Detaljert eksempel av Shellsort

```
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


// Bestemmer hva "gap(h)" skal være.
for(h = 1; h <= N/9; 3*h+1);       // N/9 = 0,7 => runnet av til 0. vil ikke kjøre.
                                   // I dette eksempelet velger vi at h = 3.


// Gap loop 1. (h = 3)
for (; h > 0; h/=3) {              // Etter hver loop vil gap bli mindre.

  // Outer loop 1.
  for (i = h+1; i <= N; i+=1) {    // i = 3+1, 4 <= 7? Ja
    v = a[i];                      // v = a[4] => v = 'Y'
    j = i;                         // j = 4

    // Inner loop 1.
    while(j > h && a[j-h] > v) {   // 4 > 3 og a[4-3] > v => 'E' > 'Y' (5 > 25)? Nei
      // While loop kjøre ikke
    }
    a[j] = v;                      // a[4] = 'Y', ingen endring
  }

  // Outer loop 2. (i var 4)
  for (i = h+1; i <=N; i+=1) {     // i ble inkrementert til 5 i forrige loop. 5<=7? Ja
    v = a[i];                      // v = a[5] => v = 'Q'
    j = i;                         // j = 5

    // Inner loop 1.
    while(j > h && a[j-h] > v) {   // 5 > 3 og a[5-3] > v => 'A' > 'Q' (1 > 17)? Nei
      // While loop kjører ikke
    }
    a[j] = v;                      // a[5] = 'Q', ingen endring
  }

  // Outer loop 3. (i var 5)
  for (i = h+1; i <=N; i+=1) {     // i ble inkrementert til 6 i forrige loop. 6<=7? Ja
    v = a[i];                      // v = a[6] => v = 'U'
    j = i;                         // j = 6

    // Inner loop 1.
    while(j > h && a[j-h] > v) {   // 6 > 3 && a[6-3] > v => 'S' > 'U' (19 >21)? Nei
      // While loop kjører ikke
    }
    a[j] = v;                      // a[6] = 'U', ingen endring
  }

  // Outer loop 4. (i var 6)
  for (i = h+1; i <=N; i+=1) {     // i ble inkrementert til 7 i forrige loop. 7<=7? Ja
    v = a[i];                      // v = a[7] => v = 'E'
    j = i;                         // j = 7

    // Inner loop 1.
    while(j > h && a[j-h] > v) {   // 7 > 3 && a[7-3] > v +> 'Y' > 'E' (25 > 5)? Ja
      a[j] = a[j-h];               // a[7] = a[7-3] => a[7] = 'Y'
      j -= h;                      // j = j-h => j = 7-3 => j = 4
    }
    // Inner loop 2 sjekkes.
    // 7 > 3 && a[4-3] > v => 'E' > 'E' (5 > 5)? Nei (stable)
    a[j] = v;                      // a[4] = 'E'
```

```
  }


    // Etter Inner loop 1 og Outer loop 4 har kjørt, har vi følgende:
    // a[7] = 'Y' og a[4] = 'E'
    // Arrayen er nå: [ '-1', 'E', 'A', 'S', 'E', 'Q', 'U', 'Y' ]


  // Outer loop 5. (i var 7)
  for (i = h+1; i <=N; i+=1) {    // i ble inkrementert til 8 i forrige loop. 8<=7? Nei
    // Outer loopen stopper, og vi faller tilbake til "gap" loopen.
  }

// Gap loop 2. (h = 3)
for (; h > 0; h/=3) {             // h = 3/3 => h = 1.
    // Siden h er nå 1, så betyr det at gap er 1.
    // Når gap er 1, så vil vanlig Insertion Sort kjøre på resten av arrayen.
    // Dette finnes det et eget eksempel på.
```