

# ALGORITMA DAN PEMROGRAMAN II

## REKURSIF

Rosa Ariani Sukamto

# ROSA ARIANI SUKAMTO

**Blog:** <http://hariiniadalahhadiah.wordpress.com>

**Facebook:** <https://www.facebook.com/rosa.ariani.sukamto>

**Email:** [rosa\\_if\\_itb\\_01@yahoo.com](mailto:rosa_if_itb_01@yahoo.com)

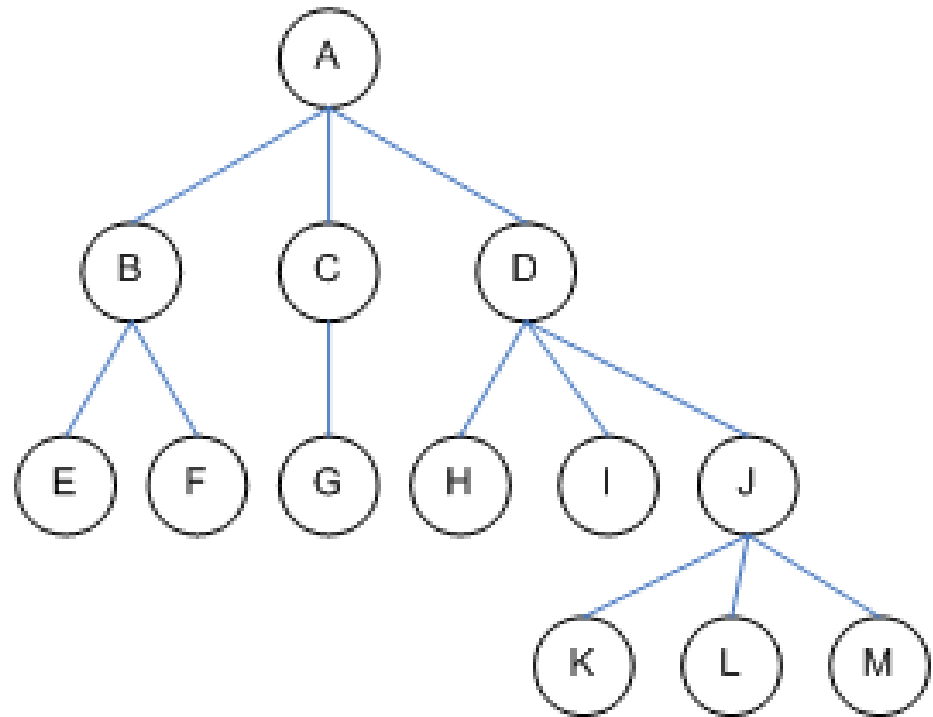
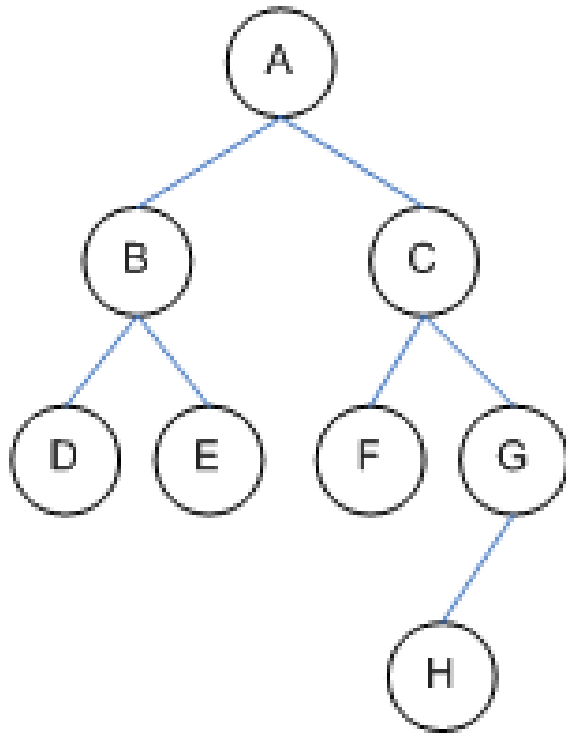


# REKURSIF

- mekanisme memanggil dirinya sendiri
- rekursif melibatkan sebuah prosedur atau fungsi yang memanggil dirinya sendiri untuk mengulangi proses
- rekursif lebih sering digunakan untuk kasus-kasus yang tidak dapat dilakukan dengan pengulangan



# KASUS POHON



# SYARAT REKURSIF

- **basis**
  - merupakan kondisi berhenti dari proses rekursif
- **rekursif**
  - kode program yang menyatakan rekursif atau pemanggilan kembali dirinya sendiri



# CONTOH REKURSIF

```
#include <stdio.h>
```

```
void tulis(int n){  
    int counter;  
    counter = n;
```

```
    if(counter > 0){  
        printf("proses rekursif dengan nilai counter: %d\n",  
            counter);  
        counter = counter - 1;  
        tulis(counter);  
    }  
}
```

```
int main(){  
    tulis(10);  
    return 0;  
}
```

jika sudah tidak  
memenuhi syarat  
if, maka rekursif  
akan berhenti  
(**basis**)

**rekursif**

# FAKTORIAL REKURSIF

```
#include <stdio.h>

int faktorial(int n){
    printf("n = %d\n", n);
    if((n ==0) || (n == 1)){
        return 1;
    }else{
        return (n * faktorial(n-1));
    }
}

int main(){
    int hasil = faktorial(3);
    printf("hasil faktorial: %d\n", hasil);
    return 0;
}
```

# PENJUMLAHAN (SUM) REKURSIF

```
#include <stdio.h>

int sum(int n){
    printf("n = %d\n", n);
    if(n == 1){
        return 1;
    }else{
        return (n + sum(n-1));
    }
}

int main(){
    int hasil = sum(3);
    printf("hasil sum: %d\n", hasil);
    return 0;
}
```



# FAKTOR PERSEKUTUAN TERBESAR REKURSIF

```
#include <stdio.h>

int CariFPB(int a, int b) {
    if(b==0)
        return a;
    else {
        int temp = a % b;
        return CariFPB(b, temp);
    }
}

int main(){
    int hasil = CariFPB(18, 12);
    printf("hasil fpb: %d\n", hasil);
    return 0;
}
```

# PANGKAT (*POWER*) REKURSIF

```
#include <stdio.h>
```

```
int pow(int x,int y){  
    int hasil;  
  
    if(y==0)  
        return 1;  
    else  
        hasil=x*pow(x,y-1);  
    return hasil;  
}
```

```
int main(){  
    int hasil = pow(2, 3);  
    printf("hasil pangkat: %d\n", hasil);  
    return 0;  
}
```

# FIBBONACI *BOTTOM UP* REKURSIF

```
#include <stdio.h>

void fib(int* arr, int n, int i){//bottom up
    if(i == 0){
        arr[i] = 0;
        fib(arr, n, i+1);
    }else if(i == 1){
        arr[i] = 1;
        fib(arr, n, i+1);
    }else{
        if(i < n){
            arr[i] = arr[i-1] + arr[i-2];
            fib(arr, n, i+1);
        }
    }
}

int main(){
    printf("masukkan bilangan fibonacci ");
    scanf("%d", &n);
    int arr[n+1];

    fib(arr, n, 0);
    int i = 0;
    for(i=0;i<(n+1);i++){
        printf("%d ", arr[i]);
    }
    return 0;
}
```

# FIBBONACI *TOP DOWN* REKURSIF

```
#include <stdio.h>
int fib2(int* arr, int n){//top down
    if(n < 2){
        arr[n] = n;
        return n;
    }else{
        arr[n] = fib2(arr, n-1) + fib2(arr, n-2);
        return arr[n];
    }
}

int main(){
    printf("masukkan bilangan fibonacci ");
    scanf("%d", &n);
    int arr[n+1];

    fib2(arr, n);
    int i = 0;
    for(i=0;i<(n+1);i++){
        printf("%d ", arr[i]);
    }
    return 0;
}
```

# MENARA HANOI REKURSIF

```
#include <stdio.h>

void tower(int n, char awal, char akhir, char antara){
    if(n == 1){
        printf("pindahkan piringan ke %d dari tonggak %c ke tonggak %c\n", n, awal, akhir);
    }else{
        tower((n-1), awal, antara, akhir);
        //memindahkan piringan berikutnya dari awal ke antara
        printf("pindahkan piringan ke %d dari tonggak %c ke tonggak %c\n", n, awal, akhir);
        tower((n-1), antara, akhir, awal);
        //memindahkan piringan berikutnya dari antara ke akhir
    }
}

int main(){
    tower(3, 'A', 'C', 'B');
    return 0;
}
```

# DYNAMIC PROGRAMMING (1)

## Kasus Koin

Diberikan masukan:

- **Uang yang mesti diberikan dengan banyaknya koin paling sedikit**
- **Banyaknya koin yang dapat dijadikan penukar uang**
  - Baris koin yang dapat dijadikan penukar uang
- Misalkan masukan

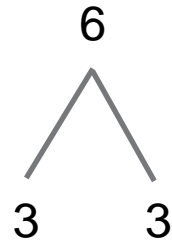
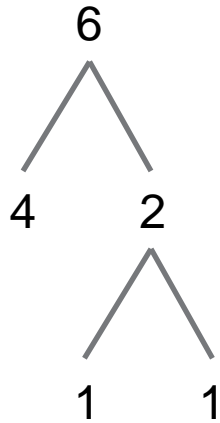
6 3

4

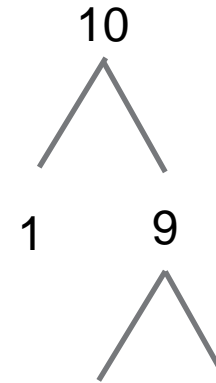
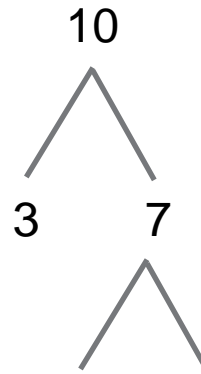
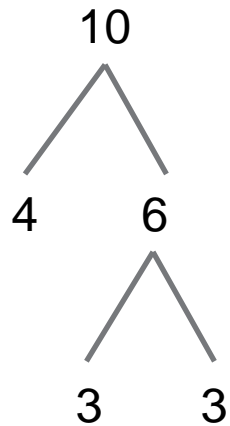
3

1

# DYNAMIC PROGRAMMING (2)



# DYNAMIC PROGRAMMING (3)



$\text{coin}(10)$  = nilai minimal dari

$\text{coin}(6) + 1$  koin (dari gambar di atas)

$\text{coin}(7) + 1$  koin

$\text{coin}(9) + 1$  koin



# DYNAMIC PROGRAMMING (4)

Memo banyaknya koin penukar


0	1	2	3	4	5	6
0	1	2	1	1	2	2
	Koin 1	Koin 1 Koin 1	Koin 3	Koin 4	Koin 4 Koin 1	Koin 3 Koin 3

```
#include <cstdio>
#include <cstdlib>
#include <iostream>
#define inf 1000000
using namespace std;

int n; // duit yang ingin diperoleh
int m; // banyak koin
int c[100]; // nilai koin
int memo[100000];

int coin(int x){
    int& p = memo[x];
    // kalau blom dicari
    if(p == -1) {
        if(x == 0) p = 0; // base case
        else{
            p = inf;
            for(int i=0; i<m; i++) // maximize!
                if(x-c[i] >= 0)
                    p = min(p, coin(x-c[i]) + 1); // recurrence
        }
    }
    return p;
}
```

```
int main(){    // baca input
    scanf("%d %d", &n, &m);
    for(int i=0; i<m; i++)    {
        scanf("%d", &c[i]);
    }
    // init
    memo memset(memo, -1, sizeof(memo));
    // dp!
    printf("%d\n", coin(n));
    return 0;
}
```



# DAFTAR PUSTAKA

S, Rosa A. dan M. Shalahuddin. 2010. Modul Pembelajaran: Algoritma dan Pemrograman. Modula: Bandung.

