Detaljert eksempel av Quicksort (uten Median-of-Three og Insertion sort)

```
Skal sorteres:
  a[]     [ '-1', 'E', 'A', 'S', 'Y', 'Q', 'U', 'E' ]
  Indeks: [   0 ,  1 ,  2 ,  3 ,  4 ,  5 ,  6 ,  7  ]
  Verdi:  [   0 ,  5 ,  1 ,  19,  25,  17,  21,  5  ]
  N = 7


// 1 - quicksort(a, l, r);
// l = 1, r = 7.  7 > 1 ? Ja, fortsetter
v = a[r];                 // v = a[7] => v = 'E'
i = l-1;                  // i = 1-1 => i = 0
j = r;                    // j = 7

// Uendelig loop 1.
for (;;) {
  while(a[++i] < v);      // Looper til i + 1 ikke er mindre enn 'v'
                          // a[1] < v? => 'E' < 'E'? Nei. i = 1
  while(a[--j] > v);      // Looper til j - 1 ikke er større enn 'v'
                          // a[6] > v? => 'U' > 'E'? Ja
                          // a[5] > v? => 'Q' > 'E'? Ja
                          // a[4] > v? => 'Y' > 'E'? Ja
                          // a[3] > v? => 'S' > 'E'? Ja
                          // a[2] > v? => 'A' > 'E'? Nei. j = 2
  if (i >= j) break;      // 1 >= 2? Nei
  swap(a, i, j);          // Bytter om a[1] med a[2]
                          // 'E' og 'A' bytter plass
}
// Sub-array etter første loop:
//  a[]     [ '-1', '*A', '*E', 'S', 'Y', 'Q', 'U', 'E' ]
//  Indeks: [   0 ,   1 ,   2 ,  3 ,  4 ,  5 ,  6 ,  7  ]

// Uendelig loop 2.
for (;;) {
  while(a[++i] < v);      // i var 1
                          // a[2] < v? => 'E' < 'E'? Nei. i = 2
  while(a[--j] > v);      // j var 2
                          // a[1] > v? => 'A' > 'E'? Nei. j = 1
  if (i >= j) break;      // 2 >= 1? Ja, breaker
}
swap(a, i, r);            // Bytter om a[2] med a[7]
                          // 'E' og 'E' bytter plass
// Sub-array etter andre loop:
//  a[]     [ '-1', 'A', '*E', 'S', 'Y', 'Q', 'U', '*E' ]
//  Indeks: [   0 ,  1 ,   2 ,  3 ,  4 ,  5 ,  6 ,   7  ]


// 1-1 quicksort(a, l, i-1);
// l = 1, r = i-1 => r = 2-1 => r = 1. 1 > 1? Nei, avslutter


// 1-2 quicksort(a, i+1, r);
// l = i+1 => l = 2+1 => l = 3, r = 7. 7 > 3? Ja, fortsetter

// Sub-array som skal sorteres:
//  a[]     [ 'S', 'Y', 'Q', 'U', 'E' ]
//  Indeks: [  3 ,  4 ,  5 ,  6 ,  7  ]

v = a[r];                 // v = a[7] => v = 'E'
i = l-1;                  // i = 3-1 => i = 2
j = r;                    // j = 7

// Uendelig loop 1.
for (;;){
  while(a[++i] < v);      // i var 2
```

```
                        // a[3] < 'E' => 'S' < 'E'? Nei. i = 3
  while(a[--j] > v);    // j var 7
                        // a[6] > 'E' => 'U' > 'E'? Ja
                        // a[5] > 'E' => 'Q' > 'E'? Ja
                        // a[4] > 'E' => 'Y' > 'E'? Ja
                        // a[3] > 'E' => 'S' > 'E'? Ja
                        // a[2] > 'E' => 'E' > 'E'? Nei. j = 2
  if (i >= j) break;    // 3 >= 2? Ja, breaker
}
swap(a, i, r);          // Bytter om a[3] med a[7]
                        // 'S' og 'E' bytter plass
// Sub-array etter første loop:
//  a[]     [ '*E', 'Y', 'Q', 'U', '*S' ]
//  Indeks: [   3 ,  4 ,  5 ,  6 ,   7  ]


// 1-2-1 quicksort(a, l, i-1);
// l = 3, r = 3-1 => r = 2. 2 > 3? Nei, avslutter


// 1-2-2 quicksort(a, i+1, r);
// l = 3+1 => l = 4, r = 7. 7 > 4? Ja, fortsetter

// Sub-array som skal sorteres:
//  a[]     [ 'Y', 'Q', 'U', 'S' ]
//  Indeks: [  4 ,  5 ,  6 ,  7  ]

v = a[r];               // v = a[7] => v = 'S'
i = l-1;                // i = 4-1 => l = 3
j = r;                  // j = 7

// Uendelig loop 1.
for (;;) {
  while(a[++i] < v);    // i var 3
                        // a[4] < 'S' => 'Y' < 'S'? Nei. i = 4
  while(a[--j] > v);    // j var 7
                        // a[6] > 'S' => 'U' > 'S'? Ja
                        // a[5] > 'S' => 'Q' > 'S'? Nei. j = 5
  if (i >= j) break;    // 4 >= 5? Nei
  swap(a, i, j);        // Bytter om a[4] med a[5]
                        // 'Y' og 'Q' bytter plass
}
// Sub-array etter første loop:
//  a[]     [ '*Q', '*Y', 'U', 'S' ]
//  Indeks: [   4 ,   5 ,  6 ,  7  ]

// Uendelig loop 2.
for (;;) {
  while(a[++i] < v);    // i var 4
                        // a[5] < 'S' => 'U' < 'S'? Nei. i = 5
  while(a[--j] > v);    // j var 5
                        // a[4] > 'S' => 'Q' > 'S'? Nei. j = 4
  if (i >= j) break;    // 5 >= 4? Ja, breaker
}
swap(a, i, r);          // Bytter om a[5] med a[7]
                        // 'Y' og 'S' bytter plass
// Sub-array etter andre loop:
//  a[]     [ 'Q', '*S', 'U', '*Y' ]
//  Indeks: [  4 ,   5 ,  6 ,   7  ]


// 1-2-2-1 quicksort(a, l, i-1);
// l = 4, r = 5-1 => r = 4. 4 > 4? Nei, avslutter


// 1-2-2-2 quicksort(a, i+1, r);
// l = 5+1 => l = 6, r = 7. 7 > 6? Ja, fortsetter
```

```
// Sub-array som skal sorteres:
//  a[]     [ 'U', 'Y' ]
//  Indeks: [  6 ,  7  ]

v = a[r];                 // v = a[7] => v = 'Y'
i = l-1;                  // i = 6-1 => i = 5
j = r;                    // j = 7

// Uendelig loop 1.
for (;;) {
  while(a[++i] < v);      // i var 5
                          // a[6] < 'Y' => 'U' < 'Y'? Ja
                          // a[7] < 'Y' => 'Y' < 'Y'? Nei. i = 7
  while(a[--j] > v);      // j var 7
                          // a[6] > 'Y' => 'U' > 'Y'? Nei. j = 6
  if (i >= j) break;      // 7 >= 6? Ja, breaker
}
swap(a, i, r);            // Bytter om a[7] med a[7]
                          // 'Y' og 'Y' bytter plass
// Sub-array etter første loop:
//  a[]     [ 'U', '*Y' ]
//  Indeks: [  6 ,   7  ]


// 1-2-2-2-1 quicksort(a, l, i-1);
// l = 6, r = 7-1 => r = 6. 6 > 6? Nei, avslutter

// 1-2-2-2-2 quicksort(a, i+1, r);
// l = 7 + 1 => l = 8, r = 7. 7 > 8? Nei, avslutter



// 1 - partisjon (1):
//  a[]     [ '-1', 'A', 'E', 'S', 'Y', 'Q', 'U', 'E' ]
//  Indeks: [   0 ,  1 ,  2 ,  3 ,  4 ,  5 ,  6 ,  7  ]

// 2 - partisjon (1):
//  a[]     [ '-1', 'A', 'E', 'S', 'Y', 'Q', 'U', 'E' ]
//  Indeks: [   0 ,  1 ,  2 ,  3 ,  4 ,  5 ,  6 ,  7  ]

// 3 - partisjon (1-2):
//  a[]                    [ 'E', 'Y', 'Q', 'U', 'S' ]
//  Indeks:                [  3 ,  4 ,  5 ,  6 ,  7  ]

// 4 - partisjon (1-2-2):
//  a[]                          [ 'Q', 'Y', 'U', 'S' ]
//  Indeks:                      [  4 ,  5 ,  6 ,  7  ]

// 5 - partisjon (1-2-2):
//  a[]                          [ 'Q', 'S', 'U', 'Y' ]
//  Indeks:                      [  4 ,  5 ,  6 ,  7  ]

// 6 - partisjon (1-2-2-2):
//  a[]                                    [ 'U', 'Y' ]
//  Indeks:                                [  6 ,  7  ]

// Dette blir sammenslått:
//  a[]     [ '-1', 'A', 'E', 'E', 'Q', 'S', 'U', 'Y' ]
//  Indeks: [   0 ,  1 ,  2 ,  3 ,  4 ,  5 ,  6 ,  7  ]
```