

Algoritma Rekursif

Oleh : Agus Priyanto, M.Kom



INSTITUT TEKNOLOGI TELKOM
Smart, Trustworthy, And Teamwork

Tujuan Pembelajaran

- Setelah mengikuti kuliah ini, mahasiswa dapat memahami algoritma rekursif
- Setelah mengikuti kuliah ini, mahasiswa dapat mengimplementasikannya algoritma rekursif dalam studi kasus nyata

Introduction

- **Rekursif** merupakan kemampuan subrutin untuk memanggil dirinya sendiri.
- Adapun suatu subrutin yang memanggil dirinya seperti itu dinamakan **subrutin rekursif**.

■■■ Persoalan-persoalan Rekursif

1. Menghitung Faktorial

Faktorial bilangan asli n adalah perkalian semua bilangan asli yang kurang atau sama dengan n . Faktorial dilambangkan dengan tanda $!$. Jadi jika $n!$, maka dibaca " n faktorial".

$$n! = 1 \times 2 \times \dots \times (n-2) \times (n-1) \times n$$

Untuk faktorial 0, hasilnya adalah 1 $\rightarrow 0! = 1$



Berikut ini adalah faktorial 0 sampai faktorial 10.

$$0! = 1$$

$$1! = 1$$

$$2! = 1 \times 2 = 2$$

$$3! = 1 \times 2 \times 3 = 6$$

$$4! = 1 \times 2 \times 3 \times 4 = 24$$

$$5! = 1 \times 2 \times 3 \times 4 \times 5 = 120$$

$$6! = 1 \times 2 \times 3 \times 4 \times 5 \times 6 = 720$$

$$7! = 1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 = 5040$$

$$8! = 1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 \times 8 = 40320$$

$$9! = 1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 \times 8 \times 9 = 362880$$

$$10! = 1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 \times 8 \times 9 \times 10 = 3628800$$



2. Deret Fibonacci

Deret Fibonacci adalah suatu deret matematika yang berasal dari penjumlahan dua bilangan sebelumnya.

1, 1, 2, 3, 5, 8, 13, 21...

Rumus : $f(n) = f(n-1) + f(n-2)$

$$f(6) = f(6-1) + f(6-2)$$

$$= 5 + 3 = 8$$



Persamaan $F(n)$ dapat dijelaskan sebagai berikut :

- Jika $n=0$, maka $F(0)=0$
- Jika $n=1$, maka $F(1)=1$
- Jika $n>1$ berlaku rumus $F(n-1) + F(n-2)$
- Jika $n=2$, maka $F(2-1) + F(2-2) = F(1) + F(0) = 1 + 0 = 1$
- Jika $n=3$ maka $F(3-1) + F(3-2) = F(2) + F(1) = 1 + 1 = 2$
- Jika $n=4$ maka $F(4-1) + F(4-2) = F(3) + F(2) = 2 + 1 = 3$

Hasil deret bilangan Fibonacci adalah : **0,1,1,2,3, dst**

Contoh Algoritma Rekursif

1. Algoritma untuk menyelesaikan
Pangkat Bilangan

Pseudocode

```
SUBROUTIN pangkat(y,n)
  JIKA n =1 MAKA
    NILAI-BALIK y
  SEBALIKNYA
    NILAI-BALIK y x pangkat(y, n-1)
  AKHIR-JIKA
AKHIR-SUBROUTIN
```


Source code

```
int main()
{
    int y, n;
    long int hasil;

    cout << " Menghitung y pangkat n\n ";
    cout << "y = ";
    cin >> y;
    cout << "n = ";
    cin >> n;

    hasil = pangkat (y, n);
    cout << y << "^" << n << " = " << hasil;
    return 0;
}
```

```
long int pangkat (unsigned int y , unsigned int n)
{
    if (n == 1)
        return y;
    else
        return y * pangkat (y, n-1);
}
```



2. Algoritma untuk menyelesaikan Faktorial

Pseudocode

```
SUBROUTIN faktorial (n)
  JIKA  $n = 0$  ATAU  $1$  MAKA
    NILAI-BALIK  $1$ 
  SEBALIKNYA
    NILAI-BALIK  $n \times \text{faktorial}(n-1)$ 
  AKHIR-JIKA
AKHIR-SUBROUTIN
```

Source code

```
int main()
{
    int n;
    long int hasil;

    cout << " n = " ;
    cin >> n ;
    hasil = faktorial (n)
    cout << n << "! = " << hasil;
    return 0;
}
```

```
long int faktorial (unsigned int n)
{
    if (n == 0 || n == 1)
        return 1;
    else
        return n * faktorial (n-1);
}
```



3. Algoritma untuk menyelesaikan Fungsi Fibonacci

Pseudocode

```
SUBROUTIN fib(n)
  JIKA  $n = 0$  MAKA
    NILAI-BALIK 0
  SEBALIKNYA
    JIKA  $n = 1$  MAKA
      NILAI-BALIK 1
    SEBALIKNYA
      NILAI-BALIK  $\text{fib}(n-1) + \text{fib}(n-2)$ 
    AKHIR-JIKA
  AKHIR-JIKA
AKHIR-SUBROUTIN
```



Source code

```
int main()
{
    int n;
    long int hasil;

    cout << " n = ";
    cin >> n ;
    hasil = fib(n)
    cout << "fib (" << n << ") = " << hasil;
    return 0;
}
```

```
long int fib(unsigned int n)
{
    if (n == 0)
        return 0;
    else
        if (n==1)
            return 1;
        else
            return fib(n-1) + fib(n-2);
}
```

Brute Force

```
int main()
{
    int data[] = { 3, 34, 4, 12, 5, 2 };
    int sum = 100;
    int n = sizeof(data) / sizeof(data[0]);
    if (cekSubsetSum(data, n, sum) == true)
        printf("Ditemukan subset yang sesuai SUM");
    else
        printf("Tidak ada subset yang sesuai SUM");
    return 0;
}
```

```
bool cekSubsetSum(int data[], int n, int sum)
{
    if (sum == 0)
        return true;
    if (n == 0 && sum != 0)
        return false;
    if (data[n - 1] > sum)
        return cekSubsetSum(data, n - 1, sum);

    return cekSubsetSum(data, n - 1, sum) ||
        cekSubsetSum(data, n - 1, sum - data[n - 1]);
}
```

Merge Sort

```
int main()
{
    int arr[] = { 12, 11, 13, 5, 6, 7 };
    int arr_size = sizeof(arr) / sizeof(arr[0]);

    printf("Array belum tersortir \n");
    printArray(arr, arr_size);

    mergeSort(arr, 0, arr_size - 1);

    printf("\nArray sudah tersortir \n");
    printArray(arr, arr_size);
    return 0;
}
```

```
void mergeSort(int arr[], int l, int r)
{
    if (l < r) {
        int m = l + (r - l) / 2;
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);
        merge(arr, l, m, r);
    }
}
```



```
void merge(int arr[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;

    int L[n1], R[n2];

    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];
```

```
    i = 0;
    j = 0;
    k = l;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        }
        else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }
```

```
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}
```





```
void printArray(int A[], int size)
{
    int i;
    for (i = 0; i < size; i++)
        printf("%d ", A[i]);
    printf("\n");
}
```

Quick Sort

```
int main()
{
    int arr[] = {10, 7, 8, 9, 1, 5};
    int n = sizeof(arr)/sizeof(arr[0]);
    quickSort(arr, 0, n-1);
    printf("Sorted array: n");
    printArray(arr, n);
    return 0;
}
```

```
int partition (int arr[], int low, int high)
{
    int pivot = arr[high];
    int i = (low - 1);

    for (int j = low; j <= high- 1; j++)
    {
        if (arr[j] <= pivot)
        {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}
```



```
void swap(int* a, int* b)
{
    int t = *a;
    *a = *b;
    *b = t;
}
```

```
void printArray(int arr[], int size)
{
    int i;
    for (i=0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}
```

```
void quickSort(int arr[], int low, int high)
{
    if (low < high)
    {
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}
```



Terimakasih