Untitled

May 24, 2022

1 Konsep

1.1 Fungsi Rekursif

Rekursif merupakan teknik pemrograman yang memecah masalah menjadi lebih kecil untuk diselesaikan. Jika pada pemmrograman terdapat fungsi rekursif yaitu menyelesaikan masalah terkecil dengan cara memanggil fungsi itu sendiri secara berulang hingga suatu kondisi tertentu. Syarat fungsi rekursif bisa berjalan harus ada base case yang merupakan statement yang digunakan untuk menghentikan fungsi rekursif tersebut, kemudian ada statement yang digunakan untuk membuat permasalahan yang besar menjadi lebih kecil hingga menjadi masalah sederhana. Dan yang terakhir adalah pemanggilan fungsi itu sendiri.

contoh permasalahan yang dapat diselesaikan menggunakan fungsi rekursif :

```
"""
Penyelesaian masalah eksponensial menggunakan rekursif
contoh seperti kode dibawah ini :
"""
def eksponen(a,e) :
    if e == 0 :
        return 1
    else :
        return a*eksponen(a,e-1)

ak = 2
ek = 5

print(eksponen(ak,ek))

"""
Pada fungsi tersebut terdapat base case bila eksponennya bernilai 0 maka fungsi rekursif akan
"""
```

1.2 Tower Of Hanoi

Tower Of Hanoi merupakan permasalahan yang dimana terdapat piringan yang berbentuk menyerupai tower dengan ukuran berbeda-beda. Dan permasalahannya adalah bagaimana cara memindahkan piringan tersebut ke tiang tujuan dengan syarat :

- Piringan harus dipindahkan satu persatu.

- Piringan kecil harus berada di atas piringan besar.

Visualisasinya seperti di bawah ini, di situ terdapat piringan dari tiang 1 dan akan dipindahkan ke tiang 3.

```
"""
    =
   ===
  =====
    |           |           |
 Tiang 1     Tiang 2     Tiang 3
"""
```

Langkah pertama pindahkan piringan kecil ke tiang 3.

```
"""

   ===
  =====                     =
    |           |           |
 Tiang 1     Tiang 2     Tiang 3
"""
```

kemudian pindahkan piringan sedang ke tiang 2.

```
"""


  =====        ===          =
    |           |           |
 Tiang 1     Tiang 2     Tiang 3
"""
```

Setelah itu pindahkan piringan kecil ke tiang 2.

```
"""

                =
  =====        ===
    |           |           |
 Tiang 1     Tiang 2     Tiang 3
"""
```

Sesudah piringan kecil dan sedang berada di tiang 2 kemudian pindahkan piringan besar ke tiang 3.

```
"""

                =
               ===        =====
    |           |           |
 Tiang 1     Tiang 2     Tiang 3
"""
```

Setelah selesai kemudian pindahkan piringan kecil ke Tiang 1.

```
"""


    =           ===         =====
    |           |             |
 Tiang 1     Tiang 2     Tiang 3
"""
```

Pindahkan piringan sedang ke Tiang 3.

```
"""


                            ===
    =                       =====
    |           |             |
 Tiang 1     Tiang 2     Tiang 3
"""
```

Dan langkah terakhir memidahkan piringan kecil ke Tiang 3.

```
"""
                            =
                           ===
                          =====
    |           |             |
 Tiang 1     Tiang 2     Tiang 3
"""
```

[21]:
```
#Contoh penyelesaian menggunakan fungsi rekursif

def towerOfHanoi(n,awal,bantuan,tujuan):
    if n==1:
        print("Piringan - 1 dari-", awal,"ke-",tujuan)
    else:
        towerOfHanoi(n-1,awal,tujuan,bantuan)
        print("Piringan -",n, "dari-",awal,"ke-", tujuan)
        towerOfHanoi(n-1,bantuan,awal,tujuan)

towerOfHanoi(3,'Tiang 1','Tiang 2','Tiang 3')
```

```
Piringan - 1 dari- Tiang 1 ke- Tiang 3
Piringan - 2 dari- Tiang 1 ke- Tiang 2
Piringan - 1 dari- Tiang 3 ke- Tiang 2
Piringan - 3 dari- Tiang 1 ke- Tiang 3
Piringan - 1 dari- Tiang 2 ke- Tiang 1
Piringan - 2 dari- Tiang 2 ke- Tiang 3
Piringan - 1 dari- Tiang 1 ke- Tiang 3
```

2 Implementasi

2.1 Factorial

[7]:
```
def factorial(num) :
    if num == 1 : #Base Case
        return 1
    else :
        print("%d != %d * call factorial( %d )" % (num,num,num-1))
        return num*factorial(num-1) #Persamaan Rekursif

print(factorial(5))
```

```
5 != 5 * call factorial( 4 )
4 != 4 * call factorial( 3 )
3 != 3 * call factorial( 2 )
2 != 2 * call factorial( 1 )
120
```

2.2 Palindrome

[19]:
```
def palindrome(word) :

    lengthWrd = len(word)

    if lengthWrd == 1 or lengthWrd == 0 : #Base Case
        return True

    else :
        print(word)
        wrdRcf  = word[1:-1]
        isMatch = word[0] == word[-1]
        if isMatch == True :
            print("word[0] == word[-1] --> ? %s == %s and call palindrome ( %s␣
 ↪)" % (
                word[0],word[-1],wrdRcf) )
        else :
            print("word[0] == word[-1] --> ? %s != %s and not palindrome" %␣
 ↪(word[0],word[-1]) )

        return isMatch and palindrome(wrdRcf) #Persamaan Rekursif

print(palindrome("tambat"))
```

```
tambat
word[0] == word[-1] --> ? t == t and call palindrome ( amba )
amba
```

```
word[0] == word[-1] --> ? a == a and call palindrome ( mb )
mb
word[0] == word[-1] --> ? m != b and not palindrome
False
```