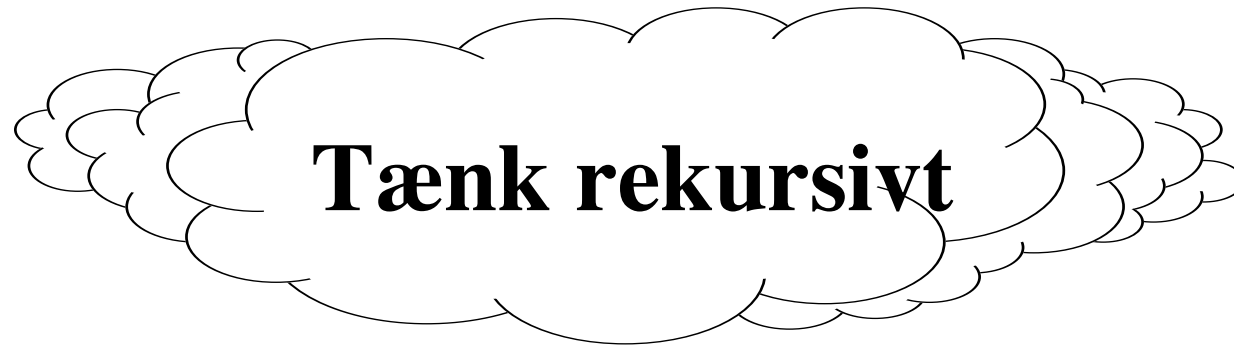


Rekursion og algoritmedesign



Rekursion

- **Rekursiv definition** af X : X defineres i termer af sig selv.
- Rekursion er nyttig, når en **generel** version af X kan defineres i termer af **simplere** versioner af X .
- Et problem **løses rekursivt** ved
 - (1) at **nedbryde** det i mindre delproblemer af samme slags,
 - (2) **fortsætte** med dette indtil delproblemerne er så simple, at de umiddelbart kan løses, og
 - (3) **kombinere** løsningerne af delproblemerne til en løsning af det oprindelige problem.



for at opnå

- simple og præcise definitioner
- elegante løsninger på problemer, der ellers er svære at løse
- algoritmer, der er simple at analysere

Del-og-hersk

- (1) **Del** problemet op i mindre delproblemer.
- (2) **Hersk** ved at løse hvert delproblem.
- (3) **Kombiner** resultaterne til en løsning for det oprindelige problem.

Hvis delproblemerne er mindre udgaver af det oprindelige problem, kan **rekursion** ofte benyttes med fordel.

Del-og-hersk ved rekursion

Pseudokode:

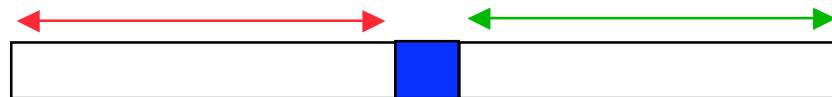
```
solve(Problem p) {  
    if (size(p) <= critical_size)  
        solve_small_problem(p);  
    else {  
        subproblem = divide(p);  
        solve(subproblem[0]);  
        solve(subproblem[1]);  
        ....  
        combine_solutions();  
    }  
}
```

Eksempel

Beregning af den maksimale delsekvenssum

Problem. Givet en sekvens (a_1, a_2, \dots, a_n) af reelle tal.
Find en delsekvens $(a_i, a_{i+1}, \dots, a_j)$ af konsekutive elementer, sådan at summen af dens elementer er størst mulig.

Opdel problemet i to omtrent lige store dele:



Enten findes den maksimale delsekvens

(1) helt i den **venstre** del,



(2) helt i den **højre** del, eller



(3) i en sekvens, der indeholder **midterelementet**



3 delproblemer

De to første værdier bestemmes ved rekursion.

Den sidste værdi bestemmes som summen af

- det maksimale **suffix** for sekvensen til venstre for midterelementet (inklusive dette), og
- det maksimale **prefix** for sekvensen til højre for midterelementet.



Den maksimale delsekvenssum bestemmes som maksimum af disse 3 værdier.

Javakode

```
int maxSum(int[] a, int left, int right) {
    if (left == right)
        return a[left] > 0 ? a[left] : 0;
    int mid = (left + right) / 2;
    return max3(maxSum(a, left, mid - 1),
                maxSum(a, mid + 1, right),
                maxSuffix(a, left, mid) + maxPrefix(a, mid + 1, right));
}
```

```
int maxSuffix(int[] a, int left, int right) {
    int sum = 0, maxSum = 0;
    for (int i = right; i >= left; i--)
        if ((sum += a[i]) > maxSum)
            maxSum = sum;
    return maxSum;
}
```

`maxPrefix` implementeres analogt

Kompleksitet

Tiden, det tager at løse et problem med n tal, $T(n)$, hvor n er en potens af 2, opfylder rekursionsligningerne

$$T(n) = 2 * T(n/2) + O(n)$$

$$T(1) = O(1)$$

Hvis $O(n)$ og $O(1)$ erstattes med henholdsvis n og 1, bliver løsningen

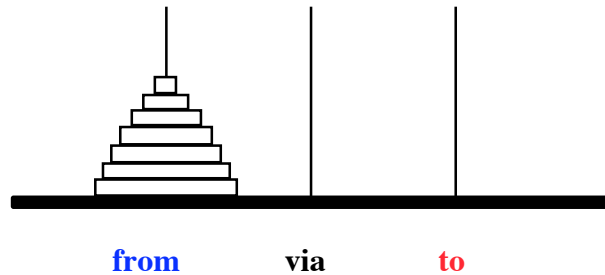
$$T(n) = n \log n + n$$

Der gælder således, at

$$T(n) = O(n \log n)$$

Tårnene i Hanoi

(fra et munkekloster i Tibet)



Problem. Flyt skiverne fra pinden **from** til pinden **to**, idet en større skive aldrig må placeres oven på en mindre skive.

At flytte n skiver fra pinden **from** til pinden **to** kan foretages ved først at flytte de øverste $n-1$ skiver fra pinden **from** til pinden **via**.

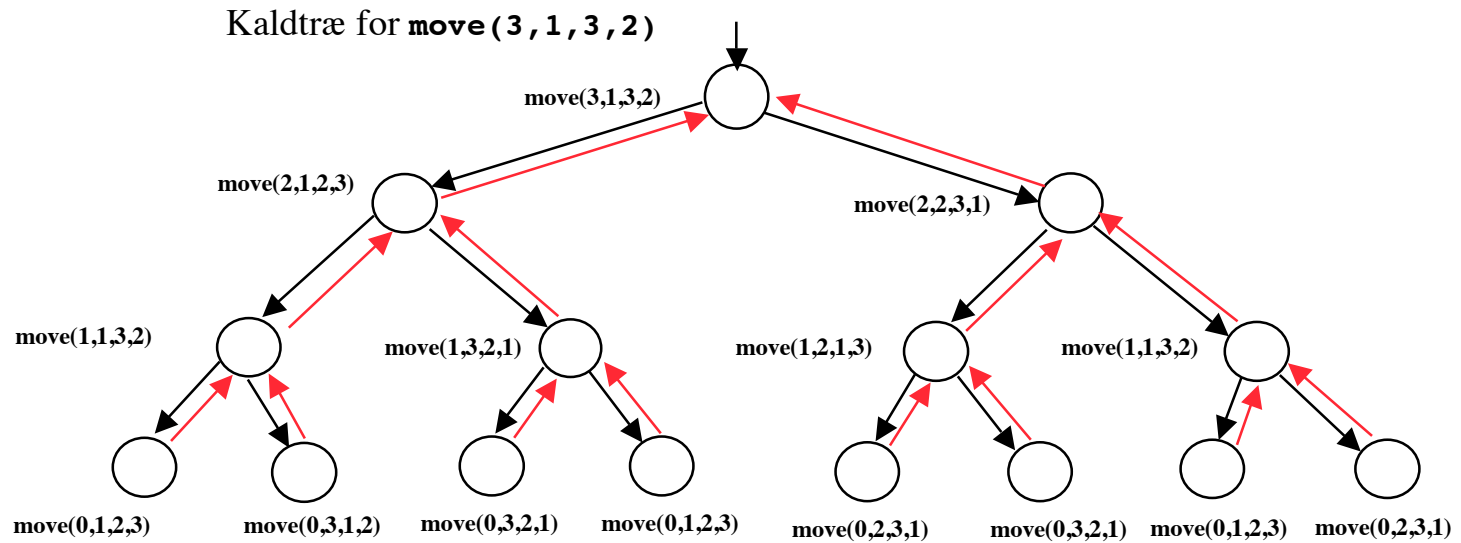
Dernæst flyttes den nederste skive fra pinden **from** til pinden **to**.

Endelig flyttes de $n-1$ skiver fra pinden **via** til pinden **to**.

```

void move(int n, int from, int to, int via) {
    if (n == 0)
        return;
    move(n-1, from, via, to);
    System.out.println("Move " + from + " to " + to);
    move(n-1, via, to, from);
}

```



Effektivitetsanalyse

Tidsforbruget er proportionalt med antallet af flytninger, $F(n)$, hvor n angiver antallet af skiver.

$$F(n) = F(n-1) + 1 + F(n-1) = 2 * F(n-1) + 1, \text{ for } n > 1$$
$$F(1) = 1$$

som har løsningen $2^n - 1$.

Pladsforbruget er proportionalt med det maksimale antal uafsluttede kald af **move**, dvs. n .

Samlet tidsforbrug for 64 skiver, hvis hver flytning tager 1 sekund:

$$2^{64} \text{ sekunder} \approx 10^{19} \text{ sekunder} \approx 10^{12} \text{ år}$$

Fundamentale regler for rekursion

- **Basistilfælde:** Hav altid mindst et tilfælde, der kan løses uden brug af rekursion.
- **Gør fremskridt:** Ethvert rekursivt kald bør nærme sig et basistilfælde.
- **Tro på det:** Antag altid, at et rekursivt kald virker som ønsket.
- **Undgå dobbeltarbejde:** Sørg for at hvert delproblem kun løses én gang.

Benyt aldrig rekursion som en erstatning for en simpel løkke.