Recursion

Erick Pranata

Edisi I
Maret 2013

Definisi

Bayangkan definisi suatu frase yang bersifat sirkular

```
Status Galau: Kondisi galau yang dicerminkan dalam bentuk tulisan
```

Perhatikan bahwa *galau* kembali digunakan untuk menerangkan *status galau*. Definisi tersebut menggunakan mengalihfungsikan kata-kata yang seharusnya dijelaskan, menjadi penjelas. Inilah yang disebut *recursion*.

Dalam dunia pemrograman, recursion tergolong dalam rumpun iteration (repetition, terdiri atas iteration dan recursion). Dengan demikian, struktur ini dapat digunakan untuk menjalankan *statement* yang berulang.

Thinking Recursively

Berpikir rekursif dapat dilakukan dengan:

1. memecah permasalahan menjadi masalah-masalah yang lebih kecil
2. menentukan pola umum yang digunakan untuk memecahkan masalah-masalah tersebut
3. menyatukannya untuk dapat menyelesaikan permasalahan secara utuh

Sebagai contoh, andaikata terdapat sebuah fungsi untuk menghitung total kuadrat dari bilangan *m* sampai *n*, SumSquares(m,n), dengan syarat $m<=n$, secara iteratif ia dapat dinyatakan sebagai berikut:

Code 1. SumSquares secara Iteratif

```
function SumSquares(m, n)
    total = 0
    for i = m to n
        total = total + i*i
    next
    SumSquares = total
end function
```

Dengan demikian, nilai dari SumSquares (5, 10) adalah = $5^2+6^2+7^2+8^2+9^2+10^2$ = 255.

Menyelesaikan problem ini secara rekursif dapat dilakukan dengan memecah masalah tersebut menjadi masalah masalah yang lebih kecil:

$$
\begin{aligned}
\texttt{SumSquares(5, 10)} &= 5^2 + 6^2 + 7^2 + 8^2 + 9^2 + 10^2 \\
\texttt{SumSquares(6, 10)} &= 6^2 + 7^2 + 8^2 + 9^2 + 10^2 \\
\texttt{SumSquares(7, 10)} &= 7^2 + 8^2 + 9^2 + 10^2 \\
\texttt{SumSquares(8, 10)} &= 8^2 + 9^2 + 10^2 \\
\texttt{SumSquares(9, 10)} &= 9^2 + 10^2 \\
\texttt{SumSquares(10, 10)} &= 10^2
\end{aligned}
$$

Perhatikan ilustrasi di atas. Bukankah, masalah-masalah tersebut dapat ditulis sebagai berikut?

$$
\begin{aligned}
\texttt{SumSquares(5, 10)} &= 5^2 + \texttt{SumSquares(6, 10)} \\
\texttt{SumSquares(6, 10)} &= 6^2 + \texttt{SumSquares(7, 10)} \\
\texttt{SumSquares(7, 10)} &= 7^2 + \texttt{SumSquares(8, 10)} \\
\texttt{SumSquares(8, 10)} &= 8^2 + \texttt{SumSquares(9, 10)} \\
\texttt{SumSquares(9, 10)} &= 9^2 + \texttt{SumSquares(10, 10)} \\
\texttt{SumSquares(10, 10)} &= 10^2
\end{aligned}
$$

Dan jika digeneralisasi, bukankah akan menjadi berikut?

$$
\begin{aligned}
\texttt{SumSquares(m, n)} &= m^2 + \texttt{SumSquares(m+1, n)} \\
\texttt{SumSquares(m, n)} &= 10^2 \quad \texttt{<- Jika m=n}
\end{aligned}
$$

Masalah tersebut ternyata hanya menjadi 2 pola saja! Perhatikan bahwa perulangan berhenti ketika nilai *m=n*. Dengan demikian bila digabungkan, fungsi tersebut dapat ditulis secara rekursif sebagai berikut:

Code 2. SumSquares secara Rekursif

```
function SumSquares(m, n)
    if m=n then
        SumSquares = m * m                    'Base Case
    else
        SumSquares = m * m + SumSquares(m+1, n)  'Recursive Case
    end if
end function
```

Beberapa contoh kasus lain yang dapat digunakan untuk mempelajari recursion adalah:

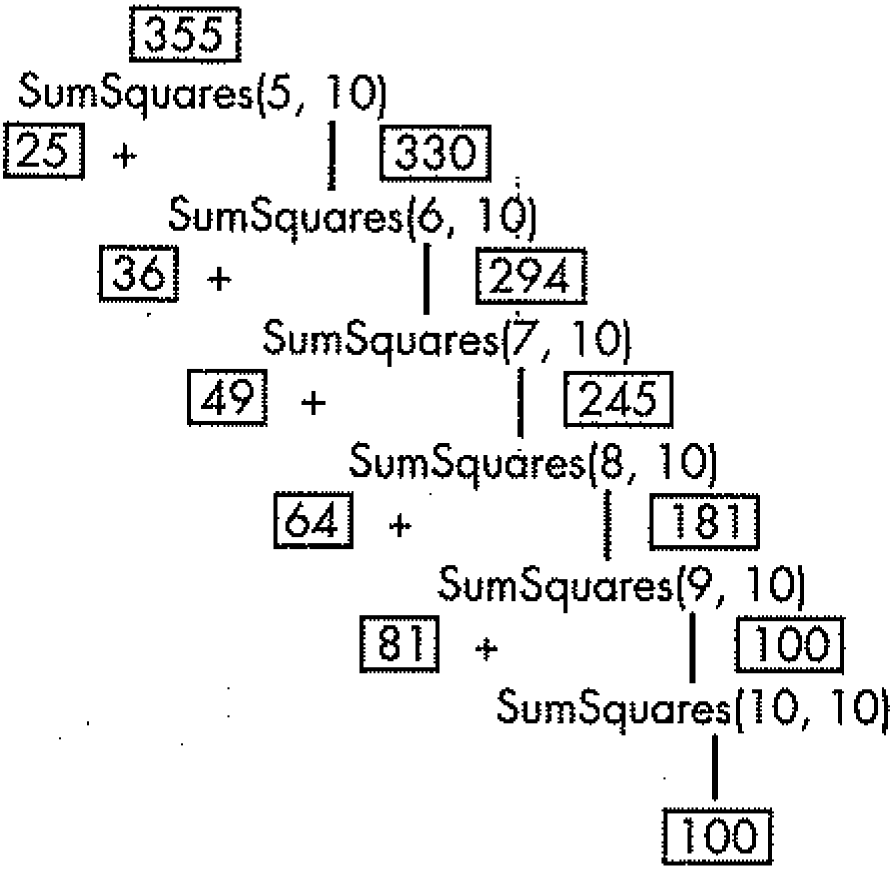
1. Faktorial
2. Fibonacci
3. Perkalian
4. Pangkat

Tracing

Tracing, atau pengkajian suatu fungsi atau prosedur rekursif dapat dilakukan dengan 2 cara:

1. Call Tree
2. Call Trace

Ambil contoh code 2. Semisal Anda ingin memeriksa apakah fungsi tersebut telah berjalan dengan benar, lakukan trace dengan menggunakan call tree sebagai berikut



Gambar 1
Call Tree

atau call trace sebagai berikut

$$
\begin{aligned}
\text{SumSquares}(5,10) &= (25 + \text{SumSquares}(6,10)) \\
&= (25 + (36 + \text{SumSquares}(7,10))) \\
&= (25 + (36 + (49 + \text{SumSquares}(8,10)))) \\
&= (25 + (36 + (49 + (64 + \text{SumSquares}(9,10))))) \\
&= (25 + (36 + (49 + (64 + (81 + \text{SumSquares}(10,10)))))) \\
&= (25 + (36 + (49 + (64 + (81 + 100))))) \\
&= (25 + (36 + (49 + (64 + 181)))) \\
&= (25 + (36 + (49 + 245))) \\
&= (25 + (36 + 294)) \\
&= (25 + 330) \\
&= 355
\end{aligned}
$$

Gambar 2
Call Trace

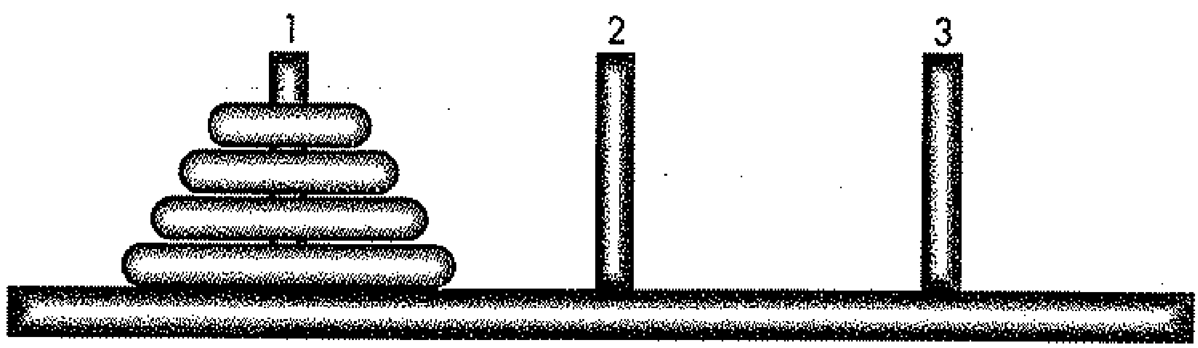
Tower of Hanoi

Alkisah di suatu daerah di Asia, terdapat sejumlah biarawan yang berusaha memindahkan kepingan emas. Konon, ketika mereka selesai memindahkan ke 64 kepingan tersebut dari pilar 1 ke pilar 3, maka dunia akan hancur dan kembali ke masa awal, ketika dunia baru diciptakan. Ke 64 keping memiliki ukuran yang berbeda, dan para biarawan harus mematuhi 2 aturan:

1. Hanya 1 keping yang dapat dipindahkan pada suatu waktu
2. Keping yang lebih besar tidak boleh diletakkan di atas keping yang lebih kecil

Jika mereka bekerja nonstop 24/7, dan butuh 1 detik untuk memindahkan sebuah keping, kapankah dunia akan hancur?

Ilustrasi permasalahan ini untuk 4 keping ditunjukkan pada gambar 3.



Gambar 3
Tower of Hanoi

Solusi untuk permasalahan ini dapat dicapai dengan langkah-langkah sebagai berikut:

1. Pindahkan 4 keping dari pilar 1 ke pilar 3
    a. Pindahkan 3 keping dari pilar 1 ke pilar 2
        i. Pindahkan 2 keping dari pilar 1 ke pilar 3
            1. Pindahkan 1 keping dari pilar 1 ke pilar 2
                a. Pindahkan sebuah keping dari pilar 1 ke pilar 2
            2. Pindahkan sebuah keping dari pilar 1 ke pilar 3
            3. Pindahkan 1 keping, dari pilar 2 ke pilar 3
                a. Pindahkan sebuah keping dari pilar 2 ke pilar 3
        ii. Pindahkan sebuah keping dari pilar 1 ke pilar 2
        iii. Pindahkan 2 keping, dari pilar 3 ke pilar 2
            1. dst…
    b. Pindahkan sebuah keping dari pilar 1 ke pilar 3
    c. Pindahkan 3 keping, dari pilar 2 ke pilar 3
        i. Pindahkan 2 keping dari pilar 2 ke pilar 1
            1. dst…
        ii. Pindahkan sebuah keping dari pilar 2 ke pilar 3
        iii. Pindahkan 2 keping, dari pilar 1 ke pilar 3
            1. dst…

Memperhatikan urutan langkah di atas, dapat diketahui bahwa base case yang cocok adalah “Pindahkan sebuah keping dari pilar *start* ke pilar *tujuan*”. Perhatikan poin a, b, dan c di atas; Prosedur rekursif untuk mencetak solusi untuk Tower of Hanoi dapat ditulis demikian

Code 3. Tower of Hanoi

```
Sub ToH(n, start, tujuan, sementara)
    If n = 1 then
        document.write " Pindahkan sebuah keping dari pilar " &
        start & " ke pilar " & tujuan & "<br/>"
    else
        ToH(n-1, start, perantara, tujuan)

        document.write " Pindahkan sebuah keping dari pilar " &
        start & " ke pilar " & tujuan & "<br/>"

        ToH(n-1, perantara, tujuan, start)

    End if
End Sub
```

Coba lakukan tracing. Benarkah prosedur tersebut?

Jika setiap aksi dihitung, berikut hasil yang diperoleh untuk tiap keping:

Tabel 1
Jumlah Aksi berdasarkan Jumlah Keping

Jumlah Keping	Jumlah Aksi
1	1
2	3
3	7
4	15
5	31
6	63
7	127
8	255

Dengan demikian dapat disimpulkan bahwa jumlah instruksi bersifat eksponensial, dimana bila terdapat *n* keping, maka akan terdapat 2^n-1 aksi.

Kembali ke cerita para biarawan; Terdapat 64 keping emas dan untuk memindahkan 1 keping, diperlukan 1 detik. Jika dalam 1 tahun, terdapat 31.536.000 detik, maka waktu yang diperlukan bagi para biarawan untuk memindahkan seluruh keping tersebut adalah (2^{64}-1)/ 31.536.000 = 584.942.417.355 tahun. Jadi, kita masih akan tetap hidup untuk beberapa saat...

Referensi

Thomas A. Standish, Data Structures, Algorithms & Software Principles in C, Addison-Wesley Publishing Company, 1995.