Divide-and-Conquer algoritmer

Divide-and-Conquer algoritmer

Det samme som rekursive algoritmer.

Divide-and-Conquer algoritmer

Det samme som rekursive algoritmer.

En generel algoritme-udviklingsmetode:

1. Opdel problem i mindre delproblemer (af samme type).
2. Løs delproblemerne ved rekursion (dvs. kald algoritmen selv, men med de mindre input).
3. Konstruer en løsning til problemet ud fra løsningen af delproblemerne.

Basistilfælde: Problemer af mindste størrelse løses direkte (uden rekursion).

Divide-and-Conquer

Generel struktur af koden:

If basistilfælde
    Lokalt arbejde (løs problem af basisstørrelse)
Else
    Lokalt arbejde (f.eks. byg et eller flere delproblemer)
    Rekursivt kald
    Lokalt arbejde (f.eks. udnyt svar til at bygge næste delproblem)
    Rekursivt kald
    Lokalt arbejde (løs hovedproblem ud fra svar på delproblemer)

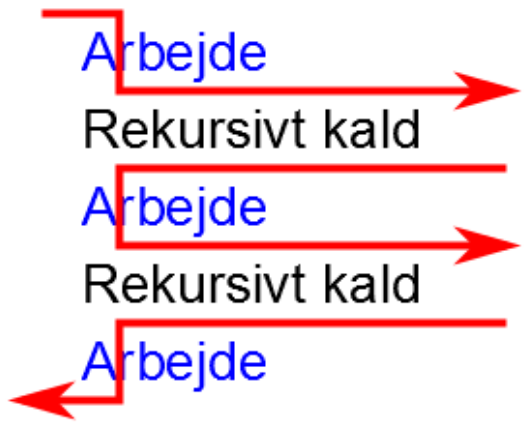
(Der behøver ikke altid være to rekursive kald. Nogle rekursive algoritmer har bare ét, og nogle har flere end to).

Divide-and-Conquer, flow of control

Flow of control (lokalt set, for ét kald af algoritmen):

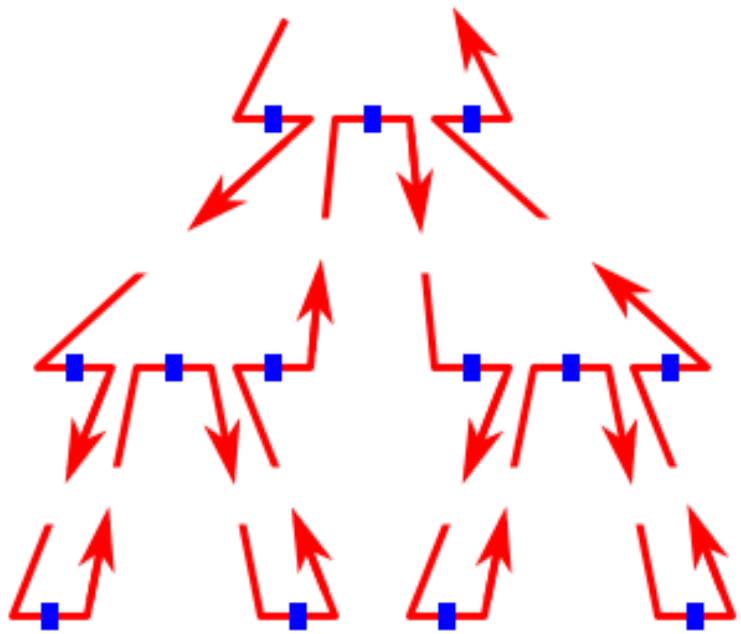
Basistilfælde

Ikke basistilfælde



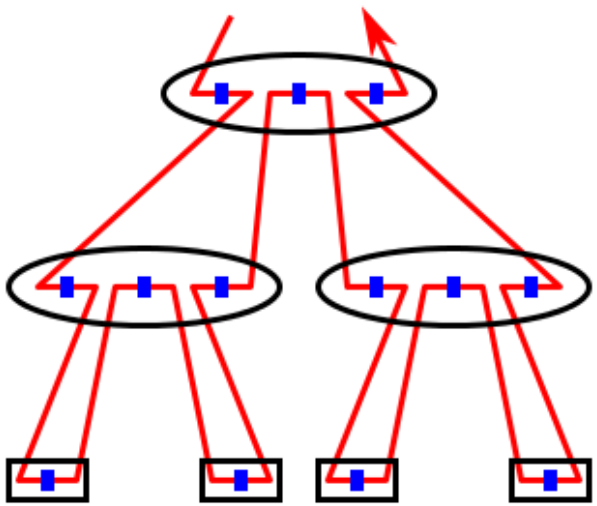
Divide-and-Conquer, udført arbejde

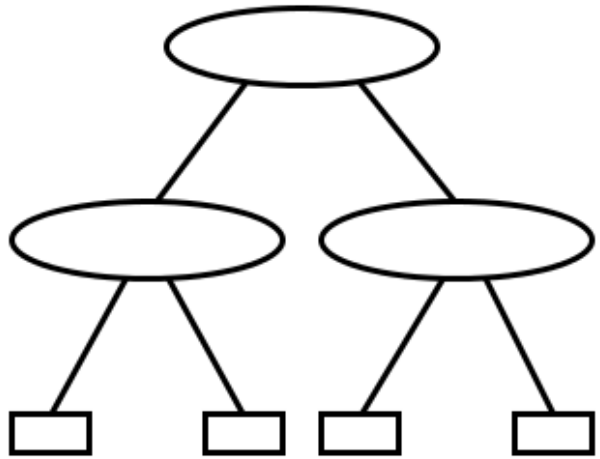
Globalt flow of control:



Rekursionstræer

Globalt flow of control = rekursionstræer:

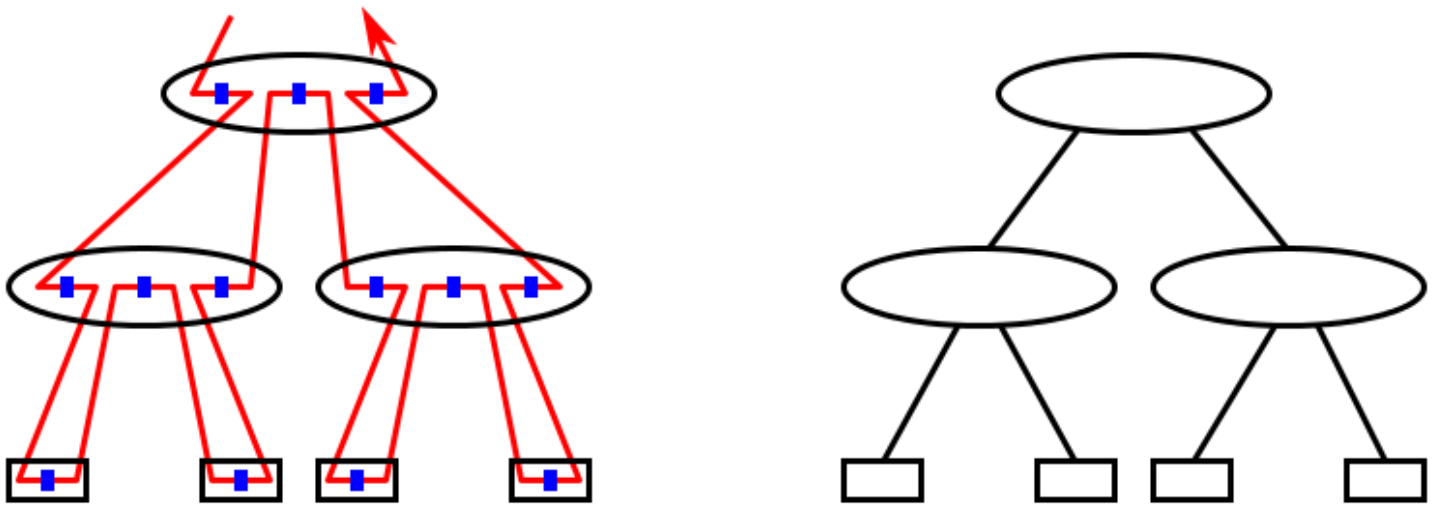




Én knude = ét kald af algoritmen.

Rekursionstræer

Globalt flow of control = rekursionstræer:



Én knude = ét kald af algoritmen.

Husk: alle kald på en sti fra roden mod aktive kald er "i gang", men sat på pause. Deres lokale variable og anden state opbevares (af operativsystemet) på en stak, så kaldenes udførsel ikke blandes sammen.

- Kald af barn i rekursionstræet = push på stak.
- Afslutning af et barns udførsel = pop fra stak.