Exercise 11

May 16, 2022

Buat sebuah program untuk membuat deret fibonacci dari 0 hingga N dengan menggunakan fungsi non-rekursif dan rekursif!

Bandingkan keduanya jika nilai N = 500, Manakah yang lebih baik? Jelaskan!

0.1 without recursion

[5]:
```
# create a function to list fibonacci sequence up to n but without recursion
def fibonacci(n):
    sequence = [0,1]
    for i in range(2,n+1):
        next_num = sequence[-1] + sequence[-2]
        sequence.append(next_num)
    return sequence
print(fibonacci(50))
```

```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584,
4181, 6765, 10946, 17711, 28657, 46368, 75025, 121393, 196418, 317811, 514229,
832040, 1346269, 2178309, 3524578, 5702887, 9227465, 14930352, 24157817,
39088169, 63245986, 102334155, 165580141, 267914296, 433494437, 701408733,
1134903170, 1836311903, 2971215073, 4807526976, 7778742049, 12586269025]
```

[8]:
```
def fibonacci(count):
        fib_list = [0, 1]
        any(map(lambda _: fib_list.append(sum(fib_list[-2:])), range(2, count)))
        return fib_list[:count]
print(fibonacci(50))
```

```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584,
4181, 6765, 10946, 17711, 28657, 46368, 75025, 121393, 196418, 317811, 514229,
832040, 1346269, 2178309, 3524578, 5702887, 9227465, 14930352, 24157817,
39088169, 63245986, 102334155, 165580141, 267914296, 433494437, 701408733,
1134903170, 1836311903, 2971215073, 4807526976, 7778742049]
```

0.2 with recursion

[13]:
```
# create a function to list fibonacci sequence up to n but wih recursion
def fibonacci_rec(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fibonacci_rec(n-1) + fibonacci_rec(n-2)

for i in range(30):
    print(fibonacci_rec(i), end=' ')
```

```
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711
28657 46368 75025 121393 196418 317811 514229
```

[14]:
```
# create a function to list fibonacci sequence up to n but wih recursion lambda␣
 ↪version
fibonacci_rec = lambda n: 0 if n == 0 else 1 if n == 1 else fibonacci_rec(n-1)␣
 ↪+ fibonacci_rec(n-2)
deret_fibo_rec = lambda n: list(map(lambda x: fibonacci_rec(x), range(n)))
print(deret_fibo_rec(30))
```

```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584,
4181, 6765, 10946, 17711, 28657, 46368, 75025, 121393, 196418, 317811, 514229]
```

0.3 Penjelasan

Whenever you are looking for time taken to complete a particular algorithm, it's best you always go for time complexity.

Evaluate the time complexity on the paper in terms of O(something).

Comparing the above two approaches, time complexity of iterative approach is O(n) whereas that of recursive approach is O(2^n).

Let's try to find the time complexity of fib(4)

Iterative approach, the loop evaluates 4 times, so it's time complexity is O(n).

Recursive approach,

```
                          fib(4)

         fib(3)             +              fib(2)

  fib(2)    +    fib(1)          fib(1)     +       fib(0)
```

fib(1) + fib(0)

so fib() is called 9 times which is slightly lower than 2^n when the value of n is large, even small also(remember that BigOh(O) takes care of upper bound) .

As a result we can say that the iterative approach is evaluating in polynomial time, whereas recursive one is evaluating in exponential time

Why is Recursion slower?

When you call your function again itself (as recursion) the compiler allocates new Activation Record (Just think as an ordinary Stack) for that new function. That stack is used to keep your states, variables, and addresses. Compiler creates a stack for each function and this creation process continues until the base case is reached. So, when the data size becomes larger, compiler needs large stack segment to calculate the whole process. Calculating and managing those Records is also counted during this process.

Also, in recursion, the stack segment is being raised during run-time. Compiler does not know how much memory will be occupied during compile time.

Referensi: 1. https://matrixread.com/fibonacci-series-iterative-vs-recursive/
2. https://www.codeproject.com/Articles/21194/Iterative-vs-Recursive-Approaches 3.
https://stackoverflow.com/questions/21710756/recursion-vs-iteration-fibonacci-sequence
4. https://syedtousifahmed.medium.com/fibonacci-iterative-vs-recursive-5182d7783055
5. https://www.fayewilliams.com/2015/05/12/fibonacci-recursively-or-not/