack[99999];
    int64_t StackPtr = 0;
    uint64_t Pivot;
    StackPtr++; Stack[StackPtr] = Left;
    StackPtr++; Stack[StackPtr] = Right;
    do {
        Right = Stack[StackPtr];
        Left = Stack[StackPtr - 1];
        StackPtr = StackPtr - 2;
        do {
            // 'Magnetica' partitioning rev.2[
            Jndx = Right;
            PL = Left;
            PR = Left;
            swap (&QWORDS[(Left + Right)>>1], &QWORDS[PR]);
            Pivot = QWORDS[PR];
            for (;PR < Jndx;) {
                PR = PR + 1;
                if (Pivot > QWORDS[PR]) {
                    swap (&QWORDS[PL], &QWORDS[PR]);
                    PL = PL + 1;
                //} else if (Pivot == QWORDS[PR]) {
                } else if (Pivot < QWORDS[PR]) {
                    for (;Pivot < QWORDS[Jndx];) {
                        Jndx = Jndx - 1;
                    }
                    if (PR < Jndx) swap (&QWORDS[PR], &QWORDS[Jndx]);
                    Jndx = Jndx - 1;
                    PR = PR - 1;
                }
            }
            Jndx = PL - 1;
            Indx = PR + 1;
            if (Indx + InsertionsortTHRESHOLD < Right) {
                StackPtr = StackPtr + 2;
                Stack[StackPtr - 1] = Indx;
                Stack[StackPtr] = Right;
            }
            Right = Jndx;
        } while (Left < Right);
    } while (StackPtr != 0);
}
// GCC 10.2.1 -O3 -S -fverbose-asm
/*
.L209: <---------------------------------------------------------------\
                leaq    1(%r13), %rax
                subq    $2, %r15
                leaq    (%rbx,%r13,8), %r14
                movq    %rax, -112(%rsp)
                .p2align 4,,10
                .p2align 3
/-----> .L207:
                leaq    (%r12,%r13), %rax
                movq    (%r14), %rdx
                sarq    %rax
                leaq    (%rbx,%rax,8), %rax
                movq    (%rax), %r10
                movq    %rdx, (%rax)
                movq    %r10, (%r14)
                cmpq    %r13, %r12      # Left, Right
                jle     .L210
                movq    %r13, %rdx      # Left, PR
                movq    %r13, %r11      # Left, PL
                movq    %r12, %rcx      # Right, Jndx
                jmp     .L205
                .p2align 4,,10
                .p2align 3
 /---> .L215:
                leaq    (%rbx,%r11,8), %rax
                addq    $1, %r11
                movq    (%rax), %r9
                movq    %rdi, (%rax)
                leaq    2(%rdx), %rax
                movq    %rsi, %rdx
                movq    %r9, (%r8)
        .L200: <-----------------------------------------------\
                cmpq    %rcx, %rdx      # Jndx, PR
                jge     .L198
        .L205: <---------------------------------------------\
                leaq    1(%rdx), %rsi
                leaq    (%rbx,%rsi,8), %r8
                movq    (%r8), %rdi
                cmpq    %r10, %rdi      # Pivot=%r10, QWORDS[PR]=%rdi
 \-----------[ jb      .L215
 /-----------[ ja      .L201
                leaq    2(%rdx), %rax
                movq    %rsi, %rdx
                cmpq    %rcx, %rdx      # Jndx, PR
                jl      .L205 ]---------------------------------/
        .L198: <-----------------------------------------------+---\
                subq    $1, %r11
                cmpq    %rax, %r12
                jle     .L206 ]---------------------------------\
                movq    %rax, %xmm0
                movq    %r12, %xmm1
                punpcklqdq      %xmm1, %xmm0
                movups  %xmm0, -96(%rsp,%r15,8)
                addq    $2, %r15
        .L206: <------------------------------------------------/
                cmpq    %r13, %r11      # Left, Jndx
                jle     .L216
                movq    %r11, %r12      # Jndx, Right
\-+-----------[ jmp     .L207
                .p2align 4,,10
                .p2align 3
  \---> .L201:
                leaq    0(,%rcx,8), %rax
                leaq    (%rbx,%rax), %rbp
                movq    0(%rbp), %r9
                cmpq    %r9, %r10
                jnb     .L202 ]-------------\
                leaq    -8(%rbx,%rax), %rax
                .p2align 4,,10
                .p2align 3
        .L203: <----------------------\
                movq    %rax, %rbp
                movq    (%rax), %r9
                subq    $8, %rax
                subq    $1, %rcx
                cmpq    %r10, %r9
                ja      .L203 ]-------/
        .L202: <--------------------------/
                cmpq    %rcx, %rsi
                jge     .L204 ]-------\
                movq    %r9, (%r8)
                movq    %rdi, 0(%rbp)
        .L204: <----------------------/
                subq    $1, %rcx
                movq    %rsi, %rax
                jmp     .L200 ]-----------------------------------------/
                .p2align 4,,10
                .p2align 3
        .L210:
                movq    -112(%rsp), %rax
                movq    %r13, %r11      # Left, PL
                jmp     .L198 ]---------------------------------------------/
                .p2align 4,,10
                .p2align 3
        .L216:
                testq   %r15, %r15      # StackPtr
                je      .L197
                movq    -104(%rsp,%r15,8), %r12
                movq    -112(%rsp,%r15,8), %r13
                jmp     .L209 ]-------------------------------------------------/
        .L197:
*/
```

```
void quicksort_Bentley_McIlroy_3way_partitioning(uint64_t a[], int64_t l, int64_t r)
{
            int64_t i = l-1, j = r, p = l-1, q = r;
            int64_t k;
            uint64_t v = a[r];
            if (r <= l) return;
            for (;;)
            {
                        while (a[++i] < v) ;
                        while (v < a[--j]) if (j == l) break;
                        if (i >= j) break;
                        swap(&a[i], &a[j]);
                        if (a[i] == v) { p++; swap(&a[p], &a[i]); }
                        if (v == a[j]) { q--; swap(&a[j], &a[q]); }
            }
            swap(&a[i], &a[r]); j = i-1; i = i+1;
            for (k = l; k < p; k++, j--) swap(&a[k], &a[j]);
            for (k = r-1; k > q; k--, i++) swap(&a[i], &a[k]);
            quicksort_Bentley_McIlroy_3way_partitioning(a, l, j);
            quicksort_Bentley_McIlroy_3way_partitioning(a, i, r);
}
// GCC 10.2.1 -O3 -S
/*
            .seh_proc           quicksort_Bentley_McIlroy_3way_partitioning
quicksort_Bentley_McIlroy_3way_partitioning:
            pushq               %r15
            .seh_pushreg        %r15
            pushq               %r14
            .seh_pushreg        %r14
            pushq               %r13
            .seh_pushreg        %r13
            pushq               %r12
            .seh_pushreg        %r12
            pushq               %rbp
            .seh_pushreg        %rbp
            pushq               %rdi
            .seh_pushreg        %rdi
            pushq               %rsi
            .seh_pushreg        %rsi
            pushq               %rbx
            .seh_pushreg        %rbx
            subq                $88, %rsp
            .seh_stackalloc     88
            .seh_endprologue
            movq                %r8, %rax
            movq                %rcx, %r13
            leaq                -1(%rdx), %rbp
            movq                %rdx, %r11
            salq                $3, %rax
            movq                %r8, 176(%rsp)
            movq                %rax, 48(%rsp)
            leaq                (%rcx,%rax), %r12
            cmpq                %rdx, %r8
            jle                 .L151
            movq                %r8, %rsi
            leaq                -16(%rcx,%rax), %rdi
            addq                $8, %rax
            movq                (%r12), %r8
            subq                $1, %rsi
            movq                %rdi, 64(%rsp)
            movq                %rsi, 40(%rsp)
            leaq                (%rcx,%rsi,8), %rsi
            movq                %rsi, 72(%rsp)
            movq                %rax, 56(%rsp)
.L173:
            movq                176(%rsp), %r15
            leaq                1(%rbp), %rsi
            movq                %rbp, %r14
            leaq                0(%r13,%rsi,8), %rbx
            movq                %r15, %r10
            .p2align 4,,10
            .p2align 3
.L153:
            movq                (%rbx), %rdi
            cmpq                %r8, %rdi
            jb                  .L154
            leaq                -1(%r10), %rdx
            leaq                0(%r13,%rdx,8), %rax
            jmp                 .L155
            .p2align 4,,10
            .p2align 3
.L196:
            cmpq                %r11, %rdx
            je                  .L176
            subq                $1, %rdx
.L155:
            movq                (%rax), %rcx
            movq                %rax, %r9
            movq                %rdx, %r10
            subq                $8, %rax
            cmpq                %r8, %rcx
            ja                  .L196
.L176:
            cmpq                %rdx, %rsi
            jge                 .L157
            movq                %rcx, (%rbx)
            movq                %rdi, (%r9)
            cmpq                %r8, (%rbx)
            jne                 .L158
            addq                $1, %r14
            leaq                0(%r13,%r14,8), %rax
            movq                (%rax), %rdx
            movq                %r8, (%rax)
            movq                %rdx, (%rbx)
            movq                (%r9), %rdi
.L158:
            cmpq                %rdi, %r8
            jne                 .L154
            subq                $1, %r15
            leaq                0(%r13,%r15,8), %rax
            movq                (%rax), %rdx
            movq                %rdx, (%r9)
            movq                %r8, (%rax)
            .p2align 4,,10
            .p2align 3
.L154:
            movq                %rsi, %rbp
            addq                $8, %rbx
            addq                $1, %rsi
            jmp                 .L153
            .p2align 4,,10
            .p2align 3
.L157:
            movq                (%r12), %rax
            movq                %rax, (%rbx)
            leaq                2(%rbp), %rbx
            movq                %rdi, (%r12)
            cmpq                %r11, %r14
            jle                 .L174
            leaq                0(,%rbp,8), %rdx
            movq                %r11, %r10
            movq                %r14, %r8
            leaq                8(%rdx), %r9
            subq                %r14, %r10
            subq                %r11, %r8
            leaq                (%r9,%r10,8), %r10
            leaq                0(,%r14,8), %rcx
            leaq                0(,%r11,8), %rax
            cmpq                %r10, %rcx
            setle               %r10b
            cmpq                %rax, %r9
            setle               %r9b
            addq                %r13, %rax
            orb                 %r9b, %r10b
            je                  .L160
            leaq                -1(%r8), %r9
            cmpq                $1, %r9
            jbe                 .L160
            movq                %r8, %rcx
            leaq                -8(%r13,%rdx), %rdx
            shrq                %rcx
            salq                $4, %rcx
            addq                %rax, %rcx
            .p2align 4,,10
            .p2align 3
.L161:
            movdqu              (%rdx), %xmm1
            movdqu              (%rax), %xmm0
            addq                $16, %rax
            subq                $16, %rdx
            shufpd              $1, %xmm1, %xmm1
            shufpd              $1, %xmm0, %xmm0
            movups              %xmm1, -16(%rax)
            movups              %xmm0, 16(%rdx)
            cmpq                %rcx, %rax
            jne                 .L161
            movq                %r8, %rcx
            movq                %rbp, %rdx
            andq                $-2, %rcx
            subq                %rcx, %rdx
            leaq                (%r11,%rcx), %rax
            cmpq                %rcx, %r8
            je                  .L165
            leaq                0(%r13,%rdx,8), %rdx
            leaq                0(%r13,%rax,8), %rax
            movq                (%rax), %rcx
            movq                (%rdx), %r8
            movq                %r8, (%rax)
            movq                %rcx, (%rdx)
.L165:
            movq                %r11, %r8
            subq                %r14, %r8
            addq                %rbp, %r8
            cmpq                40(%rsp), %r15
            jge                 .L175
.L197:
            movq                176(%rsp), %rdx
            movq                56(%rsp), %rsi
            leaq                0(,%rbx,8), %rax
            subq                %r15, %rdx
            leaq                (%rbx,%rdx), %rbp
            movq                %rdx, %r9
            negq                %r9
            leaq                -1(%rbp), %rcx
            leaq                (%rsi,%r9,8), %r10
            leaq                0(,%rcx,8), %r9
            cmpq                %r9, %r10
            setge               %r10b
            cmpq                48(%rsp), %rax
            setge               %r9b
            orb                 %r9b, %r10b
            je                  .L167
            leaq                -2(%rdx), %r9
            cmpq                $1, %r9
            jbe                 .L167
            subq                $1, %rdx
            addq                %r13, %rax
            movq                64(%rsp), %r9
            movq                %rdx, %r10
            shrq                %r10
            salq                $4, %r10
            addq                %rax, %r10
            .p2align 4,,10
            .p2align 3
.L168:
            movdqu              (%r9), %xmm1
            movdqu              (%rax), %xmm0
            addq                $16, %rax
            subq                $16, %r9
            shufpd              $1, %xmm1, %xmm1
            shufpd              $1, %xmm0, %xmm0
            movups              %xmm1, -16(%rax)
            movups              %xmm0, 16(%r9)
            cmpq                %rax, %r10
            jne                 .L168
            movq                40(%rsp), %r9
            movq                %rdx, %rax
            andq                $-2, %rax
            addq                %rax, %rbx
            subq                %rax, %r9
            cmpq                %rdx, %rax
            je                  .L172
            leaq                0(%r13,%r9,8), %rax
            leaq                0(%r13,%rbx,8), %rdx
            movq                (%rdx), %r9
            movq                (%rax), %r10
            movq                %r10, (%rdx)
            movq                %r9, (%rax)
.L172:
            movq                %rcx, %rbx
            subq                $2, %rbp
.L166:
            movq                %r11, %rdx
            movq                %r13, %rcx
            call
            quicksort_Bentley_McIlroy_3way_partitioning
            cmpq                %rbx, 176(%rsp)
            jle                 .L151
            movq                (%r12), %r8
            movq                %rbx, %r11
            jmp                 .L173
.L151:
            addq                $88, %rsp
            popq                %rbx
            popq                %rsi
            popq                %rdi
            popq                %rbp
            popq                %r12
            popq                %r13
            popq                %r14
            popq                %r15
            ret
.L167:
            movq                72(%rsp), %rdx
            addq                %r13, %rax
            leaq                0(%r13,%r15,8), %r10
            .p2align 4,,10
            .p2align 3
.L171:
            movq                (%rax), %r9
            movq                (%rdx), %rbx
            subq                $8, %rdx
            addq                $8, %rax
            movq                %rbx, -8(%rax)
            movq                %r9, 8(%rdx)
            cmpq                %r10, %rdx
            jne                 .L171
            jmp                 .L172
.L160:
            addq                %r13, %rdx
            addq                %r13, %rcx
            .p2align 4,,10
            .p2align 3
.L164:
            movq                (%rdx), %r9
            movq                (%rax), %r8
            addq                $8, %rax
            subq                $8, %rdx
            movq                %r9, -8(%rax)
            movq                %r8, 8(%rdx)
            cmpq                %rcx, %rax
            jne                 .L164
            jmp                 .L165
.L174:
            movq                %rbp, %r8
            cmpq                40(%rsp), %r15
            jl                  .L197
.L175:
            movq                %rsi, %rbp
            jmp                 .L166
            .seh_endproc
*/
```

Quicksort Showdown - quicksort_Magnetica_partitioning vs quicksort_Bentley_McIlroy_3way_partitioning
A short review by sanmayce@sanmayce.com, 2021-Nov15

Bottomline first, ON LINUX, Who can tell why Quicksort 'Magnetica' is always faster than Intel's qsort, GLIBC's qsort, Quicksort 'Classica', Quicksort 'Bentley_McIlroy_3way_partitioning'?! What causes the reversal, ON WINDOWS?!

Files being sorted:
```
10/30/2021  15:29       178,708,944 22338618_QWORDS.bin
03/26/2014  08:14        24,823,016 mobythesaurus.txt
11/12/2021  14:52     2,009,333,760 Fedora-Workstation-Live-x86_64-35-1.2.iso
```
Files used to do the sort:
```
11/13/2021  03:40       268,465,664 QS_bench_r3
11/13/2021  04:42            37,943 QS_bench_r3.c
11/14/2021  13:56       268,511,531 QS_bench_r3_GCC10.2.1.exe
11/13/2021  04:43       268,534,272 QS_bench_r3_ICL15.0.exe
```

Magnetica, take heed!
ON LINUX: 39/222/292
ON WINDOWS: 38/221/288

Bentley-McIlroy, take heed!
ON LINUX: 45/226/306
ON WINDOWS: 41/213/287

The bencmarking below, and the Assembler on the right show ICL outperforms GCC in this etude, GCC is supergood, though.

```
            // 'Magnetica' partitioning rev.2[
            Jndx = Right;
            PL = Left;
            PR = Left;
            swap (&QWORDS[(Left + Right)>>1], &QWORDS[PR]);
            Pivot = QWORDS[PR];
            for (;PR < Jndx;) {
                PR = PR + 1;
                if (Pivot > QWORDS[PR]) {
                    swap (&QWORDS[PL], &QWORDS[PR]);
                    PL = PL + 1;
                //} else if (Pivot == QWORDS[PR]) {
                } else if (Pivot < QWORDS[PR]) {
                    for (;Pivot < QWORDS[Jndx];) {
                        Jndx = Jndx - 1;
                    }
                    if (PR < Jndx) swap (&QWORDS[PR], &QWORDS[Jndx]);
                    Jndx = Jndx - 1;
                    PR = PR - 1;
                }
            }
            Jndx = PL - 1;
            Indx = PR + 1;
            // 'Magnetica' partitioning rev.2]
/*
; mark_description "Intel(R) C++ Compiler XE for applications running
on Intel(R) 64, Version 15.0.0.108 Build 20140726";
; ae-64+2=76 bytes i.e. (94-76=18 less), 5/1
conditional/unconditional jumps i.e. 2-1=1 less unconditional jump
; 'Magnetica' partitioning, mainloop rev.2[
.B4.5::
  00064 48 ff c5        inc rbp
  00067 48 8b 14 e9     mov rdx, QWORD PTR [rcx+rbp*8]
  0006b 4c 3b ea        cmp r13, rdx
  0006e 77 2c           ja .B4.14
.B4.6::
  00070 73 39           jae .B4.15
.B4.7::
  00072 4e 8b 3c d9     mov r15, QWORD PTR [rcx+r11*8]
  00076 4d 3b ef        cmp r13, r15
  00079 73 0c           jae .B4.11
.B4.9::
  0007b 49 ff cb        dec r11
  0007e 4e 8b 3c d9     mov r15, QWORD PTR [rcx+r11*8]
  00082 4d 3b ef        cmp r13, r15
  00085 72 f4           jb .B4.9
.B4.11::
  00087 49 3b eb        cmp rbp, r11
  0008a 7d 08           jge .B4.13
.B4.12::
  0008c 4c 89 3c e9     mov QWORD PTR [rcx+rbp*8], r15
  00090 4a 89 14 d9     mov QWORD PTR [rcx+r11*8], rdx
.B4.13::
  00094 49 ff cb        dec r11
  00097 48 ff cd        dec rbp
  0009a eb 0f           jmp .B4.15
.B4.14::
  0009c 4e 8b 3c f1     mov r15, QWORD PTR [rcx+r14*8]
  000a0 4a 89 14 f1     mov QWORD PTR [rcx+r14*8], rdx
  000a4 49 ff c6        inc r14
  000a7 4c 89 3c e9     mov QWORD PTR [rcx+rbp*8], r15
.B4.15::
  000ab 49 3b eb        cmp rbp, r11
  000ae 7c b4           jl .B4.5
; 'Magnetica' partitioning, mainloop rev.2]
*/
```

Laptop Intel i5-7200U 3.1GHz max turbo, 36GB DDR4 2133MHz, running Fedora 33:

Performer/Keys	FEW distinct	MANY distinct	~ALL distinct
Compiler	GCC 10.2.1	GCC 10.2.1	GCC 10.2.1
qsort	**317** seconds user	**508** seconds user	**504** seconds user
	3,279,551,895,331 instructions	**2,937,315,064,269** instructions	**2,271,179,443,512** instructions
	650,482,999,545 branches	**656,656,940,988** branches	**536,952,307,468** branches
	5,582,924,930 branch-misses	**25,008,235,993** branch-misses	**28,368,595,067** branch-misses
	734,186,209,415 L1-dcache-loads	**675,474,160,813** L1-dcache-loads	**520,144,933,578** L1-dcache-loads
	6,753,617,333 L1-dcache-load-misses	**8,374,868,618** L1-dcache-load-misses	**6,723,006,421** L1-dcache-load-misses
	404,414,575 LLC-loads	**244,454,547** LLC-loads	**193,261,861** LLC-loads
	111,254,842 LLC-load-misses	**132,791,156** LLC-load-misses	**102,759,148** LLC-load-misses
Magnetica	**39** seconds user	**222** seconds user	**292** seconds user
	112,449,514,935 instructions	**801,299,477,504** instructions	**1,077,947,210,852** instructions
	24,092,559,780 branches	**152,341,819,141** branches	**202,031,197,397** branches
	3,384,288,209 branch-misses	**19,420,019,786** branch-misses	**25,788,554,132** branch-misses
	13,329,400,797 L1-dcache-loads	**105,227,165,554** L1-dcache-loads	**143,089,373,092** L1-dcache-loads
	1,655,811,975 L1-dcache-load-misses	**9,758,775,155** L1-dcache-load-misses	**7,361,011,433** L1-dcache-load-misses
	70,845,055 LLC-loads	**331,205,631** LLC-loads	**240,367,987** LLC-loads
	62,316,430 LLC-load-misses	**260,265,720** LLC-load-misses	**189,560,929** LLC-load-misses
Bentley-McIlroy	**45** seconds user	**226** seconds user	**306** seconds user
	141,773,758,594 instructions	**656,081,987,764** instructions	**896,389,731,553** instructions
	29,021,682,740 branches	**165,692,577,479** branches	**226,893,780,871** branches
	2,964,754,548 branch-misses	**18,927,109,180** branch-misses	**26,011,308,958** branch-misses
	20,759,616,908 L1-dcache-loads	**85,146,879,545** L1-dcache-loads	**119,844,660,142** L1-dcache-loads
	2,253,507,095 L1-dcache-load-misses	**8,623,541,343** L1-dcache-load-misses	**7,940,603,646** L1-dcache-load-misses
	240,693,428 LLC-loads	**286,352,303** LLC-loads	**258,555,901** LLC-loads
	119,509,033 LLC-load-misses	**227,490,519** LLC-load-misses	**217,482,356** LLC-load-misses

Legend:
FEW = 2,233,861,800 keys, of them distinct = **10**; MANY = 2,482,300,900 keys, of them distinct = **2,847,531**; ~ALL = 2,009,333,753 keys, of them distinct = **1,912,608,132**

Laptop Intel i5-7200U 3.1GHz max turbo, 36GB DDR4 2133MHz, running Windows 10:

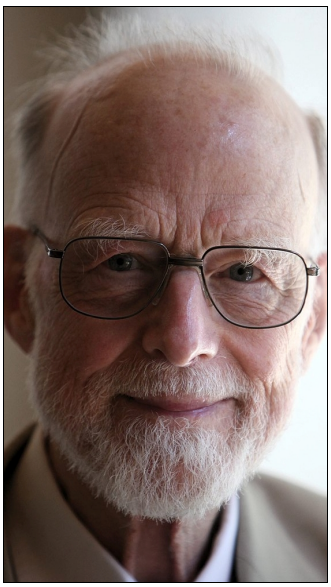
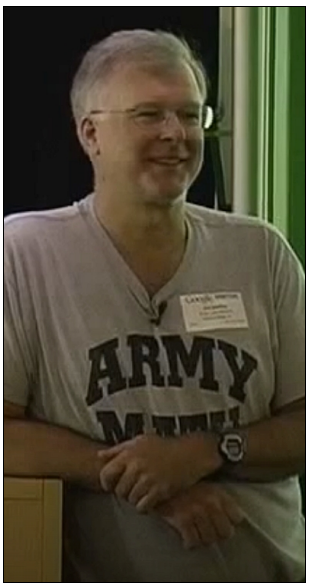
Performer/Keys	FEW distinct		MANY distinct		~ALL distinct	
Compiler	Intel v15.0	GCC 10.2.1	Intel v15.0	GCC 10.2.1	Intel v15.0	GCC 10.2.1
qsort	**66** seconds	61 seconds	**378** seconds	352 seconds	**484** seconds	458 seconds
Magnetica	**37** seconds	38 seconds	**218** seconds	221 seconds	**289** seconds	288 seconds
Bentley-McIlroy	**43** seconds	41 seconds	**209** seconds	213 seconds	**283** seconds	287 seconds

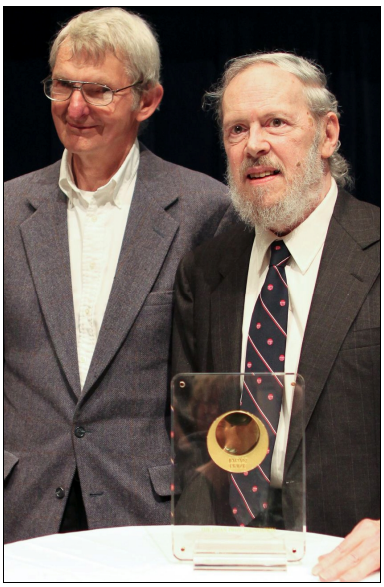
Legend (The time is the "User Time"):
FEW = 2,233,861,800 keys, of them distinct = **10**; MANY = 2,482,300,900 keys, of them distinct = **2,847,531**; ~ALL = 2,009,333,753 keys, of them distinct = **1,912,608,132**

Laptop AMD Ryzen 7 Renoir 4800H 4.3GHz max turbo, 64GB DDR4 3200MHz, running Windows 10:

Performer/Keys	FEW distinct		MANY distinct		~ALL distinct	
Compiler	Intel v15.0	GCC 10.2.1	Intel v15.0	GCC 10.2.1	Intel v15.0	GCC 10.2.1
qsort	**48** seconds	50 seconds	**273** seconds	285 seconds	**357** seconds	368 seconds
Magnetica	**26** seconds	26 seconds	**152** seconds	159 seconds	**206** seconds	214 seconds
Bentley-McIlroy	**28** seconds	28 seconds	**145** seconds	149 seconds	**197** seconds	203 seconds

Legend (The time is the "User Time"):
FEW = 2,233,861,800 keys, of them distinct = **10**; MANY = 2,482,300,900 keys, of them distinct = **2,847,531**; ~ALL = 2,009,333,753 keys, of them distinct = **1,912,608,132**




The way so far: Quicksort (1962) by **Sir Antony Hoare**, Bentley-McIlroy 3-way partitioning (1992) by **Dr. Jon Bentley** (Bell Laboratories) and **Dr. Douglas McIlroy** (Bell Laboratories) to the Dr. Dennis Ritchie's right, Magnetica (2021) by **Sanmayce**