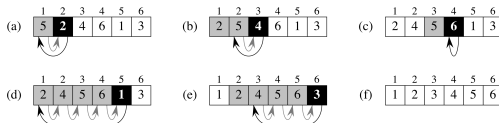


Invarianten

# Invarianter

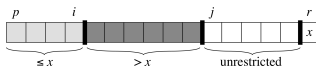
**Invariant:** Et forhold, som vedligeholdes af algoritmen gennem (dele af) dens udførelse. Udgør ofte kernen af ideen bag algoritmen.

Eksempel: Insertionsort:



Invariant: Alt til venstre for det sorte felt er sorteret.

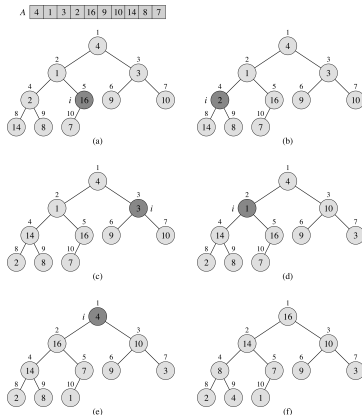
Eksempel: Partition fra Quicksort:



Invariant: Lysegrå del  $\leq x <$  mørkegrå del.

# Invarianter

Eksempel: Build-Heap:

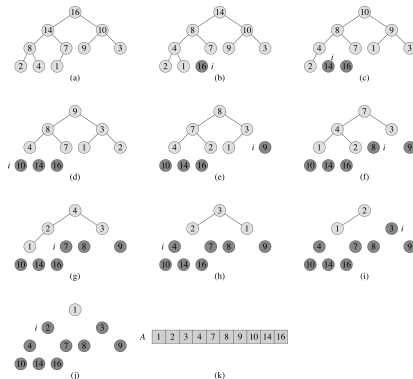


Invariant:

Undertræer, hvis rod har indeks større end den mørke knude, overholder heaporder.

# Invarianter

Eksempel: Heapsort (og enhver Selectionsort-baseret sortering):



Invariant: Det mørke er sorteret, og alt i det lyse er  $\leq$  det mørke.

# Invarianter

Eksempler fra søgetræer:

- ▶ Invariant under søgning:

Hvis søgte element findes, er det i det undertræ, vi er kommet til.

- ▶ Invariant under rebalancering efter indsættelse i et rød-sort træ:

Der kan være to røde knuder i træk på en rod-blad sti højst ét sted i træet, men bortset herfra er de rød-sortte krav overholdt.

- ▶ Invariant under rebalancering efter sletning i et rød-sort træ:

Der kan være én sværtet knude et sted i træet, og hvis sværtningen tælles med, er de rød-sortte krav overholdt.

# Invarianter, mere formelt

**Invariant** for algoritme: et udsagn om indholdet af hukommelsen (variable, arrays, ...) som:

- ▶ Er sandt efter alle skridt.
- ▶ Ved algoritmens afslutning kan korrekthed af output udledes af udsagnet (samt de omstændigheder som fik algoritmen til at stoppe).

I praksis ofte én invariant for én samling skridt (f.eks. 0 eller flere gennemløb af en **while** eller **for** løkke). En anden invariant kan så om nødvendigt bruges for næste samling skridt af algoritmen.

# Induktion

At en invariant gælder efter alle skridt vises ved hjælp af induktion:

- 1) Invariant overholdt i starten
- 2) Invariant overholdt før et skridt  $\Rightarrow$  overholdt efter

$\Rightarrow$  Invariant altid overholdt

(hvor “skridt” ofte er en iteration af en løkke). Dvs: Vis 1) og 2).

Induktion  $\sim$  “Dominoprincippet”:

- 1) Brik 1 falder
- 2) Brik  $k$  falder  $\Rightarrow$  brik  $k + 1$  falder

$\Rightarrow$  Alle brikker falder



# Brug af invarianter

Invarianter kan bruges på to forskellige detalje-niveauer (med en glidende overgang imellem dem):

1. Som værktøj til at udvikle algoritme-ideer: **Med den rette invariant fanges essensen af metoden**, og algoritmen skal “blot” skrives ud fra at denne invariant skal vedligeholdes.
2. Som værktøj til at nedskrive kode (eller detaljeret pseudo-kode) og **vise den konkrete kode korrekt**.

I første anvendelse er blødere beskrivelser (tekst, figur) passende, jvf. eksemplerne tidligere. I anden anvendelse må man nedskrive invarianten præcist i termer af konkrete variable fra koden, samt argumentere via den konkrete kodes ændringer af disse.

Eksemplet nedenfor med af finde største element i et array illustrerer dette.



# Eksempel

Find største element i array:

```
max = A[0]
i = 1
while i < A.length
  if A[i] > max
    max = A[i]
  i++
```

**Invariant  $S(k)$ :** “Efter den  $k$ 'te iteration af while-løkken indeholder  $\text{max}$  den største værdi af  $A[0..(i-1)]$ ”.

