Latihan Soal: Analisis Rekurens dalam Konteks Prosedural

IF2110 – Algoritma dan Struktur Data
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

Latihan-1

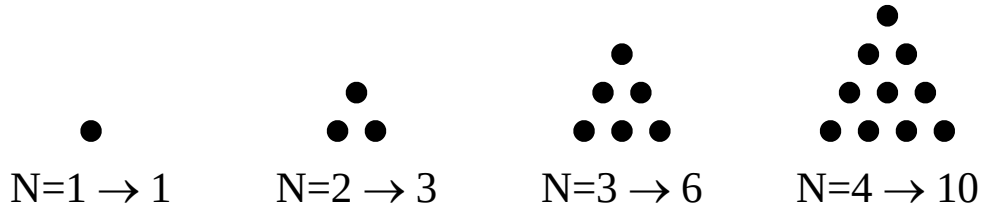
a. Deret Fibonacci:

   0, 1, 1, 2, 3, 5, … , Fib(N-1) + Fib(N-2)

   Buatlah fungsi untuk menghitung bilangan Fibonacci (N)

b. Deret Segitiga:

   Segitiga(N) = N + Segitiga(N-1)



   Buatlah fungsi untuk menghitung bilangan deret segitiga pada urutan N.

Latihan-2

Dengan memanfaatkan kamus global seperti pada slide 20, buatlah fungsi/prosedur sebagai berikut secara rekursif:

- Fungsi yang menghasilkan **nilai ekstrem** (minimum atau maksimum) pada list.
- Prosedur untuk **mencari suatu nilai x dalam list**, menghasilkan indeks di mana X ditemukan (bernilai idxUndef jika tidak ditemukan) dan found (true jika ditemukan, false jika tidak).
- Prosedur untuk **menambahkan elemen x di awal list** sehingga menggeser semua elemen list dan list.nEff bertambah 1. Kasus khusus jika list.nEff = idxMax, x tidak ditambahkan karena sudah tidak ada tempat.

Studi Kasus Pemrosesan List secara Rekursif

List dapat diproses secara rekursif dengan memperhatikan rentang indeks elemen yang diproses

```
constant capacity: integer = 100
constant idxUndef: integer = -999

type  ElType: integer { elemen list }
type  List: < contents: array [0..capacity-1] of ElType,
                    { memori tempat penyimpan elemen (container) }
              nEff: integer ≥ 0 { banyaknya elemen efektif } >
{ Contoh penggunaan:
  Deklarasi variabel: list: List
  Pengaksesan elemen type: list.contents[i]
                           list.nEff
  List kosong adalah List dengan list.Neff = 0 }
```

Latihan-2 (cont.)

```
function max (l: List, startIdx: integer) → integer
{ Menghasilkan nilai maksimum dari list l }
{ Prekondisi: l tidak kosong }
{ Definisi rekursif pencarian nilai maksimum: }
{    Basis: startIdx = l.nEff-1: max = l.contents[startIdx]
  Rekurens: startIdx < l.nEff-1: max = max2(l.contents[startIdx],max(l,startIdx+1)) }

procedure search (input x: integer, input l: List, input startIdx: integer,
                  output idx: integer, output found: boolean)
{ I.S: x, l terdefinisi, startIdx ≥ 0 }
{ F.S: idx adalah nilai ditemukannya x di l pada interval [startIdx..l.nEff-1],
       found = true jika ditemukan }

procedure addX (input/output l: List, input x: integer)
{ I.S: l, x terdefinisi }
{ F.S: Jika l.nEff < capacity, maka x ditambahkan pada indeks 0,
       elemen lain digeser, l.nEff bertambah (panggil prosedur addXRec);
       Kasus khusus: jika l.nEff = capacity maka x tidak ditambahkan }
```