Tampereen yliopisto

Järjestysalgoritmien tehokkuuden vertailu

COMP.CS.300 Tietorakenteet ja algoritmit 1

Matti Rintala (matti.rintala@tuni.fi)

Mitä voidaan vertailla?

- Asymptoottinen tehokkuus
  - Huonoin tapaus (tai ainakin O)
  - Paras tapaus (tai ainakin Ω)
  - Keskimääräinen tapaus (mitä tarkoittaa?)

Tampereen yliopisto

Mitä voidaan vertailla?

- Todellinen tehokkuus
  - Riippuu toteutuksesta!
  - Usein voidaan vain valistuneesta arvailla
  - ...tai testata jollain toteutuksella

Mitä voidaan vertailla?

- Asymptoottinen tehokkuus
  - Huonoin tapaus (tai ainakin O)
  - Paras tapaus (tai ainakin Ω)
  - Keskimääräinen tapaus (mitä tarkoittaa?)
- Todellinen tehokkuus
  - Riippuu toteutuksesta!
  - Usein voidaan vain valistuneesta arvailla
  - ...tai testata jollain toteutuksella

Insertion-Sort(*A*)

```
1 for next_elem := 2 to A.length do
2       key := A[next_elem]
3       place := next_elem − 1
4       while place > 0 and A[place] > key do
5             A[place + 1] := A[place]
6             place := place − 1
7       A[place + 1] := key
```

Quicksort(*A*, *left*, *right*)

```
1 if left < right then                    (triviaalitapaukselle ei tehdä mitään)
2     pivot := Partition(A, left, right) (jaetaan pieniin ja suuriin, pivot jakokohta)
3     Quicksort(A, left, pivot−1 )       (järjestetään jakoalkiota pienemmät)
4     Quicksort(A, pivot+1, right)       (järjestetään jakoalkiota suuremmat)
```

Partition(*A*, *left*, *right*)

```
1 pivot := A[right]          (otetaan pivotiksi viimeinen alkio)
2 cut := left − 1            (merkitään cut:lla pienten puolen loppua)
3 for i := left to right−1 do  (käydään läpi toiseksi viimeiseen alkioon asti)
4     if A[i] ≤ pivot        (jos A[i] kuuluu pienten puolelle...)
5         cut := cut + 1     (... kasvatetaan pienten puolta...)
6         A[cut] ⇄ A[i]      (... ja siirretään A[i] sinne)
7 A[ cut+1 ] ⇄ A[ right ]    (sijoitetaan pivot pienten ja isojen puolten väliin)
8 return cut+1               (palautetaan pivot-alkion uusi sijainti)
```

Mergesort(*A*, *left*, *right*)

```
1 if left < right then                (triviaalitapaukselle ei tehdä mitään)
2     mid := ⌊(left + right) / 2⌋     (lasketaan puoliväli)
3     Mergesort(A, left, mid )        (järjestetään vasen puoli)
4     Mergesort(A, mid+1, right)      (järjestetään oikea puoli)
5     Merge(A, left, mid, right)      (lomitetaan järjestetyt puolikkaat yhteen)
```

Merge(*A*, *left*, *mid*, *right*)

```
1  for i := left to right do
2      Tmp[i] := A[i]
3  out := left
4  in_l := left; in_r := mid + 1
5  while in_l ≤ mid and in_r ≤ right do
6      if Tmp[in_l] ≤ Tmp[in_r] then
7          A[out] := Tmp[in_l]
8          in_l := in_l + 1
9      else
10         A[out] := Tmp[in_r]
11         in_r := in_r + 1
12     out := out + 1
13 if in_l > mid then
14     in_rest := in_r
15 else
16     in_rest := in_l
17 for i := 0 to right-out do
18     A[out + i] := Tmp[in_rest + i]
```