計算機ソフトウェア
第五回

電気電子工学科
黒橋禎夫

再帰呼び出し

- ハノイの塔

  n: ディスク枚数, A: スタート, B: 作業域, C: ゴール

  アルゴリズム MOVE(n, A, B, C)

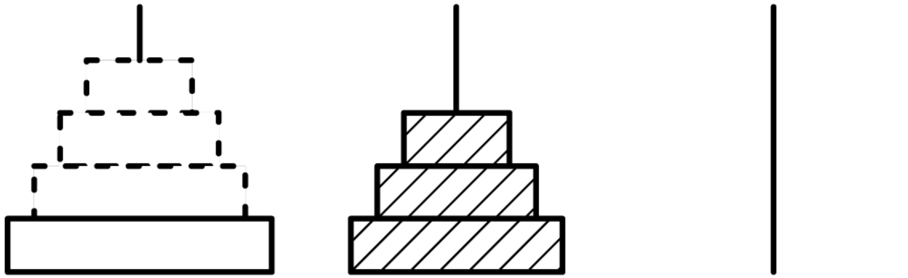
    if n=1, AからCへ移す

    else MOVE(n-1, A, C, B)

    n枚目のディスクをAからCへ移す

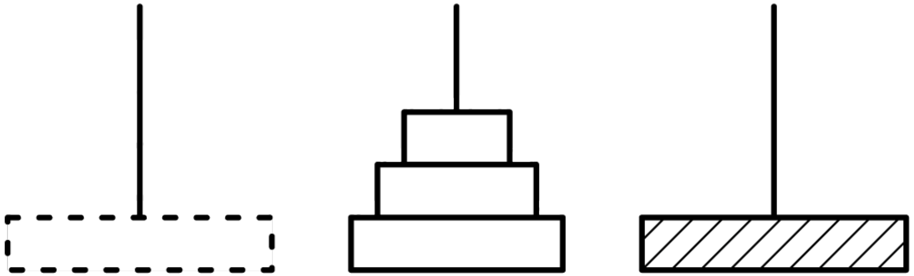
    MOVE(n-1, B, A, C)

ハノイの塔

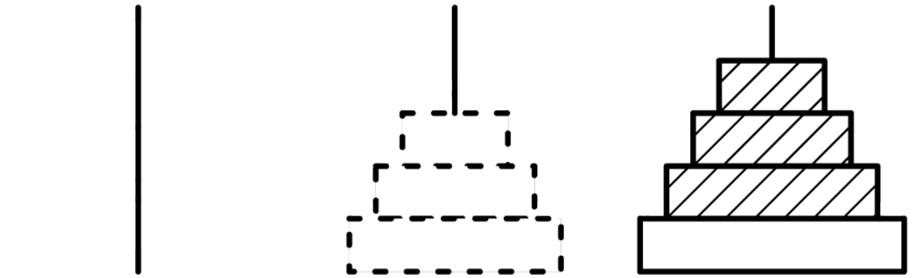


MOVE(n-1, A, C, B)

▪再帰呼び出し



A → C



MOVE(n-1, B, A, C)

▪再帰呼び出し

分割統治法
divide-and-conquer method

- 再帰的に分割すると計算量が減らせる問題
  ex.マージソート

  アルゴリズム MERGESORT(s)
  if |s| = 2, return(min(s), max(s))
  else　sをだいたい同じ大きさの s1, s2へ分割
  MERGESORT(s1) と MERGESORT(s2) を
  マージする

再帰呼び出しの計算量

- ハノイの塔

  $f(n) = 2f(n-1) + a = O(2^n)$

- マージソート

  $f(n) = 2f(n/2) + cn = O(n \cdot \log n)$

  同じ再帰なのに、

  なんでマージソート（分割統治）は速いのか？

  nが減るのが速いから！

分割統治法の計算量を一般的に議論する

$$f(n) = \begin{cases} c & (n = 1) \\ af(n/b) + cn & (n \geq 2) \end{cases}$$

ならば

$$f(n) = \begin{cases} O(n) & (a < b) \\ O(n \log n) & (a = b) \\ O(n^{\log_b a}) & (a > b) \end{cases}$$

かけ算

x: $\boxed{p \mid q}$, y: $\boxed{r \mid s}$

上位ビットと下位ビットに分割して

$p \times r,\ q \times s,\ (p+q) \times (r+s)$

を再帰的に計算する

$f(n) = 3f(n/2) = O(n^{1.58}) < O(n^2)$

単純にかけ算を実行するより速い

フィボナッチ数列（１）

$F(0) = F(1) = 1$
$F(n) = F(n-2) + F(n-1)$　（ $n \geqq 2$ ）
　　1, 1, 2, 3, 5, 8, 13, …

アルゴリズム FIBO1(n)
　　if n = 0 または n = 1, return 1
　　else return FIBO(n-1) + FIBO(n-2)

計算量： $f(n) = f(n-1) + f(n-2) + c = O(2^n)$
　（指数オーダでは計算終わらない）

フィボナッチ数列（２）

アルゴリズムFIBO2(n)

```
    if  n = 0 または n = 1,  return 1
    else  x ← 1, y ← 1
        for  i ← 2, until n
            z ← x + y
            x ← y
            y ← z
```

計算量: O(n) （結構大変）