

Analisis Rekurens

Tim Pengajar IF1210

Sekolah Teknik Elektro dan Informatika

The Handshake Problem

Ada n orang di dalam sebuah ruangan. Jika masing-masing orang harus bersalaman dengan setiap orang lainnya, ada berapa *handshakes* $h(n)$ yang terjadi?



Bagaimana cara memecahkan persoalan ini?

Latihan: The Handshake Problem

Ide penyelesaian:

- Setiap orang dapat berpartisipasi untuk menyelesaikan persoalan ini.
 - Bagaimana “versi kecil” dari persoalan yang dapat dibantu penyelesaiannya?

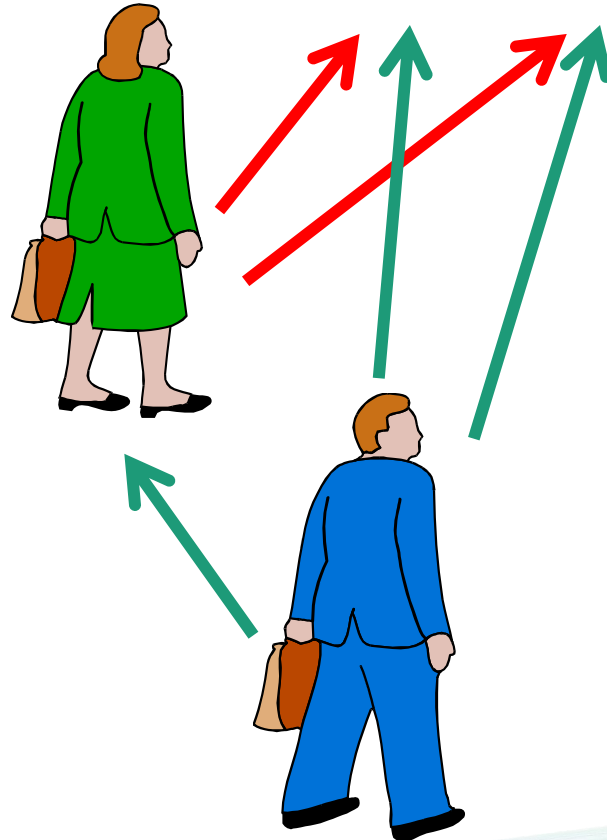
1 kali salaman

+

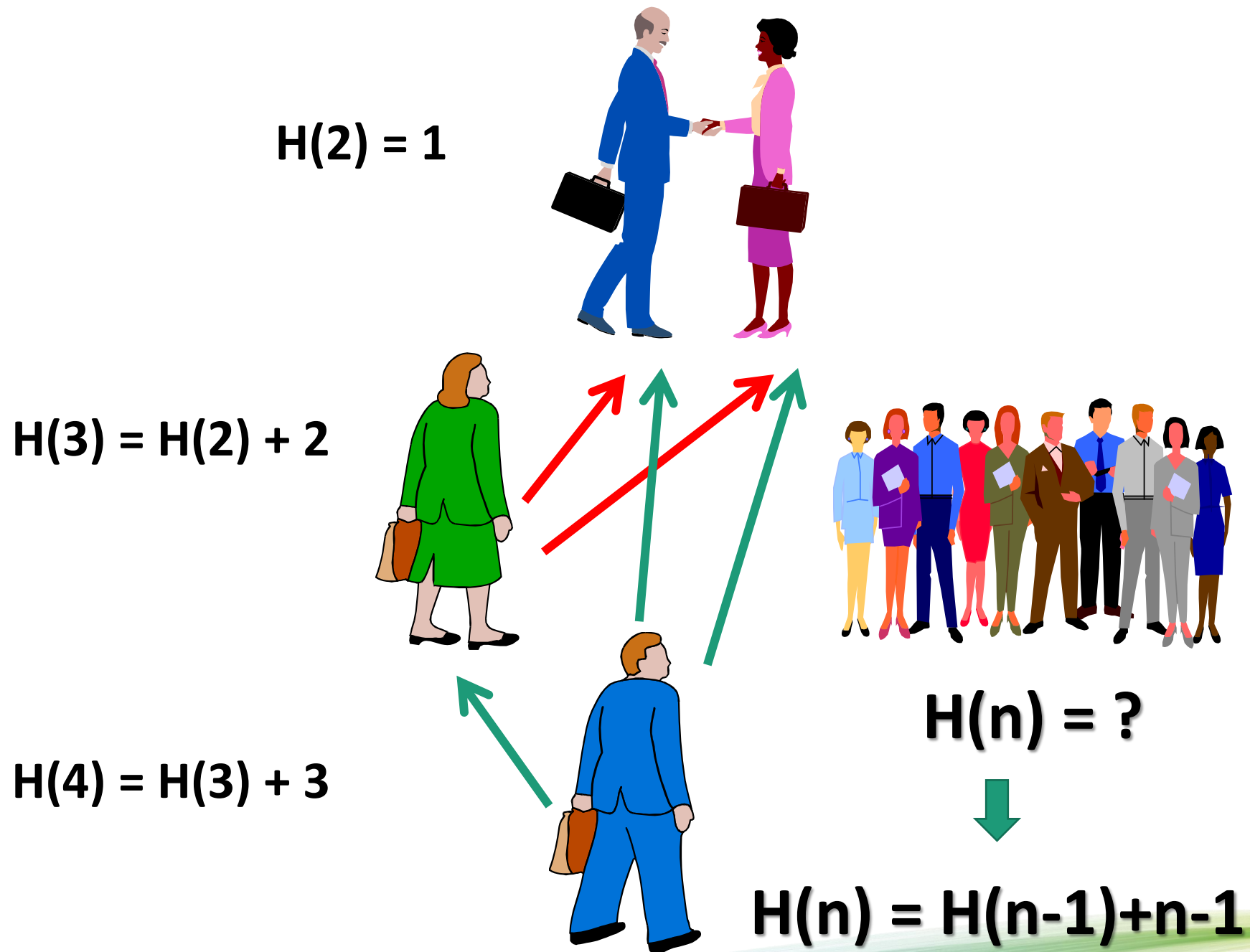


2 kali salaman

+



3 kali salaman



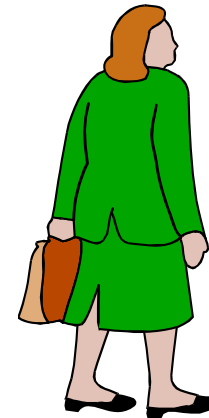
$$h(n) = h(n-1) + n-1$$



$$h(4) = h(3) + 3$$



$$h(3) = h(2) + 2$$



Kasus basis:
 $h(2) = 1$

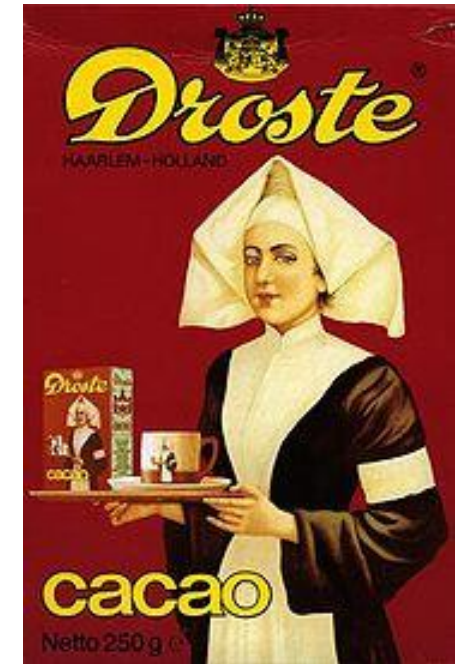


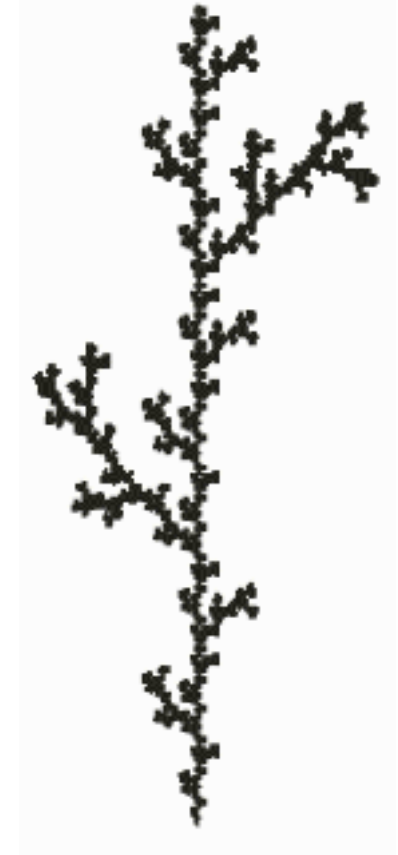
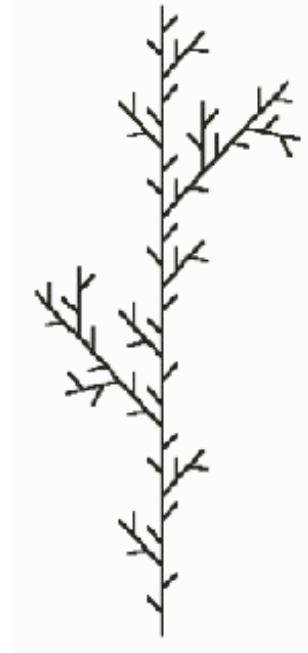
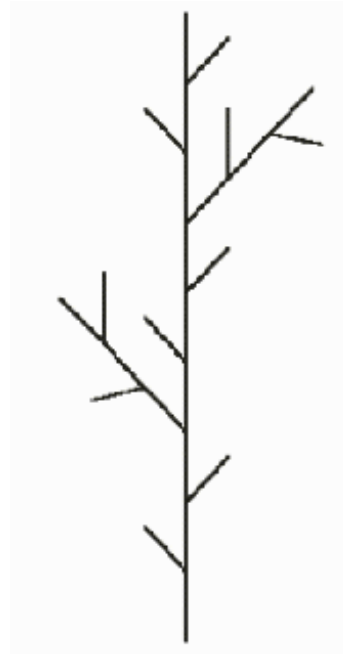
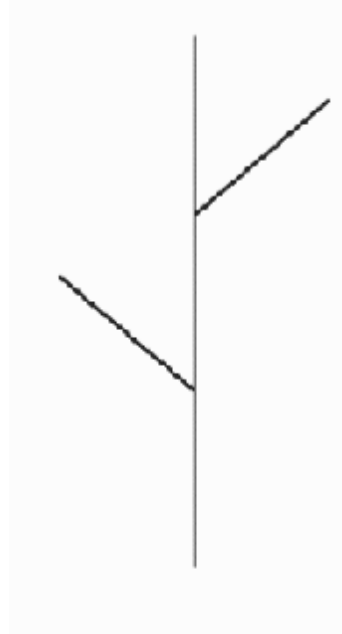
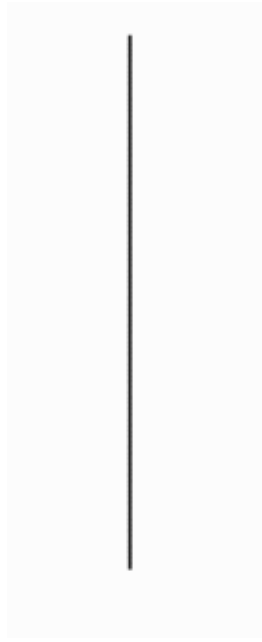
$$h(n) \Rightarrow \text{Sum dari integer mulai 1 hingga } n-1 \\ = n(n-1) / 2$$

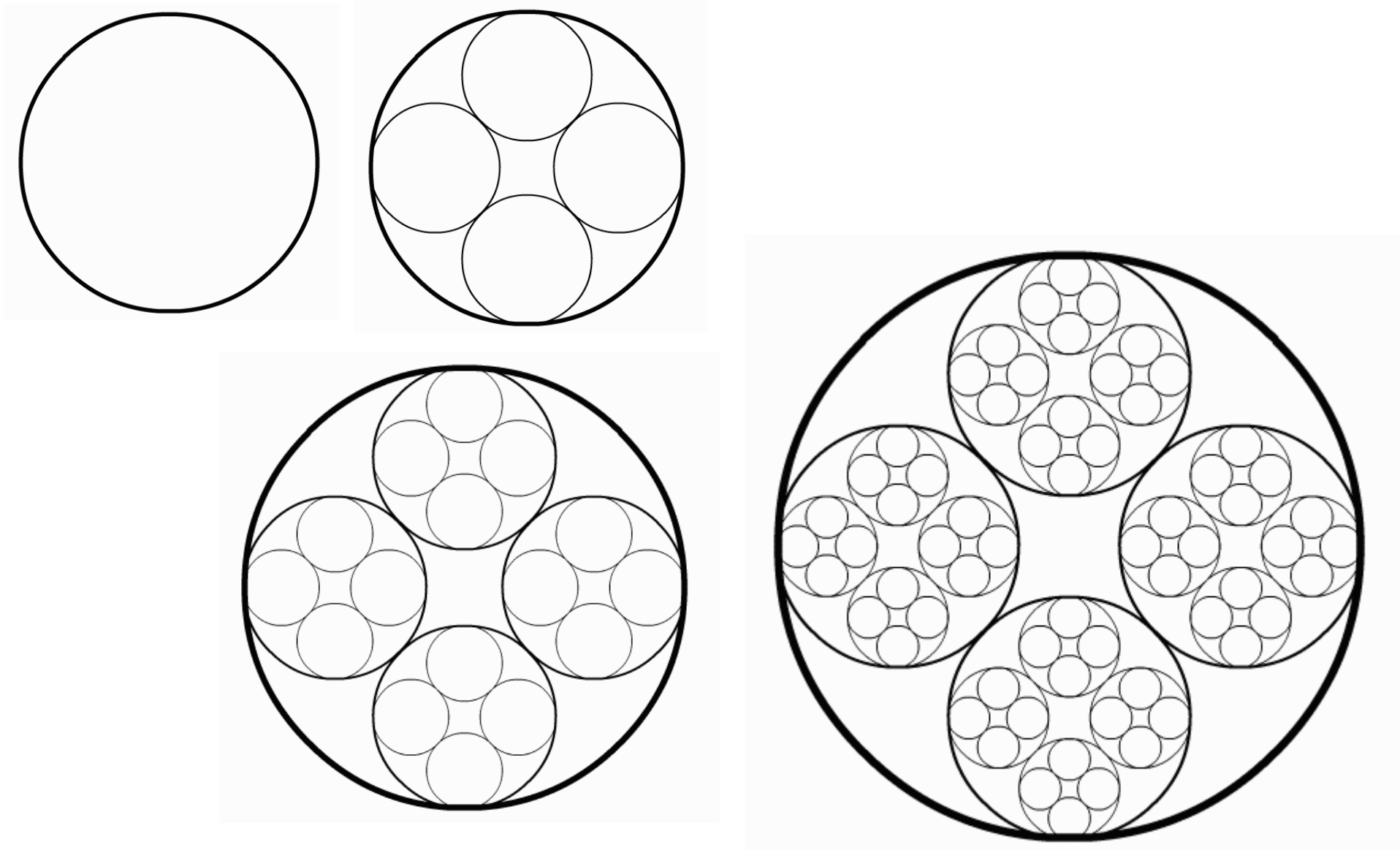
**Recursion is all about
breaking a big problem into
smaller occurrences of that
same problem.**

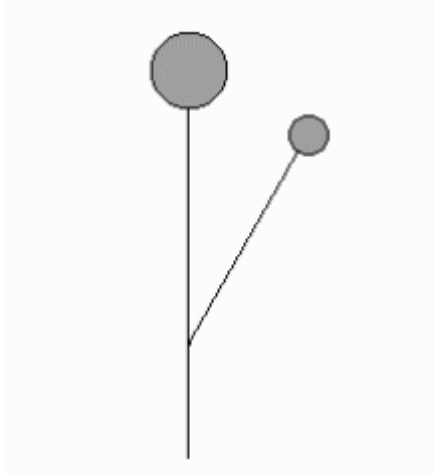
Rekursifitas

- Definisi:
 - Suatu entitas disebut **rekursif** jika pada definisinya terkandung terminologi dirinya sendiri

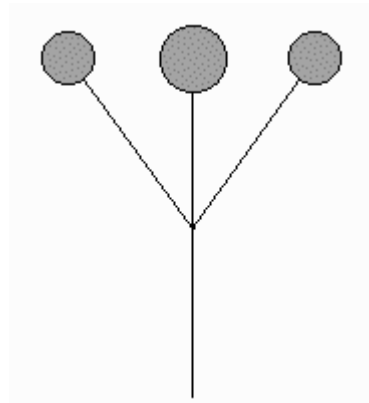




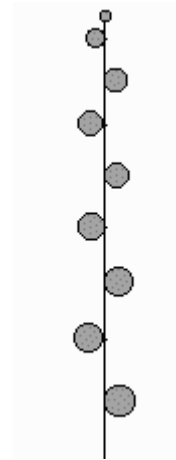




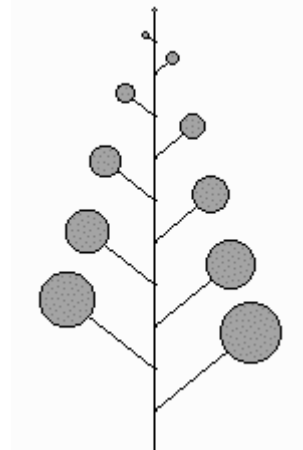
monochasium



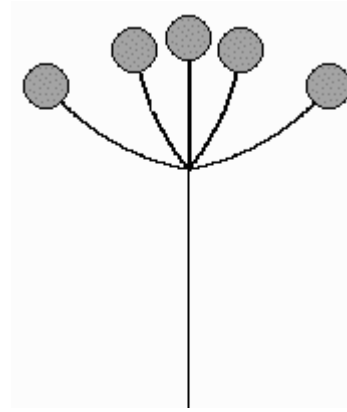
dichasium



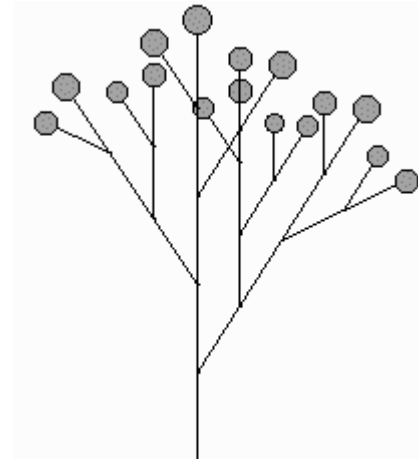
spike



raceme



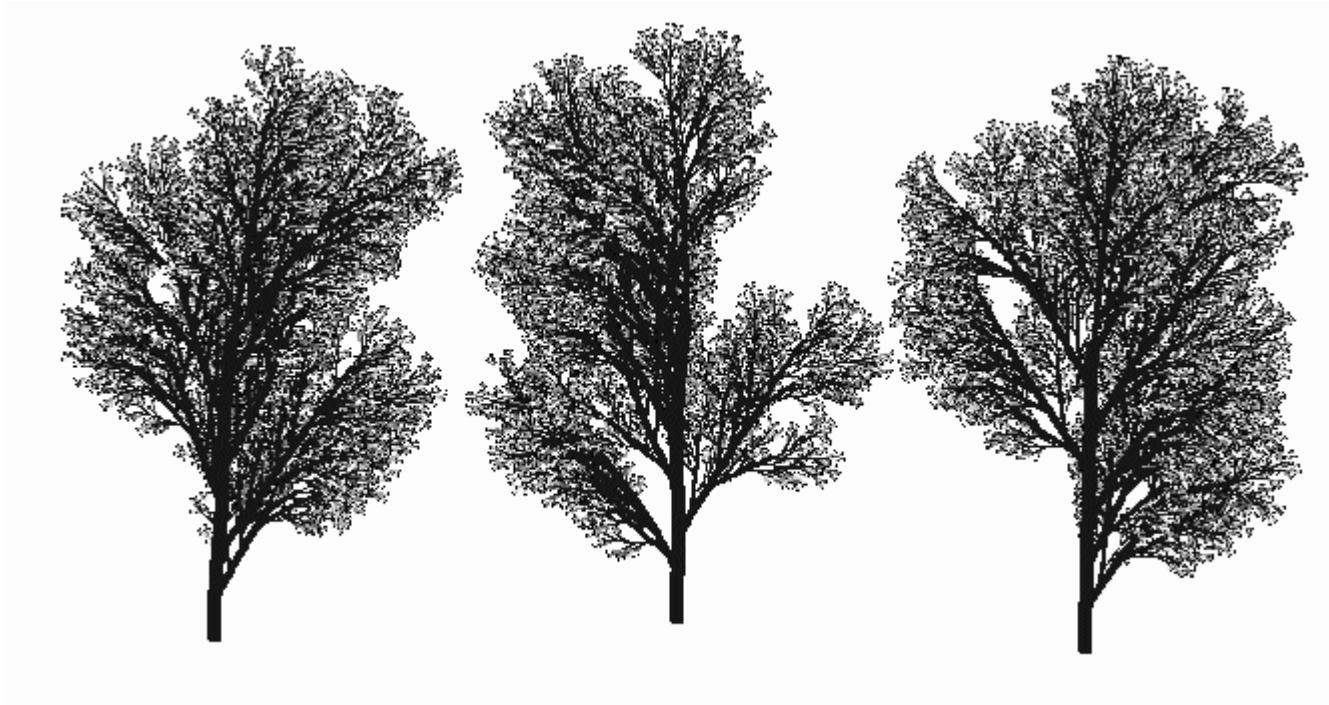
umbel



panicle

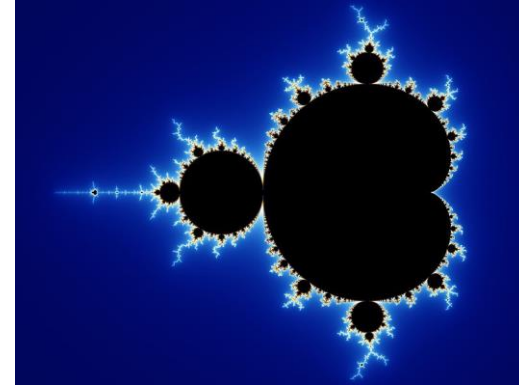
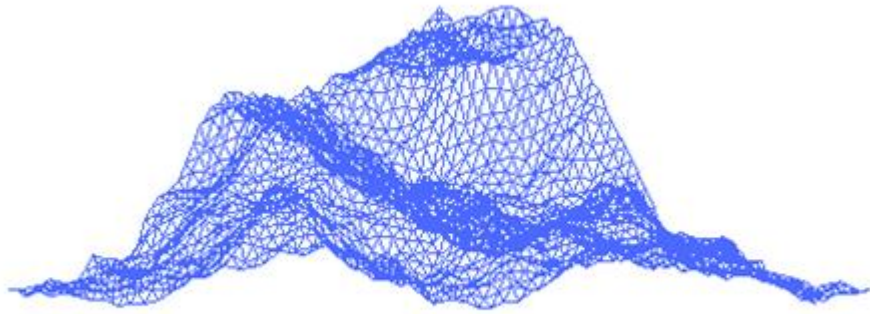
Plant Forms

some common inflorescences



Trees modeled using deReffye's method

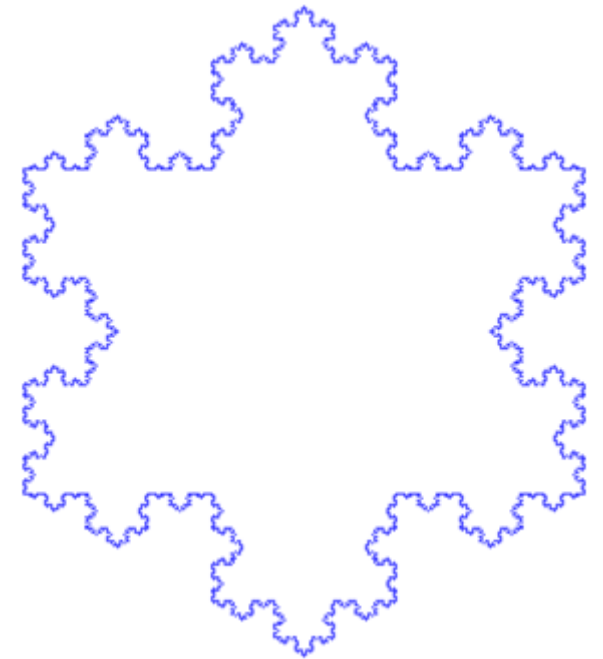
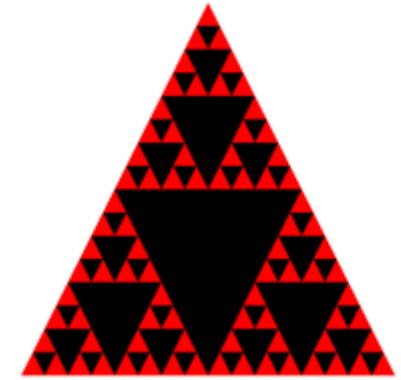
Contoh-contoh gambar fraktal



Aplikasi Rekursifitas dalam bidang informatika

- **Fraktal:**

- bentuk-bentuk geometris yang terdiri atas bagian-bagian kecil, tiap bagian adalah kopi (dalam ukuran yang lebih kecil) dari bentuk keseluruhan
- Biasanya diimplementasikan dari fungsi matematis yang bersifat rekursif



Ekspresi Rekursif

- Definisi **entitas** (type, fungsi) disebut **rekursif** jika definisi tersebut mengandung terminologi dirinya sendiri
- **Ekspresi rekursif** direalisasikan dengan membuat fungsi rekursif dan didasari **analisis rekurens**



Akan dibahas pada materi berikutnya

Analisis Rekurens

- Solusi rekursif terdiri dari dua bagian:
 - **Basis**, kasus yang menyebabkan fungsi berhenti, karena jawaban sudah bisa diperoleh
 - Bagian **rekurens** : mengandung call terhadap fungsi tersebut (aplikasi dari fungsi), dengan parameter bernilai mengecil (menuju basis).
- Solusi rekursif memiliki sekurang-kurangnya satu kasus basis dan satu kasus rekursif.
 - Dimungkinkan ada lebih dari satu kasus basis maupun rekursif.
- Bagian terpenting dari analisis rekurens adalah mengidentifikasi kasus-kasus ini.

Contoh Lain...



- Bagaimana caranya kita memisahkan menjadi 2 bagian yang sama permen M&M dari mangkok besar ini, tanpa harus **terlebih dahulu** menumpahkan seluruh isinya atau menghitung jumlahnya?
- Bagaimana jika beberapa orang membantu memecahkan persoalan ini? Bisakah tiap orang melakukan sebagian kecil dari pekerjaan?
 - Berapa jumlah M&M yang mudah dibagi menjadi dua (bahkan kalau untuk anak kecil yang tidak bisa menghitung?)

Kerangka Fungsi Rekursif (Notasi Haskell dengan Guard)

```
<fungsi> <list-parameter>  
  | <kondisi-basis>      = <ekspresi-1 >  
  | <kondisi-rekurens> = <fungsi> (<ekspresi-2>)
```

Catatan:

<kondisi-rekurens> bisa menggunakan
"otherwise"

Nilai parameter
mengecil
menuju basis



Tulishlah secara eksplisit dalam teks program anda: mana bagian basis, mana rekurens berupa komentar dalam program

Kerangka Fungsi Rekursif (Notasi Haskell dengan if-then-else)

```
<fungsi> <list-parameter> =  
  if <kondisi-basis> then <ekspresi-1 >  
  else <fungsi>(<ekspresi-2>) -- kondisi rekurens
```

Nilai parameter
mengecil
menuju basis



Tuliskan secara eksplisit dalam teks program anda: mana bagian basis, mana rekurens berupa komentar dalam program

Studi Kasus 1: Definisi Faktorial

- Definisi 1:

$$0! : 1$$

$$N! : N * (N-1) !$$

Nilai N mengecil
menuju basis

- Definisi 2:

$$1! : 1$$

$$N! : N * (N-1) !$$

- Definisi 3:

$$1! : 1$$

$$N! : (N+1)! / (N+1)$$

Nilai N tidak pernah mencapai basis
→ **TIDAK** bisa diimplementasi
menjadi fungsi rekursif

Jika domain nilai terkecil 0,
Menggunakan basis $n=0$

-- Faktorial (definisi-1)

fac(n)

-- DEFINISI DAN SPESIFIKASI

fac :: Int -> Int

{- fac(n) = $n!$ sesuai dengan definisi rekursif factorial,
dengan basis $n = 0$ -}

-- REALISASI { menggunakan Guard }

fac n

	n == 0	= 1	-- Basis
	otherwise	= fac (n-1) * n	-- Rekurens

Terdapat aplikasi fungsi fac di
dalam realisasi → rekurens

```
-- Faktorial2 (definisi-2)
```

```
fac2(n)
```

```
-- DEFINISI DAN SPESIFIKASI
```

```
fac2 :: Int -> Int
```

```
{- fac2(n) = n! sesuai dengan definisi rekursif factorial,  
   dengan basis 1 -}
```

```
-- REALISASI menggunakan if-then-else
```

```
fac2 n = if n==1 then      -- Basis  
         1  
        else              -- Rekurens  
         fac2 (n-1) * n
```

Karena nilai terkecil dari domain adalah 1,
maka harus menggunakan basis $n = 1$

Studi Kasus 2: FPB(m,n) Faktor Persekutuan Terbesar

- **m,n: integer > 0**
- Alternatif solusi 1: hasil kali faktor prima yang sama dari m dan n → pendekatan prosedural
- Alternatif solusi 2: $\text{fpb} \leq \min(m,n)$
 - Cari minimum m,n → kurangi satu sampai ketemu bilangan yang merupakan pembagi m dan n
- Alternatif solusi 3:
 $\text{fpb}(m,n) = \text{fpb}(n, m \bmod n)$

-- FPB (solusi-3)

fpb(m,n)

-- DEFINISI DAN SPESIFIKASI

fpb :: Int -> Int -> Int

{- fpb(m,n) menghasilkan bilangan integer terbesar yang dapat membagi m dan n tanpa sisa -}

-- REALISASI

```
fpb m n =
  if ((mod m n)==0) then -- Basis
    n
  else -- Rekurens
    fpb n (mod m n)
```

Ide: $\text{fpb}(m,n) = \text{fpb}(n, m \bmod n)$.
Kasus basis?

Jika n habis membagi m, maka
 $(m \bmod n) = 0$ dan $\text{fpb} = n$

Latihan Soal

Buatlah definisi, spesifikasi, dan realisasi dari fungsi-fungsi berikut (menggunakan pendekatan rekursif):

1. **DeretSegitiga**, merupakan fungsi untuk mencari nilai bilangan ke- n pada deret segitiga.
Deret segitiga: 1, 3, 6, 10, 15, ...
2. **IsGanjil**, merupakan predikat untuk memeriksa apakah sebuah bilangan integer (≥ 0) merupakan bilangan ganjil.
3. **LuasBS**, merupakan fungsi untuk menghitung luas bujur sangkar dengan panjang sisi tertentu.

Latihan Soal

4. **SumOfDigits**, menghitung hasil penjumlahan dari setiap bilangan tunggal yang terdapat di dalam sebuah bilangan integer positif.

Misalnya:

- $\text{SumOfDigits}(234) = 2 + 3 + 4 = 9$
- $\text{SumOfDigits}(38) = 3 + 8 = 11$
- $\text{SumOfDigits}(5) = 5$

Apabila didefinisikan bahwa **SumOfDigits** dari bilangan negatif dilakukan dengan mengabaikan tanda '-', buatlah fungsi **SumOfDigitsPosNeg** yang menangani hal ini.

Misalnya: $\text{SumOfDigitsPosNeg}(-45) = 4 + 5 = 9$

Latihan Soal

5. Buatlah fungsi **sumRange** yaitu fungsi yang menerima masukan 2 bilangan integer, misalnya a dan b yang menyatakan rentang bilangan dengan syarat: $a \leq b$; a dan b bilangan positif; dan menghasilkan penjumlahan semua bilangan dari a s.d. b.

Harus dikerjakan secara rekursif.

Contoh aplikasi:

- $\text{SumRange}(2,2) = 2$
- $\text{SumRange}(2,4) = 2+3+4 = 9$

Kenapa Membuat Program Rekursif?

- Persoalan memang mengandung definisi “rekursif”.
Contoh: **faktorial**.
- Mengolah **data bertype rekursif**.
Contoh: **pemrosesan list** → **materi berikutnya**
- Programmernya ingin menulis solusi rekursif. Persoalan yang tidak rekursif tidak perlu diselesaikan secara rekursif, tapi bisa diselesaikan secara rekursif. Ada banyak jalan ke Roma.....
Contoh: **fpb (Faktor Persekutuan Terbesar)**

Mencari Solusi Rekursif

- Solusi rekursif karena definisi persoalan rekursif
 - Baca baik-baik persoalan, dan buat definisi rekursif sesuai definisi persoalan
- Solusi rekursif karena persoalan diwakili oleh struktur data rekursif → **materi berikutnya**
- Solusi rekursif untuk persoalan non rekursif
 - Buat fungsi rekursif dengan parameter

Intermezzo: How many pairs of rabbits can be produced from a single pair in a year's time?

- Asumsi:

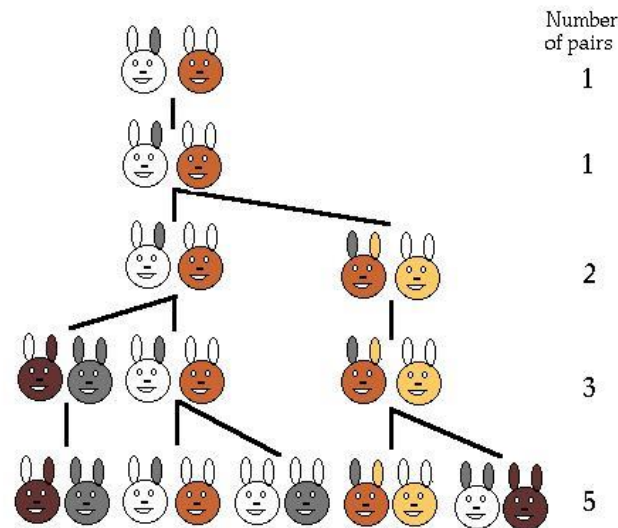
- Setiap pasang kelinci akan memberikan sepasang anak setiap bulan;
- setiap pasang kelinci baru akan mulai melahirkan setelah berusia dua bulan;
- tidak ada kelinci yang mati pada tahun itu.

- Contoh:

- Setelah 1 bulan akan ada 2 pasang kelinci;
- setelah 2 bulan akan ada 3 pasang kelinci;
- setelah 3 bulan akan ada 5 pasang kelinci (karena pasangan yang lahir pada bulan pertama sudah akan mulai melahirkan anak).



Population Growth in Nature



- Leonardo Pisano (Leonardo Fibonacci = Leonardo, son of Bonaccio) proposed the sequence in 1202 in *The Book of the Abacus*.
- Fibonacci numbers are believed to model nature to a certain extent, such as Kepler's observation of leaves and flowers in 1611.

Fibonacci Numbers

- Fibonacci numbers:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

where each number is the sum of the preceding two.

- Recursive definition:

- $F(0) = 0;$

- $F(1) = 1;$

- $F(\text{number}) = F(\text{number}-1) + F(\text{number}-2);$



Apa yang sudah dipelajari?

- Rekursifitas dan analisis rekurens
- Memanfaatkan analisis rekurens untuk konstruksi program rekursif

Sumber

- Diktat “Dasar Pemrograman, Bag. Pemrograman Fungsional” oleh Inggriani Liem, revisi Februari 2014