

A. Algoritma Rekursif

Sebelum masuk ke dalam materi TREE, sebagai pengantar kita akan mempejari terlebih dahulu terkait algoritma rekursif. Contoh ilustrasi rekursif:



Gambar 1. Rekursi dalam sebuah gambar yang berisi lukisan seseorang yang sedang melukis dirinya sendiri yang sedang melukis (secara berulang). Ref: <http://pattern-blog.com.ua/article/ponimanie-i-primeneniye-rekursii-v-css>

Rekursif secara sederhana dapat diartikan sebagai cara menyelesaikan suatu masalah dengan cara menyelesaikan submasalah yang identik dari masalah utama. Misalnya contoh kasusnya adalah membagi kertas ke dalam delapan bagian sama besar. Pada permasalahan ini terdapat 2 cara penyelesaian:

1. Menggunakan perulangan, di mana kertas cukup dibagi menjadi delapan bagian secara membujur seperti gambar berikut ini.

➔

1

2

3

4

5

6

7

8

2. Menggunakan rekursif, dengan cara ini, kertas dibagi dua terlebih dahulu menjadi potongan A dan B, kemudian untuk masing-masing potongan di potong kembali menjadi dua bagian. A menjadi A1 dan A2, B menjadi B1 dan B2. Selanjutnya dari 4 potongan yang dihasilkan, masing-masingnya dipotong kembali menjadi dua bagian sama besar yaitu A1a, A1b, A2a, A2b, B1a, B1b, B2a, dan B2b.

➔

A

B

A1a

A1b

➔

A1

➔

B1

➔

B1a

B1b

A2a

A2b

➔

A2

➔

B2

➔

B2a

B2b

Adakah yang pernah menonton film **Inception** yang dibintangi oleh Leonardo Dicaprio dan Ken Watanabe. The movie tells about someone who can dream inside a dream. If you think that the “dream” in the movie is a function, then you’ll find it really similar concept to the recursive function.

Induksi Matematika

Rekursif sangat erat kaitannya dengan induksi matematika. Contohnya pada perhitungan pangkat dan juga nilai faktorial.

1. Dua Pangkat n

$2^0 = 1$

$2^n = 2 \times 2^{n-1}$, misalnya n = 10, maka $2^{10} = 2 \times 2^9$.

2. Faktorial

$0! = 1$

$A! = A \times (A - 1)!$, misalnya A = 5, maka $5! = 5 \times 4!$

TREE (Rekursif)

Struktur Data – CCH1A4

Rekursif dan Pemrograman

Umumnya rekursif menggantikan peran perulangan, dengan memanfaatkan proses induksi matematika pada suatu subprogram (function atau procedure). Oleh karena itu biasanya tidak ada instruksi perulangan (while/for) di dalam algoritma rekursif.

Misalnya diberikan sebuah subprogram berikut ini!

```
function f (x : integer) → integer
kamus
    hasil : integer
algoritma
    hasil ← f(x+1) + 5
    → hasil
Endfunction
```

Apabila subprogram tersebut dijalankan atau dipanggil dengan nilai x adalah 5, berapah hasil atau nilai yang dikembalikan dari function tersebut?

Hasilnya ??? function akan terus berjalan tiada henti, dalam perulangan dikenal dengan looping forever. Berikut ini adalah penjelasannya!

1. Fungsi f dipanggil dengan nilai x = 5, kemudian di dalam badan function hasil $\leftarrow f(6) + 5$, artinya kita masuk kedalam function f kembali dengan nilai X = 6.
2. Fungsi f dipanggil dengan nilai x = 6, kemudian di dalam badan function hasil $\leftarrow f(7) + 5$, artinya kita masuk kedalam function f kembali dengan nilai X = 7.
3. Fungsi f dipanggil dengan nilai x = 7, kemudian di dalam badan function hasil $\leftarrow f(8) + 5$, artinya kita masuk kedalam function f kembali dengan nilai X = 8.
4. ...
5. ...

Terlihat pada penjelasan di atas, bahwa fungsi f tidak akan pernah berhenti. Di sini kita bisa melihat bahwa seolah-olah terjadi perulangan dengan nilai x yang berubah-ubah dari 5, 6, 7, 8 hingga seterusnya. Konsep ini lah yang digunakan dalam rekursif untuk menggantikan peran loop seperti for dan while.

Masalah selanjutnya adalah bagaimana cara membuat proses rekursif ini berhenti. Caranya adalah dengan menggunakan percabangan atau if-then. Perhatikan modifikasi fungsi f sebelumnya.

```
function f (x : integer) → integer
kamus
    hasil : integer
algoritma
    if x == 7 then
        hasil ← 2
    else
        hasil ← f(x+1) + 5
    endif
    → hasil
Endfunction
```

Dengan menambahkan if-then pada function di atas maka proses rekursif dapat berhenti ketika x bernilai 7. Sehingga apabila function f dipanggil dengan nilai x = 5. Maka penjelasannya adalah sebagai berikut ini.

1. Fungsi f dipanggil dengan nilai x = 5, kemudian di dalam badan function hasil $\leftarrow f(6) + 5$, artinya kita masuk kedalam function f kembali dengan nilai X = 6.
2. Fungsi f dipanggil dengan nilai x = 6, kemudian di dalam badan function hasil $\leftarrow f(7) + 5$, artinya kita masuk kedalam function f kembali dengan nilai X = 7.
3. Fungsi f dipanggil dengan nilai x = 7, kemudian di dalam badan function diperoleh hasil $\leftarrow 2$.
4. Sehingga nilai hasil $\leftarrow f(7) + 5$ pada saat x bernilai 6 adalah hasil $\leftarrow 2 + 5$, atau hasil $\leftarrow 7$. Selanjutnya nilai hasil $\leftarrow f(6) + 5$ pada saat x bernilai 5 adalah hasil $\leftarrow 7 + 5$, atau hasil $\leftarrow 13$.
5. Terakhir. Hasil dari f(5) adalah 13.

KOMPONEN ALGORITMA REKURSIF

Algoritma rekursif terdiri dari dua komponen utama:

- 1. **Basis** atau **base case**, yaitu bagian untuk menghentikan proses rekursif. Oleh karena itu **HAL TERPENTING** dalam algoritma rekursif adalah **MENENTUKAN BASE CASE** dari algoritma rekursif.
- 2. **Induksi**, yaitu bagian pemanggilan subprogramnya

Problem	Dua Pangkat	Faktorial
Base case	$2^0 = 1$	$0! = 1$
Induksi	$2^n = 2 \times 2^{n-1}$	$A! = A \times (A - 1)!$
Function Rekursif	<u>function</u> power(n: <u>integer</u>) → <u>integer</u> <u>algoritma</u> <u>if</u> n == 0 <u>then</u> → 1 <u>else</u> → 2 * power(n-1) <u>endif</u> <u>endfunction</u>	<u>function</u> faktorial(A: <u>integer</u>)→ <u>integer</u> <u>algoritma</u> <u>if</u> A == 0 <u>then</u> → 1 <u>else</u> → A * faktorial(A-1) <u>endif</u> <u>endfunction</u>

DERET FIBONANCI

Sebuah deret dengan nilai suku ke-0 dan ke-1 adalah 0 dan 1, dan nilai suku ke-n selanjutnya adalah hasil penjumlahan dua suku sebelumnya. Secara umum dapat diformulasikan $S_n = S_{n-1} + S_{n-2}$. Berikut ini adalah contoh nilai deret fibonanci hingga suku ke-10.

n	0	1	2	3	4	5	6	7	8	9	10
S_n	0	1	1	2	3	5	8	13	21	34	55

Pada kasus ini kita tahu bahwa **nilai suku ke-0 dan ke-1 telah terdefinisi sejak awal, sehingga hal ini bisa menjadi base case untuk algoritma fibonanci**, sedangkan untuk kondisi rekursif atau **induksinya adalah nilai faktorial suku ke-n adalah hasil penjumlahan nilai faktorial pada dua suku sebelumnya**. Dengan demikian dapat dibuat fungsi rekursif untuk mencari nilai fibonanci pada suku ke-n adalah sebagai berikut ini.

```
Function fibo(n:integer) → integer  
Algoritma  
    If n == 0 OR n == 1 then  
        → n  
    else  
        → fibo(n-1) + fibo(n-2)  
    endif  
endfunction
```

KESIMPULAN

Ada beberapa konsep penting yang perlu diperhatikan dalam membuat sebuah algoritma rekursif.

- 1. Tidak semua problem dalam pemrograman bisa dengan mudah dirubah ke dalam bentuk rekursif.
- 2. Proses rekursif terbentuk dengan cara pemanggilang suatu subprogram di dalam suprogram itu sendiri, sehingga seolah-olah terjadi proses perulangan.
- 3. Definisikan kondisi base case dari algoritma untuk memastikan bahwa proses rekursif yang akan dibuat bisa berhenti.
- 4. Secara umum algoritma rekursif menggunakan percabangan (IF THEN) untuk menentukan base case atau induksi yang akan di proses.
- 5. Tidak lazim dijumpai perulangan seperti for/while dan repeat di dalam algoritma rekursif.

SOAL-SOAL

1. **Palindrom**, menentukan suatu Array adalah Palindrom atau tidak. Contoh palindrom : KATAK, ADA, KAKAK, tamat, Ibu Ratna antaR ubl.

Perulangan	Rekursif
<p>function isPalindrom(S : text, N : integer)→boolean {Mengembalikan TRUE apabila array S yang berisi N char tersusun palindrom, atau FALSE apabila sebaliknya, asumsi text = array [0..N-1] of char }</p> <p>kamus i,j:integer status:boolean</p> <p>algoritma status ← true i←0 ; j←N-1 while i <= j and status do status ← S[i] == S[j] i++;j++ endwhile → status</p> <p>Endfunction</p> <p>Penjelasan: Loop berhenti ketika i > j, maka ini akan menjadi kondisi base case 1. Loop berhenti ketika status == false, maka ini akan menjadi kondisi base case 2.</p>	<p>function palindrom(S:text, i,j:integer)→boolean // VERSI 1</p> <p>kamus status : boolean</p> <p>algoritma if i > j then // BASE CASE 1 → true else status ← S[i] == S[j] if NOT status then // BASE CASE 2 → status else // INDUKSI → palindrom(S,i+1,j-1) endif endfunction</p> <p>function palindrom(S:text, i,j:integer)→boolean // VERSI 2</p> <p>kamus status : boolean</p> <p>algoritma if i > j then // BASE CASE 1 → true else // INDUKSI status ← S[i] == S[j] → palindrom(S,i+1,j-1) AND status endif endfunction</p>
<p>Contoh Pemanggilan: kalimat ← {'K','A','T','A','K'} output(isPalindrom(Kalimat,5))</p>	<p>Contoh Pemanggilan: kalimat ← {'K','A','T','A','K'} output(isPalindrom(Kalimat,0,4))</p>

2. **Binary Search**, Menentukan algoritma rekursif untuk binary Search.

Perulangan	Rekursif
<p>function biSearch(T:tabInt, X, N : integer)→integer {Mengembalikan INDEKS apabila nilai X berada di dalam array T yang berisi N elemen array, atau -1 sebaliknya, asumsi tabInt = array [0..N-1] of integer, dan array T terurut ASCENDING }</p> <p>kamus status, tengah,kiri,kanan:integer</p> <p>algoritma kiri ← 0 kanan ← N-1 status ← -1 // asumsi tidak ketemua while kiri <= kanan and status == -1 do tengah ← (kiri + kanan) div 2 if X > T[tengah] then kiri ← tengah + 1 else if X < T[tengah] then kanan ← tengah - 1 else status ← tengah endif endwhile → status</p> <p>endfunction</p> <p>Penjelasan: Loop berhenti ketika kiri > kanan, maka ini akan menjadi kondisi base case 1. Loop berhenti ketika status == true, maka ini akan menjadi kondisi base case 2</p>	<p>function biSearch(T:tabInt, X, kiri, kanan : integer)→ integer</p> <p>kamus tengah:integer</p> <p>algoritma tengah ← (kiri + kanan) div 2 if kiri > kanan then // BASE CASE 1 → -1 else if X == T[tengah] then // BASE CASE 2 → tengah else if X > T[tengah] then // INDUKSI 1 → biSearch(T,X,tengah+1, kanan) else // X < T[tengah] // INDUKSI 2 → biSearch(T,X,kiri, tengah-1) endif Endfunction</p>
<p>Contoh Pemanggilan: T ← {'12','30','32','55','65','73','99'} N ← 7 output(biSearch(T,N,5))</p>	<p>Contoh Pemanggilan: T ← {'12','30','32','55','65','73','99'} N ← 7 output(biSearch(T,5,0,N-1))</p>

3. **Count Zero**, Menghitung jumlah kemunculan 0 atau Zero pada satu Array.

Perulangan	Rekursif
<p>function countZero(T:tabInt, N : integer) → integer {Mengembalikan jumlah kemunculan 0 pada array T yang berisi N elemen nilai } kamus i, count:integer algoritma i ← N-1 count ← 0 while i >= 0 do if T[i] == 0 then count++ endif i-- endwhile → count endfunction</p> <p>Penjelasan: Loop berhenti ketika i < N, maka ini akan menjadi kondisi base case.</p>	<p>function countZero(T:tabInt, N : integer) → integer kamus count : integer algoritma if N < 0 then // BASE CASE → 0 else // INDUKSI count ← 0 if T[N] == 0 then count ← 1 endif → count + countZero(T,N-1) endif Endfunction</p>
<p><u>Contoh Pemanggilan:</u> T ← {'12', '30', '32', '55', '65', '73', '99'} N ← 7 output(countZero(T,N))</p>	<p><u>Contoh Pemanggilan:</u> T ← {'12', '30', '32', '55', '65', '73', '99'} N ← 7 output(countZero(T,N-1))</p>

4. **Reverse an Array**, Membalikkan susunan nilai dari Array.

Perulangan	Rekursif
<p>Procedure Reverse(in/out T:tabInt, input N : integer) {IS. terdefinisi sebuah array T yang berisi N nilai FS. melakukan reverse susunan array T} kamus i, j, temp:integer algoritma j ← N-1 i ← 0 while i <= j do temp ← T[i] T[i] ← T[j] T[j] ← temp i++ ; j-- endwhile endfunction</p> <p>Penjelasan: Loop berhenti ketika i > j, maka ini akan menjadi kondisi base case.</p>	<p>Procedure Reverse(in/out T:tabInt, input i, j : integer) kamus temp : integer algoritma // BASE CASE i > j dan tidak ada aksi if i <= j then // INDUKSI temp ← T[i] T[i] ← T[j] T[j] ← temp Reverse(T, i+1, j-1) endif endfunction</p>
<p><u>Contoh Pemanggilan:</u> T ← {'12', '30', '32', '55', '65', '73', '99'} N ← 7 Reverse (T,N)</p>	<p><u>Contoh Pemanggilan:</u> T ← {'12', '30', '32', '55', '65', '73', '99'} N ← 7 Reverse (T,0,N-1)</p>

5. **Drawing Triangle**, Menampilkan character * dari 1 hingga ke N seperti contoh berikut (N = 5)

*
**

Perulangan	Rekursif
<p>Procedure Triangle(<u>input</u> N : <u>integer</u>) {IS. terdefinisi sebuah bilangan bulat N FS. menampilkan segitiga dengan pola * dari 1 hingga ke N seperti pada contoh} kamus i : <u>integer</u> temp : <u>string</u> algoritma i ← N temp ← "" while i >= 1 do temp ← temp + "*" output(temp) i-- endwhile endfunction</p> <p>Penjelasan: Loop berhenti ketika i = 0, maka ini akan menjadi kondisi base case.</p>	<p>function Triangle(<u>input</u> N : <u>integer</u>) → <u>string</u> kamus temp : <u>string</u> algoritma if N == 0 then // BASE CASE → "" else // INDUKSI temp ← "*" + Triangle(N-1) output(temp) → temp endif endfunction</p>
<p><u>Contoh Pemanggilan:</u> input(N) Triangle(N)</p>	<p><u>Contoh Pemanggilan:</u> input(N) Triangle(N)</p>