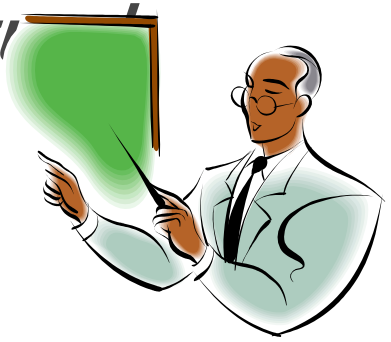
Bab 5

- *Proof Strategy*
- *Sequences and Summations*
- *Mathematical Induction*
- *Recursive Definitions and Structural Induction*
- *Recursive Algorithms*
- *Program Correctness*



Sequences and Summations
(Baris dan Deret)

Tujuan Instruktional Khusus

- Memahami konsep *sequence* (Baris)
- Memahami konsep *summmations* (Deret)

Baris

- Adalah sebuah fungsi dari himpunan bagian integer ke suatu himpunan S.
- Himpunan bagian integer yang dimaksud adalah {0, 1, 2, 3, …} atau {1, 2, 3, …}
- Notasi a_n adalah term dari sebuah Baris
- Notasi $\{a_n\}$ menggambarkan sebuah Baris
- S = { a_0, a_1, a_2, a_3, …, a_n } atau { a_1, a_2, a_3, …, a_n}
- Contoh : Sebuah Baris $\{a_n\}$ dimana $a_n = 1/n$
  - Maka Baris tersebut adalah 1, 1/2, 1/3, 1/4,….

Baris

- Geometric progression:
  - a, ar, ar^2, ar^3, …, ar^n dimana a adalah *initial term* dan r adalah *common ratio*
  - Contoh: Baris $\{b_n\}$ dimana $b_n = (-1)^n$ maka Baris tersebut adalah −1, 1, −1, 1, …

- Arithmetic progression:
  - a, a+d, a+2d, …, a+nd dimana a adalah initial term dan d adalah common difference merupakan bilangan real.
  - Contoh: Baris $\{s_n\}$ dimana $s_n = -1 + 4n$ maka Baris tersebut adalah −1, 3, 7, 11, …

- String: $a_1 a_2 a_3 \ldots a_n$
- Empty string = λ

Contoh

- Baris $\{b_n\}$ dengan $b_n = (-1)^n$, $\{c_n\}$ dengan $c_n = 2 \times 5^n$, $\{d_n\}$ dengan $d_n = 6 \times (1/3)^n$ merupakan *geometric progressions* dengan *initial term* dan *common ratio* –1 dan –1; 10 dan 5; 2 dan 1/3.
  - Maka Baris yang dihasilkan :
    - $\{b_n\} = b_1, b_2, b_3, b_4,\ldots = -1,1,-1,1,\ldots$
    - $\{c_n\} = c_1, c_2, c_3, c_4,\ldots = 10,50,250,1250,\ldots$
    - $\{d_n\} = d_1, d_2, d_3, d_4,\ldots = 2,2/3,2/9,2/27,\ldots$
- Baris $\{s_n\}$ dengan $s_n = -1 + 4n$, $\{t_n\}$ dengan $t_n = 7 - 3n$ merupakan *arithmetic progressions* dengan *initial term* dan *common differences* –1 dan 4; 7 dan –3. Dan n dimulai dari 0.
  - Maka Baris yang dihasilkan :
    - $\{s_n\} = s_0, s_1, s_2, s_3,\ldots = -1,3,7,11,\ldots$
    - $\{t_n\} = t_0, t_1, t_2, t_3,\ldots = 7,4,1,-2,\ldots$

Deret

$$\sum_{j=m}^{n} a_j = a_m + a_{m+1} + a_{m+2} + \ldots + a_n$$

- m disebut **batas bawah**, n disebut **batas atas**, j disebut **indeks**
- *Double summation* bisa dilihat sebagai berikut:

$$\sum_{i=m}^{n}\sum_{j=p}^{q} j = \sum_{i=m}^{n}\left[p + (p+1) + (p+2) + \ldots + q\right]$$

Contoh

- Berapa nilai dari $\sum_{j=1}^{5} j^2$

$$\sum_{j=1}^{5} j^2 = 1^2 + 2^2 + 3^2 + 4^2 + 5^2$$
$$= 1 + 4 + 9 + 16 + 25$$
$$= 55$$

- Berapa nilai dari $\sum_{i=1}^{4}\sum_{j=1}^{3} ij$

$$\sum_{i=1}^{4}\sum_{j=1}^{3} ij = \sum_{i=1}^{4}(i + 2i + 3i)$$
$$= \sum_{i=1}^{4} 6i$$

Definisi rekursif

- **Definisi yang menggunakan "diri sendiri" dalam ukuran yang lebih kecil (definisi rekursif), dan penjelasan eksplisit untuk nilai awal (nilai basis).**
- **Contoh**: definisi rekursif himpunan Ekspresi Aritmatika EA
- Basis: $1, 2, 3, 4, 5 \in EA$
- Rekursif: jika $a \in EA$ dan $b \in EA$, maka
  - $a + b \in EA$
  - $a - b \in EA$
  - $a \times b \in EA$
  - $a \div b \in EA$

Fungsi Rekursif

Fungsi yang dinyatakan dengan "diri sendiri" dalam ukuran yang lebih kecil (secara rekursif), dan nilai eksplisit untuk nilai(-nilai) basis.

Contoh: fungsi Fibonacci

Basis: $fib(0) = 0;\ fib(1) = 1$

Rekursif: $fib(n) = fib(n-1) + fib(n-2)$

Ditulis dengan cara lain:

$$fib(n) = \begin{cases} n & \text{jika } n = 0, 1 \\ fib(n-1) + fib(n-2) & \text{jika } n > 1 \end{cases}$$

Algoritma Rekursif

Sub-bab 3.5

```
Procedure X ( a: non-zero real,
                    n: non-negative integer );
    if n = 0 then
        X(a, n) := 1
    else
        X(a, n) := a * X(a, n – 1)
```

```
Procedure Y( n: non-negative integer );
    if n = 0 then
        Y(n) := 1
    else
        Y(n) := n * Y(n – 1)
```

Procedure power (a: non-zero real,
n: non-negative integer);

```
if n = 0 then
    power (a, n) := 1
else
    power (a, n) := a * power (a, n – 1)
```

Procedure factorial (n: non-negative integer);

```
if n = 0 then
    factorial (n) := 1
else
    factorial (n) := n * factorial (n – 1)
```

Procedure binary-search (x: key, i, j: index);

```
m = ⌊ (i +j) / 2 ⌋
if    x = a_m    then
      location := m
else
      if  (x < a_m   and   i < m) then
            binary-search (x, i, m – 1)
else
      if  (x > a_m   and   j > m) then
            binary-search (x, m + 1, j)
else
      location := 0
```

Procedure merge-sort ($L = a_1, ..., a_n$);

if $n > 1$ then

$m = \lfloor n / 2 \rfloor$

$L_1 := a_1, a_2, ..., a_m$

$L_2 := a_{m+1}, a_{m+2}, ..., a_n$

L := merge (merge-sort(L_1), merge-sort(L_2))

Procedure merge (L_1, L_2 : list);

L := empty list

while L_1 and L_2 are both non-empty

begin

remove smaller of 1st element of L_1 and L_2 from the list it is in

and put it at the right end of L

if removal of this element makes one list empty then

remove all elements from the other list and append them to L

end

8	2	4	6	9	7	10	1
1	2	4	6	7	8	9	10

P.R.

5.4. 10, 14, 15 (programming)

Email: arya.wijaya@gmail.com

Subject:

MATDIS <kelas> Tugas Programming 01 – <NRP> – <Nama>

Nama file zip = nama Subject

Paling lambat Kamis depan 08.30