3.3 Suunnitteluperiaate: Hajota ja hallitse

Suunnitteluperiaate *hajota ja hallitse* on todennäköisesti periaatteista kuuluisin.

Se toimii useiden tunnettujen tehokkaiden algoritmien periaatteena

Perusidea:

- ongelma jaetaan alkuperäisen kaltaisiksi, mutta pienemmiksi osaongelmiksi.
- pienet osaongelmat ratkaistaan suoraviivaisesti
- suuremmat osaongelmat jaetaan edelleen pienempiin osiin
- lopuksi osaongelmien ratkaisut kootaan alkuperäisen ongelman ratkaisuksi

Hajota ja hallitse on usein rekursiivinen rakenteeltaan: algoritmi kutsuu itseään osaongelmille

Pienten osaongelmien ratkaisemiseksi voidaan myös hyödyntää toista algoritmia

3.4 QUICKSORT

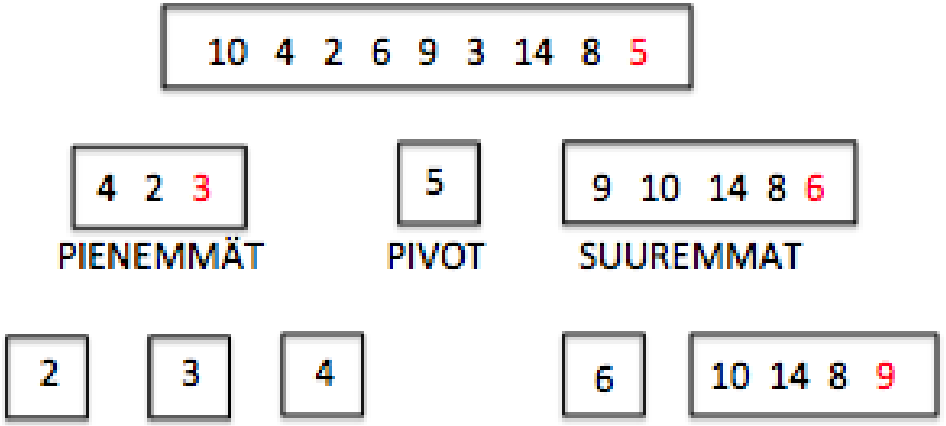
Ongelman jakaminen pienemmiksi osaongelmiksi

- Valitaan jokin taulukon alkioista jakoalkioksi eli *pivot-alkioksi*.
- Muutetaan taulukon alkioiden järjestystä siten, että kaikki jakoalkiota pienemmät tai yhtäsuuret alkiot ovat taulukossa ennen jakoalkiota ja suuremmat alkiot sijaitsevat jakoalkion jälkeen.
- Jatketaan alku ja loppuosien jakamista pienemmiksi, kunnes ollaan päästy 0:n tai 1:n kokoisiin osataulukoihin.

QUICKSORT-algoritmi

QUICKSORT($A, left, right$)

```
1   if left < right then                  (triviaalitapaukselle ei tehdä mitään)
2       p := PARTITION( A, left, right )  (muuten jaetaan alkuosaan ja loppuosaan)
3       QUICKSORT( A, left, p − 1 )       (järjestetään jakoalkiota pienemmät)
4       QUICKSORT( A, p + 1, right )      (järjestetään jakoalkiota suuremmat)
```



Kuva 3: Jako pienempiin ja suurempiin

Pienet osaongelmat:

- 0:n tai 1:n kokoiset osataulukot ovat valmiiksi järjestyksessä.

Järjestyksessä olevien osataulukoiden yhdistäminen:

- Kun alkuosa ja loppuosa on järjestetty on koko (osa)taulukko automaattisesti järjestyksessä.
  - kaikki alkuosan alkiothan ovat loppuosan alkioita pienempiä, kuten pitääkin

Ositus- eli *partitiointialgoritmi* jakaa taulukon paikallaan vaaditulla tavalla.

PARTITION($A, left, right$)

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

3.5 MERGESORT

Erinomainen esimerkki hajota ja hallitse -periaatteesta on MERGE-SORT järjestämisalgoritmi:

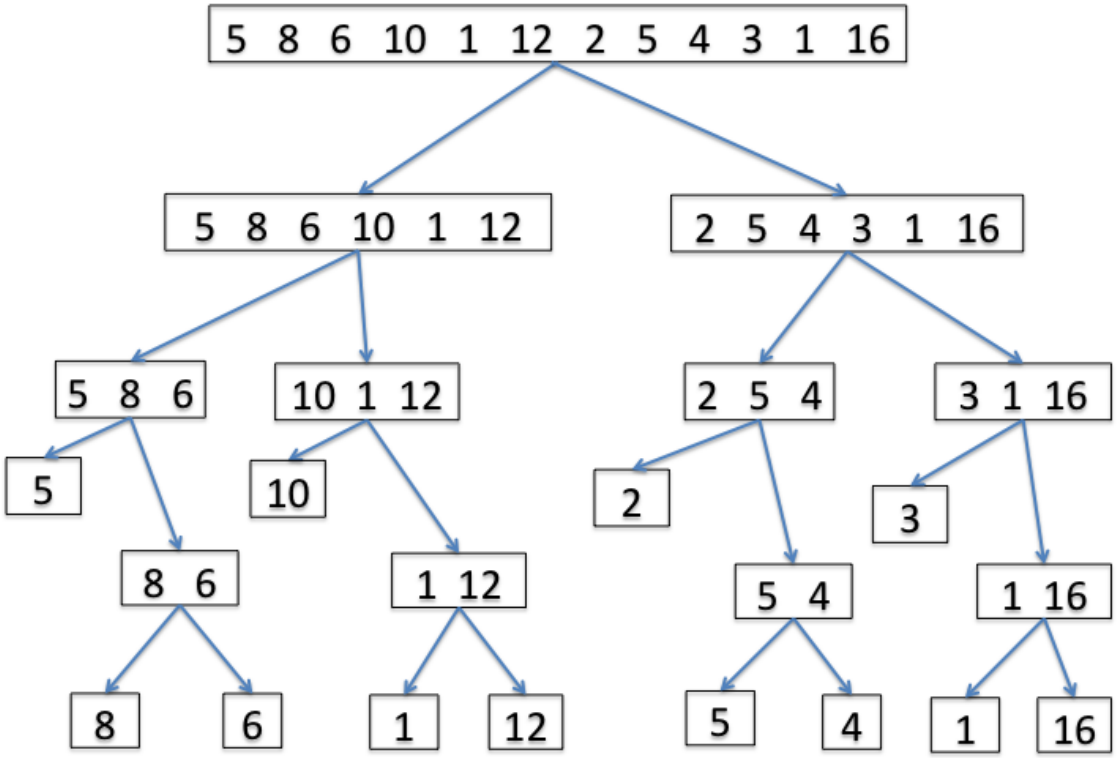
1. Taulukko jaetaan kahteen osaan $A[1..\lfloor n/2 \rfloor]$ ja $A[\lfloor n/2 \rfloor + 1..n]$.
2. Järjestetään puolikkaat rekursiivisesti
3. Limitetään järjestetyt puolikkaat järjestetyksi taulukoksi

- MERGE-SORT -ALGORITMI

MERGE-SORT($A, left, right$)

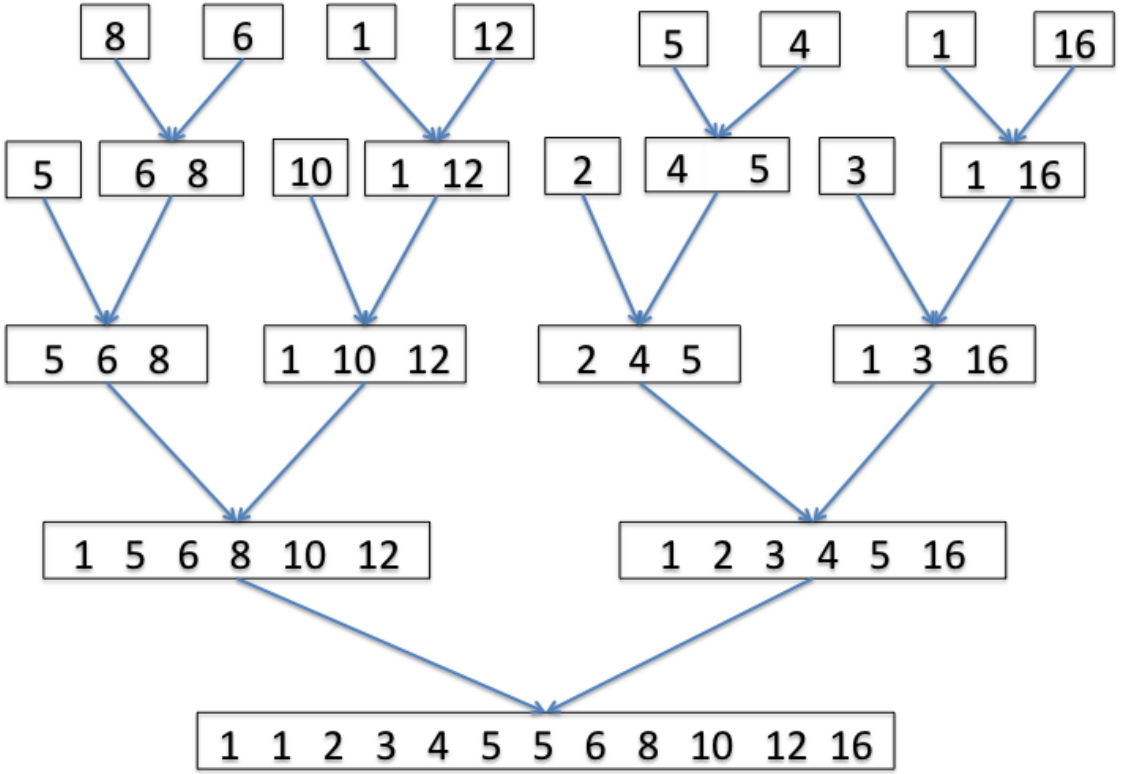
```
1   if left < right then                    (jos taulukossa on alkioita...)
2       mid := ⌊( left + right )/2⌋         (... jaetaan se kahtia)
3       MERGE-SORT( A, left, mid )          (järjestetään alkuosa...)
4       MERGE-SORT( A, mid + 1, right )     (... ja loppuosa)
5       MERGE( A, left, mid, right )        (limitetään osat siten, että järjestys säilyy)
```

Hajota:



Kuva 4: Jako osaongelmiin

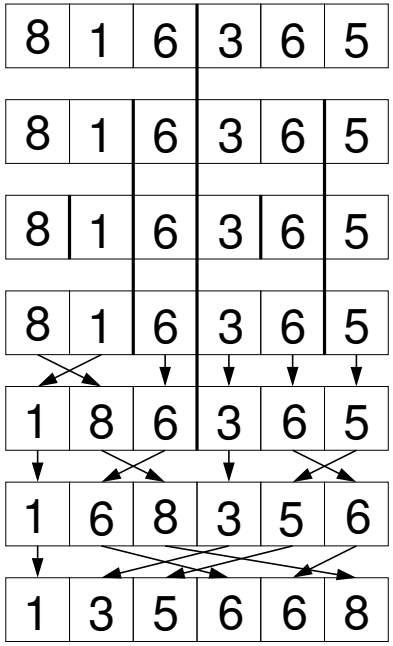
Hallitse:



Kuva 5: Osaongelmien ratkaisujen limitys

Eli:

- jaetaan järjestettävä taulukko kahteen osaan
- jatketaan edelleen osien jakamista kahtia, kunnes osataulukot ovat 0 tai 1 alkion kokoisia
- 0 ja 1 kokoiset taulukot ovat valmiiksi järjestyksessä eivätkä vaadi mitään toimenpiteitä
- lopuksi yhdistetään järjestyksessä olevat osataulukot limittämällä
- huomaa, että rekursiivinen algoritmi ei toimi kuvan tavalla molemmat puolet rinnakkain



– limityksen suorittava MERGE-algoritmi:

```
MERGE( A, left, mid, right )
1   for i := left to right do          (käydään koko alue läpi...)
2       B[ i ] := A[ i ]               (... ja kopioidaan se aputaulukkoon)
3   i := left                          (asetetaan i osoittamaan valmiin osan loppua)
4   j := left; k := mid + 1            (asetetaan j ja k osoittamaan osien alkuja)
5   while j ≤ mid and k ≤ right do     (käydään läpi, kunnes jompikumpi osa loppuu)
6       if B[ j ] ≤ B[ k ] then        (jos alkuosan ensimmäinen alkio on pienempi...)
7           A[ i ] := B[ j ]           (... sijoitetaan se tulostaulukkoon...)
8           j := j + 1                 (... ja siirretään alkuosan alkukohtaa)
9       else                           (muuten...)
10          A[ i ] := B[ k ]           (... sijoitetaan loppuosan alkio tulostaulukkoon...)
11          k := k + 1                 (... ja siirretään loppuosan alkukohtaa)
12      i := i + 1                     (siirretään myös valmiin osan alkukohtaa)
13  if j > mid then
14      k := 0
15  else
16      k := mid − right
17  for j := i to right do  (siirretään loput alkiot valmiin osan loppuun)
18      A[ j ] := B[ j + k ]
```

MERGE limittää taulukot käyttäen “pala kerrallaan” -menetelmää.

Tuottaako hajota ja hallitse tehokkaamman ratkaisun kuin pala kerrallaan?

Ei aina, mutta tarkastellaksemme tilannetta tarkemmin, meidän täytyy tutustua algoritmin analyysiin