Laporan Akhir Praktikum

Dipergunakan untuk memenuhi tugas individu

Mata Kuliah Pemrograman Berbasis Fungsi

Dosen Pengampu:

Riksa Meidy Karim, S.Kom., M.Si., M.Sc.

Acep Purqon S.Si.,M.Si.,Ph.D.

Ahmad Luky Ramdani, S.Kom., M.Kom.

Disusun Oleh:

Dimas Wahyu Saputro

NIM. 120450081

Kelas : PBF RA

Program Studi Sains Data

Jurusan Sains

Institut Teknologi Sumatera

2021/2022

JURNAL
PRAKTIKUM

Jurnal MODUL 1

April 11, 2022

1 Pengantar Pemrograman Fungsional

Waktu: 120 menit

1.1 Soal

Seorang mahasiswa sains data ingin menyewa buku dari sebuah startup yang menyediakan layanan sewa buku. Startup tersebut memiliki ketentuan sewa dengan aturan sebagai berikut:

a. Harga sewa buku berbeda-beda sesuai dengan kategorinya

b. Harga sewa buku dihitung berdasarkan jumlah halaman nya

c. Harga sewa buku dihitung per hari nya

d. Maksimal durasi sewa adalah 26 hari

Startup tersebut masih dalam tahap awal pengembangan, sehingga ingin melakukan uji coba penyewaan 5 kategori buku. Berikut rincian kategori nya:

```
- Kategori 1 : 100 rupiah per lembar per hari
- Kategori 2 : 200 rupiah per lembar per hari
- Kategori 3 : 250 rupiah per lembar per hari
- Kategori 4 : 300 rupiah per lembar per hari
- Kategori 5 : 500 rupiah per lembar per hari
```

Startup tersebut memerlukan sebuah program untuk:

- menghitung total biaya dari customer
- mencatat tanggal awal sewa, dan durasi hari
- menampilkan informasi kapan tanggal pengembalian buku dari customer

Format input tanggal adalah yyyy-mm-dd

Bantulah startup tersebut membuat program tersebut dengan menggunakan konsep modularisasi!

Jurnal_PBF_M1_120450081_Dimas Wahyu Saputro

May 16, 2022

1 Jurnal - PBF - M1

Dimas Wahyu Saputro - 120450081

[4]:
```
# kategori, harga per lembar per hari
# input tanggal 2020-01-02
# input durasi 25
tanggal = input('Tanggal Pinjam: ')
durasi = int(input('Durasi Pinjam (hari): '))

kategoris = {
    1: 100,
    2: 200,
    3: 250,
    4: 300,
    5: 500,
}
```

[5]:
```
def dtl (s_tgl):
    return [ int(k) for k in s_tgl.split('-')]

def is_cm (tgl_p,d,c):
    return tgl_p[2]+ d > c

def thn_back (tgl_p,d,c):
    return tgl_p[0]+1 if ( is_cm(tgl_p,d,c) and tgl_p[1] == 12) else tgl_p[0]

def bln_back (tgl_p,d,c):
    return ( tgl_p[1] % 12 )+1 if is_cm(tgl_p,d,c) else (tgl_p[1])

def tgl_back (tgl_p,d,c):
    return tgl_p[2] + d - c if is_cm(tgl_p,d,c) else tgl_p[2] + d

def is_awal_abad(thn):
    return thn % 100 == 0

def kabisat (thn):
```

```
    return(is_awal_abad(thn) and thn % 400 == 0) or (not is_awal_abad(thn) and␣
 ↪thn % 4 == 0)

def dec_c(t):
    return 30 +( t[1]%2 if t[1]<= 8 else abs((t[1]%2)-1)) if t[1] !=  2 else(29␣
 ↪if kabisat(t[0]) else 28)

def wkt_kembali (tgl_p,d):
    return [thn_back(tgl_p,d, dec_c(tgl_p)),bln_back(tgl_p,d,␣
 ↪dec_c(tgl_p)),tgl_back(tgl_p,d, dec_c(tgl_p))]
```

[6]:
```
tgl_p = dtl(tanggal)
wkt_kembali(tgl_p,durasi)
```

[6]: [2020, 1, 27]

[7]:
```
sewaan_all = [ [1,5], [2,3], [3,0], [4,1], [5,2] ]

def calc_biaya_per_kategori(kategoris, sewaan):
    return sewaan[1] * kategoris.get(sewaan[0])

def calc_all_biaya(kategoris, sewaan_all, durasi):
    return sum([calc_biaya_per_kategori(kategoris, sewaan) for sewaan in␣
 ↪sewaan_all]) * durasi
```

[9]:
```
calc_all_biaya(kategoris, sewaan_all, durasi)
```

[9]: 60000

Jurnal MODUL 2

April 25, 2022

1 Higher Order Function

Waktu: 120 menit

1.1 Soal

Kerjakan seluruh soal berikut dengan menggunakan higher order function map,filter dan reduce!

1. Buatlah sebuah fungsi bernama ulangi_NIM, ulangi memiliki input sebuah bilangan skalar a, dan mengeluarkan vektor 1xn dengan seluruh elemen nya adalah a !
2. Buatlah deret bilangan sebagai berikut dengan input n sebagai panjang deret:

   $\frac{1}{2}$, $-\frac{1}{4}$, $\frac{1}{8}$, ... , $(-1)^n \frac{1}{2^{n+1}}$
3. Jumlahkan deret bilangan tersebut!
4. Sebuah DNA dimodelkan dalam sebuah string menjadi sequence TCGA dan disimpan ke dalam data :

   `https://drive.google.com/file/d/18C1ylsTXrY9pglqqlhijoS8LYmcxdIjM/view?usp=sharing`

hitunglah jumlah kemunculan pola berikut pada data tersebut:

a. A

b. AT

c. GGT

d. AAGC

e. AGCTA

5. Reverse complement dari suatu sequence string DNA memiliki aturan sebagai berikut:

   A adalah komplemen dari T

   C adalah komplemen dari G

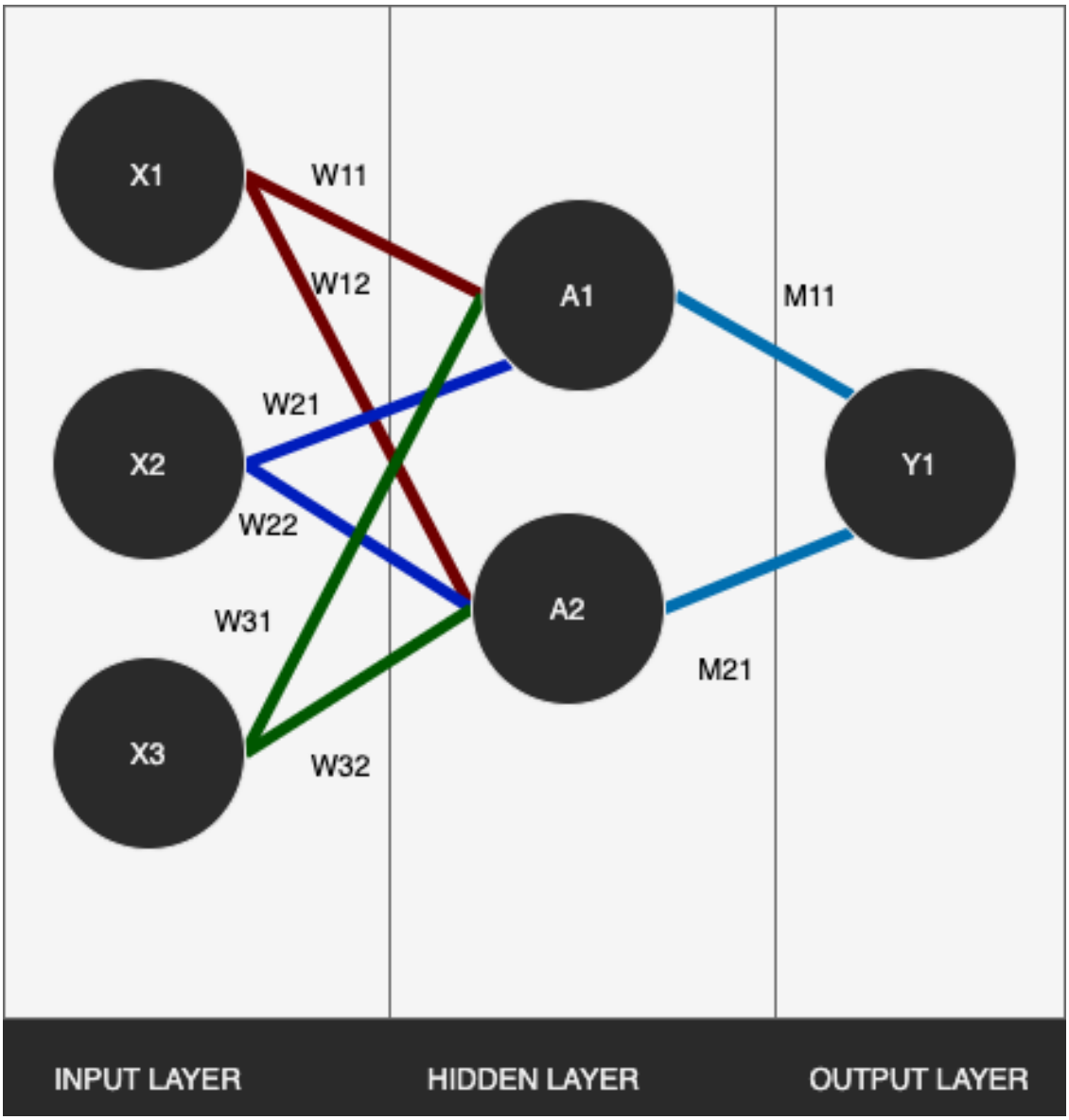
   Contoh reverse complement:

   input DNA : ACTGA

   Reverse complmenet : TGACT

   Buatlah fungsi untuk mencari inverse komplemen dari data pada nomor 4 !

6. Perhatikan Neural Network dibawah ini:



Terdapat proses yang dinamakan feed-forward. Input dalam sebuah neural network diproses ke hidden layer hingga ke output layer.

Setiap Node, menunjukan neuron dan setiap garis menunjukan weight.

Proses Feed-Forward berjalanan dari input layer menuju output layer.

Nilai yang masuk ke neuron di hidden layer adalah penjumlahan antara perkalian weight dengan nilai yang masuk pada input neuron setelah itu diaktifkan dengan fungsi aktivasi. Atau dapat dimodelkan sebagai berikut:

$$S_1 = X_1.W_{11} + X_2.W_{21} + X_3.W_{31}$$

$$S_2 = X_2.W_{12} + X_2.W_{22} + X_3.W_{32}$$

$$A_1 = \frac{1}{1 + e^{-S_1}}$$

$$A_2 = \frac{1}{1 + e^{-S_2}}$$

$$Z_1 = M_{11}.A_1 + M_{21}.A_2$$

$$Y_1 = \frac{1}{1 + e^{-Z_1}}$$

Buatlah fungsi feed-forward dengan input berikut:

W11 = 0.5

W12 = 0.4

W21 = 0.3

W22 = 0.7

W31 = 0.25

W32 = 0.9

M11 = 0.34

M21 = 0.45

dan X1 = 9 , X2 = 10 , X3 = -4

Jurnal_PBF_M2_120450081_Dimas_Wahyu_Saputro

May 16, 2022

1 Tugas Pendahuluan - PBF - Modul 2

Dimas Wahyu Saputro 120450081

1.1 Soal

Kerjakan seluruh soal berikut dengan menggunakan higher order function map,filter dan reduce!

1.2 1

Buatlah sebuah fungsi bernama ulangi_NIM, ulangi memiliki input sebuah bilangan skalar a, dan mengeluarkan vektor 1xn dengan seluruh elemen nya adalah a !

Referensi: Matrix

[ ]:
```
'''
Karena ingin mencetak bilangan skalar a, sebanyak n, digunakan map().
map() mempunyai struktur map(function, iterable(s))

1. pada function, saya menggunakan lambda. x menjadi parameter, dan a menjadi␣
↪ekspresi.
    akan selalu mengembalikan nilai a.
2. iterable, yaitu range(0,n). Artinya, akan membuat urutan angka dari 0 hingga␣
↪n.

Ketika map(lambda x: a, range(0, n)), akan mengembalikan bilangan skalar a,␣
↪sebanyak n.
'''
def ulangi_081(a, n):
    return list(map(lambda x: a, range(0, n)))

''' saya disini ingin mencetak vektor 1xn, dengan bilangan = 10, dan n = 4 '''
ulangi_081(10, 4)
```

[ ]:
```
[10, 10, 10, 10]
```

1.3 2

Buatlah deret bilangan sebagai berikut dengan input n sebagai panjang deret:

$$\frac{1}{2}, \frac{-1}{4}, , (-1)^n \frac{1}{2}$$

Referensi: Deret - RuangGuru

[ ]:
```
import functools

''' Membuat fungsi untuk menghitung nilai deret ke-n '''
def calc_seq(n):
    return pow(-1, n + 1) * (1/pow(2, n))

'''
Membuat fungsi untuk mencetak deret dari deret ke-1 sampai ke-n
Langkah awal, menggunakan map(). map() mempunyai struktur: map(function,␣
↪iterable(s))

1. pada function, saya menggunakan lambda. x menjadi parameter, dan calc_seq(x)␣
↪menjadi ekspresi.
    Artinya, dari setiap nilai iterable nanti, akan mereturn nilai calc_seq(x).
2. iterable, yaitu range(1, n+1). Artinya, akan membuat urutan angka dari 1␣
↪hingga n+1.
'''
def gen_seq(n):
    return list(map(lambda x: calc_seq(x), range(1, n+1)))

gen_seq(4)
```

[ ]:
```
[0.5, -0.25, 0.125, -0.0625]
```

1.4 3

Jumlahkan deret bilangan tersebut!

[ ]:
```
import functools

'''
Untuk menjumlahkan dari deret yang sudah ada, dibutuhkan reduce() function.
fungsi reduce() menghasilkan suatu nilai kumulatif dari operasi fungsi masukan␣
↪terhadap nilai pada iterable masukan.

'''
def sum_seq(n):
    seq = gen_seq(n)
    return functools.reduce(lambda x, y: x+y, seq, 0)

sum_seq(4)
```

[ ]:
```
0.3125
```

1.5 4

Sebuah DNA dimodelkan dalam sebuah string menjadi sequence TCGA dan disimpan ke dalam. Hitunglah kemunculan pola!

Referensi: Count the number of times a letter

[ ]:
```
''' Membuka file .txt '''
filename = 'DNA.txt'
dat = open(filename, 'r').read()

''' Ketika file .txt dibuka, diakhir akan terdapat string \n. Hal ini harus␣
 ↪kita cegah, karena kita hanya ingin string DNA '''
dat = dat[:-1]
#dat
```

[ ]:
```
'TGTCTTCCGGCTGAGCGGTTCCTAACCAGCAGACTGATACTGGTCGAATATCGACGGGCAAGAGCCCTGGGATTGATGC
GTTTCACCATGCGCGTCTCAGTGCAGGCAGGAATGCAGAGCTTACTTCAAACTAGTTACTGGCAAAAAATACAAATTTTT
TCGATCGACCTTGAGTTTATTCATTACCGCACAGTCTTTTACCGCACCTGTTACCGCACATCCGTAAGTTTACCGCACGT
TACCGCACTACCTCTCTATATTACCGCACTTCGTTTACCGCACGCTGAGGAACGGTTACCGCACTTACCGCACCACAAGG
TGCGTGCTCTGTTATTACCGCACCACCATTACCGCACGCACTTTTATTACCGCACCAGGGCACAGCCACGTAGGGTAGCG
TCGTTCTCACTGTATTGCGGCGACGGTCGTAATTTACCGCATTACCGCACCACTCGTTAGCTTACCGCACCTAGGGTTGT
TACCGCACGACTTACCGCACAGCCGTTACCGCACGTGTTACTTGACGCTCTAACTCCCACTCATATCAGTCTTATTACCG
CACACTGGGCTTACCGCACCCGCACCTTAAGTAGGCAGTTACCGCACGTATTACCGCACGTAATTACCGCACACCTGTAA
AGGCAGGGTAAAGTACAGACTTACCGCTTACCGCACGGTTGCACCACGACAAATCTAACGTTAGGTACGTTACCGCACGG
GAAATTACCGCACTCCAGGGTTTTACCGCACAGATATCCATTCGGGAATGTGACCCCTGGAGTGGAGTTGTGCGAAAGAT
ACGGAGTTTTCAAGGGCACACCCAGCTATGTTATTAAGCGTTACAGTGGCCGCTGCATCATGTCAATGTTCAGGTCATTC
TCTATCTTGCTATGTACGAACCCTCGTTAAGAGGGAGTAAGCGATCTTTTGACAAAATCGTATGCATGTAGGCGAGGCAA
TGCCGATTACATTGAACGGCGGGACTTTTCGTATGAGACACCGCGGTTGAAATATTTTTTTATGCAAGAGCGGGATTGGG
CGGAAGGAGACTTAACGCAGTGCCTAGCACTGTTAACTGCGGCATGGCCGGATGGACTACCTATTTTGCAGCTCCAGCGT
TTGAGTTCCACGTACTGACGGAACAGTCCCGAGATAGGCCATGTGGTCGATCCCAGTGAGAAATGAGACTCGAGATGCCG
GTACCGGTAGCATCACCACATTGCTCCAGTATGATATCAGTCTTCACTGTCAGCAATTAATGCAGCGATCTTGAAGAGAG
TTATTCATCTCTTATCACCTGACAATAAATCAATTTACCAGTCAAATTCTCTTTAACATCGTGCCGAACTGCGATGCGTC
GTAGTCTAGATTAGGATATATTTTCTTAGCTGGCTTCGATGATTGGCTGTACGCTAAGGTGATTGAATTTCGATCTGCAT
TGGAGCTGTACCCCACCTTGCATGGCATTGACAGCCTAAAGCGTGAAGAATGCAATACAGCTGACAGAAAAATAACGGGC
TCGATAACGTTCCAAGATTCTGACTTAACGACGGCTAGCGAGCGAGTCATAAATCCCGTCCACACCGGGCAATCGGGTCG
GAGTGGAAAGGGCGGGATTTTATTATTACGTGACGCAGATCTCCGTGTCACTATACTCACATCCTCTCTGTAGATAAAGT
TATACCAACCTCCATATTCTTCTTACGCTAAGTTCGGGCTATCCGAGTCTCGGCCCATAGCAGGAGCACTTTAAGGGAAG
TCCTATTGCCGAATACAGTACGTTCCCCGCAATATGTTATACTCACCCAAATATGTTAATATGTTATATGTTAAAACGCA
GTGTGGGAATATGTTAATATGTTATGTGAATATGTTAAAATATGTTAAAATATGTTAACGATGTTAGCCGTGATAAATAT
GTTATTAACGGCGTGCGTTAATATGTTAGCGACGACTGGGGGTCAATATGTTAGCCAACTTCCTCAATATGTTAACCGGT
TAATATGTTAGTTAAGATCAATAAATATGTTAGCTACGTAGACAATAAAAGCATAAGCAATATGTTATAATATGTTAGAC
AGTTCTCTAACCGATAATATGTTAAGGCATACTTAACCAGCGAATGACAGAAATATGTTAATATGTTAAATAAATATGTT
AGATAATATGTTACGATATTACCCGCACATTGCTCCGAATATGAATATGTTAAGGTGGTTCTCCGTATTTAATATTGTGA
GAGATAGCTTGTGAGAGATGTTGTTGTGAGAGAGCTGTGAGAGCTTTGCGAGCCTTTAAATATTGTGAGAGTTATGTAGT
CGGCCTGTGAGAGATGTGTGAGAGAGTTAAATATGTTAGTTAGCTGAGAGCTACGCTGTGAGAGGCGAATTGACGTAGTG
```

CTTTTTGTTCTGTGAGAGATGTGTGAGAGTGTGAGAGCGCGTGTGAGTGTGAGAGTGATTGTGCATGGTCCAGTAAGATG
TGAGAGTGTGAGAGTGATCTAACGCTATGTGTGAGAGAGGGTGTGAGAGGCTGCGTATGAAGCACAAATGTGAGAGTTGT
GAGAGATACGTTAAGAGCCCGGAAGCTCGGCATCATAAGCTGAGCAGATTCAATGTGAGAGGGCGAGCCGACGGTAGGCT
GTGAGAGTCATTATTGTGAGAGTCGCGTGGTGTGAGAGTCCCATTTTATGTGAGATGTGAGAGCTCTGGGGCTGTGATGT
GAGAGAGTACGCCGAAGCGTGTGAGAGTCCTGTGAGAGATTCGGAGGTCTGGATGACATTGTGAGAGCCTGCTTACGCGA
CGTGATGAACGCGACCGACTAGCGACCGCCCACTACTACTCGCAGTTGGTCTAGAGGCATTGCTTTACTGAAATACGCAG
GATGCTTATGACGCTCGCGCCAATACATCGCGCTCGCACTGTATGTCGCTTCACCTTAATCCTAAAGCTCAAATATAACG
GAAAAAGAGAAATTAGGACGACCGAGGGTCGTCCTCCGGTGGTTTTCACGACTTCGCCAATGGCGTGCTGCGTCGAAATG
TGCTCAAAGCCCCGTAAAGCTCAGACACCATGCAGGAATGGGAATGTGTACCCAGAGATCCCTAGTAAGAGAGATCCAAG
ACTTAAAGCCGTTCCGAGAGAGATCTAATCACTAGAGATCTTAACACCAATAGAGATCCTCTAAGAGATCAGTAGAGATC
GCTTTTCAGAGATAGAGATCACTCACCGAGAGATCTTACAGTTTGATATGTCAGTTCGGTTAAAAGCAGAGATCGTCTGC
AGAGATCGGTAGCGTAGAGATCCCGTGTCGTACAAAAACTTAGAGATCAGATCGCGCCTCGAACTGTACTTAGAGATCTA
CATTATCTAAGAGAGAGATCAGAGATCACAAGGCCACACACGACAAAGTTAGAGATCTACACACGATAGGTGGTGCCGAA
CCTGAGAGATCCGGGTTTTGAGAGAGATCAAGAGATAGAGATCGTTAGAGAAGAGATCTAGAGATCGCACGGGTTTTGGA
GAGATCGTTCGGGTTTTGTCGGGAAGAGATTAATGCGGTGAGTTAATGCGGGTATAATGCGGCAGATAATGTAATGCGGT
CTAATGTAATGCGGGAGATAATGCGGTGATGAAACTTAATGCGGCTAATGCGGTTAATGCGGTCGAACGCTAATGCGGAG
CTAATGCGGGCGTAACATAATGTAATGCGGTTGTCAATATTGTTTTCAATATTAATATTCAATTACAATATTCAAACGCA
ATATTAAACGGCCGGTAATGCGGGGTCAATCAATATTTTCGTAATGCGGGGTTAATGCGGTTTTCAATATTATCGGTAAT
GCGGGAGCTGGCAATATTGGTTTTGGTAATGCGGTTTAATAATGCGGGGCGACAATATTGGGTAATGCGGATATTTATCA
ATATTGTGTTTCAATATTTAACACAATATTTGCCGTAGGTATGACCTAATTAATGCGGATATTAGGGGCCAATTAATGCG
GATCGTAATGCGGGTCGGGCTTATAACAATTAATGCGGTCAATATTACTAATGCTAATGCGGCGGACTACAATATTTACA
AAAGACTACCAATAATGCGGATAATGCGGTCAATAATGCGGAAGATAACGCGGCAATATTGCCCGACAATATTTGACTAC
ACAAGACTACACAATATTCCGTTATTCTGTGCCAACGCCAGGTCAATGCGTCGAACCAATATTCTTGATTGTGATGCAGA
CTACACGACTACAATATTTACCCCCGGGACTACATATCCACGACTACAGGGCGAGACTACATAGGGACTACAGACTACAA
CAATTATGGTCACATTAACTCTGCCCGGCGGCTCTTCCCTAAATCTCACGTGATGGACTAGACTACACCGACTACAACAT
ACTTTGCAACGACTACAGTACGTTAAGACTACAGGATTCAGACTACACTTGATTTCCTGACTACACTTCTGACAACCCGC
ACATTGCCCGCTAACTCTGATGGCCCCCAGAGACTACATACCATCGAGCGCGACTACAGGACTACAGCCGTAGACCCTTT
AGACTACACGCCAGGGCCAACTGACACGGATAAGGTCTTTGCCCCGCAAGTGCTCGCCGAATGTGATTAATCTCAACATT
CCGACCTGCAAGAGCACACGCATTTGATATGGGTATAAGGAAGATCTCGTCCAGCTATAATGTACAACATTTCCCCGTCA
TGACTTGCTACATAAACAGAATAAGACGTGACGTCGCCAATATAAGACGTACTCGATTGACCGTAAAATTTTTCTAAGAA
CCAAAATAAGACGTAAGACGTTCACTTAAATAAGACGTAGGGCGTTACCGATAAGACGTTAAGACGTGGATCGCCATCGC
CCGTGAGTCGCTCTCCCGCATAAGACGTTAAGACGTCCCAATAGTGCTCCCTACACTTTACCGGTGGTAGATAAGACGTA
GACGTTATAAGACGTCGGGTAAATATAAGACGTTATTCCCAATAAATAAGACGTAATCCCTGTAACACTGGAAGTGATAA
GACGTTTGTTCTAACATAAGACGTTGTAACTGCCCTAACCCTGATAAGACGTTTTAAAAAGTACTATAAGACGTTCGAGG
AATGAGACCATAAGACGTCGTCCCTCCCTCAGCACTGAATTTTTTCGAAGATAAGACGTAAGACGTGTTGGTTTATCGTT
AGAAATAAGACGTACGTTTATAAGACGTAATGGTCATAAGACGTACGTTAAGACGTAATAAGACGTTATCCATCCCAAAA
TTACACGTCAGAAATCATGGCAACCGCCGTGATGGAAGAGAGTAGCAACCGACTACATACAGTATACTGTGGGCAGACTC
GTTTGTACACCAACACTTCCGCCGCCATTATTAAATACGATTGGTGCTTTACGCATCTTGATGACCATGGTTACTCACCT
CGGGTGCTGACCCGCCTGTCTCCTATGACGTCGGGCTCCACTACGGCCCCGTTTCGACAGATAGGGGGGAGTTGACCTCG
AATGCGGGTTACTTCGCCTGCCTTTCGACGAATCGGTATGGCTAGCTTGGACAAGTATAGGATTGGTCTTTCAAGCTGCA
CTGTTTTGCAGCTTCTAGCGAGATAAGGCTGAAGCCTCCAGCGATATTGTCCAGTTGGAAAAAAGTTGGAAAAATGGGGG
TTTGGAAAAAAGAAAACGCCCGGGTTACACCGGGGACATAATTGGAAAAAACCAGTTGGAAAAAGCTTAGAAAGCTTGGA
AAAAGTCTTGGGAACAATTATTGGAAAAACGATGGGCGACTGAGAGTTGGAAAAAAAAATTGGAAAAATTGGAAAAACGT
GGCTTTGGAAAAATGGAAAAAGATTGGAAAAATTGGAAAAACTTTTGGAATTGGAAAAAAAAGCCACTGCGGGTGCTTTG
GAAAAAATATTTGGAAAAAACTAGCAAAGCGGCATTCTGAGAGATTGGAAAAACGTGCTAAGCTTCTTTGGAAAAAGAAT
TGGAAAAAAAAAAGCGCACCACTCAGGAAGACATGTCTGGCACTTTAGCGTTAAAGTTTGGAAAAAACTCCTCCCACATT
TGGAAAAATGGAAAAAGAATCGGTTAGAGCGGCACGTGTCATATTGGAAAAATACTCAGCGCGTTAGCAGTTGGAAAAAA

```
ATGATGACTATGTTTGGAAGACAAGGAGAAAGTCTCCGAACAACATCCATGACAAGGAGGAGGCTGGACAAGGATTCAGG
CTGTTCAGACAAGGAGGGACGACAAGGAAGGACTGTTCAGGCTGGACAAGACAAGGACTGTTGACAAGGACAGGACAAGG
ACGAAAGGCTGTTCAGGGACAAGGAAGGACAGGCTGTTCAGAGGACGAGGACGACAAGGAAGGCTGTTCAGGCTGTTCAG
GAGGACGAGGAAGGATGTTCGACAAGAGGACGACAGGCTAGGACGACGAAGAGGACGACTGTTCAGGCTGTAGGACGAGG
ACGAAAGGATGTTGACAAGGAGGAGAGGAGGACGAGGAAGGACGAAAGGAGACAAGGAGACAAGGAGACAAGGAAGGACG
AAGGACGAAGGACGAAGGACGAAGGACGAAGGACGAAGGACAGAGGACAGGACGACGAAGGACGAAGGACGAAGGACAGG
ACGAAGGACGAAGGACGAAGGACGACAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCA
TCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCA
ATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATC
ATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCAT'
```

[ ]:
```python
import functools as ft

'''
Karena harus dimodularkan, fungsi append_n berfungsi untuk mendapatkan string terpecah dari data, mirip seperti split().
Pada fungsi append_n digunakan reduce, supaya mengembalikan satu nilai.
Misalkan append_n(dat, 0, 3), maka akan mengembalikan nilai 'TGT'
'''
def append_n(dat, i, n):
    return ft.reduce( lambda a,b:a+b , dat[i:i+n] )

'''
fungsi remap() berguna untuk membuat list yang berisi string terpecah dari data.

Digunakan map, dengan fungsi lambda yang akan memproses setiap item dari iterable,
dengan iterable range(len(dat) - len(seq)), artinya panjang data - panjang seq.

Misalkan list(remap(dat, 'ACT')), akan mengembalikan ['TGT', 'GTC', 'TCT', ....]
'''
def remap(dat, seq):
    return map( lambda x: append_n(dat,x,len(seq)) , range(len(dat) - len(seq)+ 1 ) )

'''
Setelah didapatkan list, dibuat fungsi count_mer() untuk menghitung berapa kali seq tertentu muncul.
digunakan reduce() akan menghasilkan suatu nilai kumulatif dari operasi fungsi masukan.
'''
def count_mer(dat, seq):
    return ft.reduce( lambda a, b: a + (1 if b == seq else 0) , remap(dat, seq), 0 )
```

[ ]:
```
append_n(dat, 1, 4)
```

[ ]:
```
'GTCT'
```

[ ]:
```
sequences = [ 'A', 'AT', 'GGT', 'AAGC', 'AGCTA' ]

''' fungsi untuk menghitung kemunculan sequences, menggunakan fungsi yang sudah␣
 ↪dicoba di atas '''
def count_all(dat, sequences):
    return map ( lambda x: count_mer(dat,x), sequences )

res = count_all(dat, sequences)
print(*res)
```

```
2112 557 77 22 5
```

1.6 5

Reverse complement dari suatu sequence string DNA memiliki aturan sebagai berikut:

A adalah komplemen dari T

C adalah komplemen dari G

Contoh reverse complement:

input DNA : ACTGA

Reverse complmenet : TGACT

Buatlah fungsi untuk mencari inverse komplemen dari data pada nomor 4 !

[ ]:
```
''' Metode get() mengembalikan nilai item dengan kunci yang ditentukan. '''
def komplemen(x):
    return {'A':'T', 'T':'A', 'C':'G', 'G':'C' }.get(x)

''' Mereverse komplemen data menggunakan map '''
def reverse_komplemen(dat):
    return map( lambda x: komplemen(x), dat)
```

[ ]:
```
res = reverse_komplemen(dat)
print(*res)
```

```
A C A G A A G G C C G A C T C G C C A A G G A T T G G T C G T C T G A C T A T G
A C C A G C T T A T A G C T G C C C G T T C T C G G G A C C C T A A C T A C G C
A A A G T G G T A C G C G C A G A G T C A C G T C C G T C C T T A C G T C T C G
A A T G A A G T T T G A T C A A T G A C C G T T T T T T A T G T T T A A A A A A
G C T A G C T G G A A C T C A A A T A A G T A A T G G C G T G T C A G A A A A T
G G C G T G G A C A A T G G C G T G T A G G C A T T C A A A T G G C G T G C A A
T G G C G T G A T G G A G A G A T A T A A T G G C G T G A A G C A A A T G G C G
T G C G A C T C C T T G C C C A A T G G C G T G A A T G G C G T G G T G T T C C A
```

C G C A C G A G A C A A T A A T G G C G T G G T G G T A A T G G C G T G C G T G
A A A A T A A T G G C G T G G T C C C G T G T C G G T G C A T C C C A T C G C A
G C A A G A G T G A C A T A A C G C C G C T G C C A G C A T T A A A T G G C G T
A A T G G C G T G G T G A G C A A T C G A A T G G C G T G G A T C C C A A C A A
T G G C G T G C T G A A T G G C G T G T C G G C A A T G G C G T G C A C A A T G
A A C T G C G A G A T T G A G G G T G A G T A T A G T C A G A A T A A T G G C G
T G T G A C C C G A A T G G C G T G G G C G T G G A A T T C A T C C G T C A A T
G G C G T G C A T A A T G G C G T G C A T T A A T G G C G T G T G G A C A T T T
C C G T C C C A T T T C A T G T C T G A A T G G C G A A T G G C G T G C C A A C
G T G G T G C T G T T T A G A T T G C A A T C C A T G C A A T G G C G T G C C C
T T T A A T G G C G T G A G G T C C C A A A A T G G C G T G T C T A T A G G T A
A G C C C T T A C A C T G G G G A C C T C A C C T C A A C A C G C T T T C T A T
G C C T C A A A A G T T C C C G T G T G G G T C G A T A C A A T A A T T C G C A
A T G T C A C C G G G C G A C G T A G T A C A G T T A C A A G T C C A G T A A G A
G A T A G A A C G A T A C A T G C T T G G G A G C A A T T C T C C C T C A T T C
G C T A G A A A A C T G T T T T A G C A T A C G T A C A T C C G C T C C G T T A
C G G C T A A T G T A A C T T G C C G C C C T G A A A A G C A T A C T C T G T G
G C G C C A A C T T T A T A A A A A A A A T A C G T T C T C G C C C T A A C C C G
C C T T C C T C T G A A T T G C G T C A C G G A T C G T G A C A A T T G A C G C
C G T A C C G G C C T A C C T G A T G G A T A A A A C G T C G A G G T C G C A A
A C T C A A G G T G C A T G A C T G C C T T G T C A G G G C T C T A T C C G G T
A C A C C A G C T A G G G T C A C T C T T T A C T C T G A G C T C T A C G G C C
A T G G C C A T C G T A G T G G T G T A A C G A G G T C A T A C T A T A G T C A
G A A G T G A C A G T C G T T A A T T A C G T C G C T A G A A C T T C T C T C A
A T A A G T A G A G A A T A G T G G A C T G T T A T T T A G T T A A A T G G T C
A G T T T A A G A G A A A T T G T A G C A C G G C T T G A C G C T A C G C A G C
A T C A G A T C T A A T C C T A T A T A A A A G A A T C G A C C G A A G C T A C
T A A C C G A C A T G C G A T T C C A C T A A C T T A A A G C T A G A C G T A A
C C T C G A C A T G G G G T G G A A C G T A C C G T A A C T G T C G G A T T T C
G C A C T T C T T A C G T T A T G T C G A C T G T C T T T T T A T T G C C C G A
G C T A T T G C A A G G T T C T A A G A C T G A A T T G C T G C C G A T C G C T
C G C T C A G T A T T T A G G G C A G G T G T G G C C C G T T A G C C C A G C C
T C A C C T T T C C C C G C C C T A A A A T A A T A A T G C A C T G C G T C T A G
A G G C A C A G T G A T A T G A G T G T A G G A G A G A C A T C T A T T T C A A
T A T G G T T G G A G G T A T A A G A A G A A T G C G A T T C A A G C C C G A T
A G G C T C A G A G C C G G G T A T C G T C C T C G T G A A A T T C C C T T C A
G G A T A A C G G C T T A T G T C A T G C A A G G G G C G T T A T A C A A T A T
G A G T G G G T T T A T A C A A T T A T A C A A T A T A C A A T T T T G C G T C
A C A C C C T T A T A C A A T T A T A C A A T A C A C T T A T A C A A T T T T A
T A C A A T T T T A T A C A A T T G C T A C A A T C G G C A C T A T T T A T A C
A A T A A T T G C C G C A C G C A A T T A T A C A A T C G C T G C T G A C C C C
C A G T T A T A C A A T C G G T T G A A G G A G T T A T A C A A T T G G C C A A
T T A T A C A A T C A A T T C T A G T T A T T T A T A C A A T C G A T G C A T C
T G T T A T T T T C G T A T T C G T T A T A C A A T A T T A T A C A A T C T G T
C A A G A G A T T G G C T A T T A T A C A A T T C C G T A T G A A T T G G T C G
C T T A C T G T C T T T A T A C A A T T A T A C A A T T T A T T T A T A C A A T
C T A T T A T A C A A T G C T A T A A T G G G C G T G T A A C G A G G C T T A T
A C T T A T A C A A T T C C A C C A A G A G G C A T A A A T T A T A A C A C T C

T C T A T C G A A C A C T C T C T A C A A C A A C A C T C T C T C G A C A C T C
T C G A A A C G C T C G G A A A T T T A T A A C A C T C T C A A T A C A T C A G
C C G G A C A C T C T C T A C A C A C T C T C T C A A T T T A T A C A A T C A A
T C G A C T C T C G A T G C G A C A C T C T C C G C T T A A C T G C A T C A C G
A A A A A C A A G A C A C T C T C T A C A C A C T C T C A C A C T C T C G C G C
A C A C T C A C A C T C T C A C T A A C A C G T A C C A G G T C A T T C T A C A
C T C T C A C A C T C T C A C T A G A T T G C G A T A C A C A C T C T C T C C C
A C A C T C T C C G A C G C A T A C T T C G T G T T T A C A C T C T C A A C A C
T C T C T A T G C A A T T C T C G G G C C T T C G A G C C G T A G T A T T C G A
C T C G T C T A A G T T A C A C T C T C C C G C T C G G C T G C C A T C C G A C
A C T C T C A G T A A T A A C A C T C T C A G C G C A C C A C A C T C T C A G G
G T A A A A T A C A C T C T A C A C T C T C G A G A C C C C G A C A C T A C A C
T C T C T C A T G C G G C T T C G C A C A C T C T C A G G A C A C T C T C T A A
G C C T C C A G A C C T A C T G T A A C A C T C T C G G A C G A A T G C G C T G
C A C T A C T T G C G C T G G C T G A T C G C T G G C G G G T G A T G A T G A G
C G T C A A C C A G A T C T C C G T A A C G A A A T G A C T T T A T G C G T C C
T A C G A A T A C T G C G A G C G C G G T T A T G T A G C G C G A G C G T G A C
A T A C A G C G A A G T G G A A T T A G G A T T T C G A G T T T A T A T T G C C
T T T T T C T C T T T A A T C C T G C T G G C T C C C A G C A G G A G G C C A C
C A A A A G T G C T G A A G C G G T T A C C G C A C G A C G C A G C T T T A C A
C G A G T T T C G G G G C A T T T C G A G T C T G T G G T A C G T C C T T A C C
C T T A C A C A T G G G T C T C T A G G G A T C A T T C T C T C T A G G T T C T
G A A T T T C G G C A A G G C T C T C T C T A G A T T A G T G A T C T C T A G A
A T T G T G G T T A T C T C T A G G A G A T T C T C T A G T C A T C T C T A G C
G A A A A G T C T C T A T C T C T A G T G A G T G G C T C T C T A G A A T G T C
A A A C T A T A C A G T C A A G C C A A T T T T C G T C T C T A G C A G A C G T
C T C T A G C C A T C G C A T C T C T A G G G C A C A G C A T G T T T T T G A A
T C T C T A G T C T A G C G C G G A G C T T G A C A T G A A T C T C T A G A T G
T A A T A G A T T C T C T C T C T A G T C T C T A G T G T T C C G G T G T G T G
C T G T T T C A A T C T C T A G A T G T G T G C T A T C C A C C A C G G C T T G
G A C T C T C T A G G C C C A A A A C T C T C T C T A G T T C T C T A T C T C T
A G C A A T C T C T T T C T C T A G A T C T C T C T A G C G T G C C C C A A A A C C T C
T C T A G C A A G C C C C A A A A C A G C C C C T T C T C T A A T T A C G C C A C T
C A A T T A C G C C C A T A T T A C G C C G T C T A T T A C A T T A C G C C A G
A T T A C A T T A C G C C C T C T A T T A C G C C A C T A C T T T G A A T T A C
G C C G A T T A C G C C A A T T A C G C C A G C T T G C G A T T A C G C C T C G
A T T A C G C C C C G C A T T G T A T T A C A T T A C G C C A A C A G T T A T A A
C A A A A G T T A T A A T T A T A A G T T A A T G T T A T A A G T T T G C G T T
A T A A T T T G C C G G C C A T T A C G C C C C C A G T T A G T T A T A A A A G C
A T T A C G C C C C C A A T T A C G C C A A A A A G T T A T A A T A G C C A T T A C
G C C C T C G A C C G T T A T A A C C A A A A A C C A T T A C G C C A A A T T A T
T A C G C C C C C G C T G T T A T A A C C C C A T T A C G C C T A T A A A A T A G T T
A T A A C A C A A A G T T A T A A A T T G T G T T A T A A A C G G C A T C C A T
A C T G G A T T A A T T A C G C C T A T A A T C C C C C G G T T A A T T A C G C C
T A G C A T T A C G C C C C A G C C C C G A A T A T T G T T A A T T A C G C C A G T
T A T A A T G A T T A C G A T T A C G C C C G C C T G A T G T T A T A A A T G T T
T T C T G A T G G T T A T T A C G C C T A T T A C G C C A G T T A T T A C G C C
T T C T A T T G C G C C G T T A T A A C G G G C T G T T A T A A A C T G A T G T

G T T C T G A T G T G T T A T A A G G C A A T A A G A C A C G G T T G C G G T C
C A G T T A C G C A G C T T G G T T A T A A G A A C T A A C A C T A C G T C T G
A T G T G C T G A T G T T A T A A A T G G G G G C C C T G A T G T A T A G G T G
C T G A T G T C C C G C T C T G A T G T A T C C C T G A T G T C T G A T G T T G
T T A A T A C C A G T G T A A T T G A G A C G G G C C G C C G A G A A G G G A T
T T A G A G T G C A C T A C C T G A T C T G A T G T G G C T G A T G T T G T A T
G A A A C G T T G C T G A T G T C A T G C A A T T C T G A T G T C C T A A G T C
T G A T G T G A A C T A A A G G A C T G A T G T G A A G A C T G T T G G G C G T
G T A A C G G G C G A T T G A G A C T A C C G G G G G T C T C T G A T G T A T G
G T A G C T C G C G C T G A T G T C C T G A T G T C G G C A T C T G G G A A A T
C T G A T G T G C G G T C C C G G T T G A C T G T G C C T A T T C C A G A A A C
G G G G C G T T C A C G A G C G G C T T A C A C T A A T T A G A G T T G T A A G
G C T G G A C G T T C T C G T G T G C G T A A A C T A T A C C C A T A T T C C T
T C T A G A G C A G G T C G A T A T T A C A T G T T G T A A A G G G G C A G T A
C T G A A C G A T G T A T T T G T C T T A T T C T G C A C T G C A G C G G T T A
T A T T C T G C A T G A G C T A A C T G G C A T T T T A A A A A G A T T C T T G
G T T T T A T T C T G C A T T C T G C A A G T G A A T T T A T T C T G C A T C C
C G C A A T G G C T A T T C T G C A A T T C T G C A C C T A G C G G T A G C G G
G C A C T C A G C G A G A G G G C G T A T T C T G C A A T T C T G C A G G G T T
A T C A C G A G G G A T G T G A A A T G G C C A C C A T C T A T T C T G C A T C
T G C A A T A T T C T G C A G C C C A T T T A T A T T C T G C A A T A A G G G T
T A T T T A T T C T G C A T T A G G G A C A T T G T G A C C T T C A C T A T T C
T G C A A A C A A G A T T G T A T T C T G C A A C A T T G A C G G G A T T G G G
A C T A T T C T G C A A A A T T T T T C A T G A T A T T C T G C A A G C T C C T
T A C T C T G G T A T T C T G C A G C A G G G A G G G A G T C G T G A C T T A A
A A A A G C T T C T A T T C T G C A T T C T G C A C A A C C A A A T A G C A A T
C T T T A T T C T G C A T G C A A A T A T T C T G C A T T A C C A G T A T T C T
G C A T G C A A T T C T G C A T T A T T C T G C A A T A G G T A G G G T T T T A
A T G T G C A G T C T T T A G T A C C G T T G G C G G C A C T A C C T T C T C T
C A T C G T T G G C T G A T G T A T G T C A T A T G A C A C C C G T C T G A G C
A A A C A T G T G G T T G T G A A G G C G G C G G T A A T A A T T T A T G C T A
A C C A C G A A A T G C G T A G A A C T A C T G G T A C C A A T G A G T G G A G
C C C A C G A C T G G G C G G A C A G A G G A T A C T G C A G C C C G A G G T G
A T G C C G G G G C A A A G C T G T C T A T C C C C C C T C A A C T G G A G C T
T A C G C C C A A T G A A G C G G A C G G A A A G C T G C T T A G C C A T A C C
G A T C G A A C C T G T T C A T A T C C T A A C C A G A A A G T T C G A C G T G
A C A A A A A C G T C G A A G A T C G C T C T A T T C C G A C T T C G G A G G T C
G C T A T A A C A G G T C A A C C T T T T T T C A A C C T T T T T A C C C C C A
A A C C T T T T T T C T T T T G C G G G C C C A A T G T G G C C C C T G T A T T
A A C C T T T T T T G G T C A A C C T T T T T C G A A T C T T T C G A A C C T T
T T T C A G A A C C C T T G T T A A T A A C C T T T T T T G C T A C C C G C T G A
C T C T C A A C C T T T T T T T T T T A A C C T T T T T T A A C C T T T T T T G C A C
C G A A A C C T T T T T T A C C T T T T T T C T A A C C T T T T T T A A C C T T T T T
G A A A A A C C T T A A C C T T T T T T T T T C G G T G A C G C C C A C G A A A C C
T T T T T T A T A A A C C T T T T T T T G A T C G T T T C G C C C G T A A G A C T C
T C T A A C C T T T T T T G C A C G A T T C G A A G A A A C C T T T T T T C T T A A
C C T T T T T T T T T T T C G C G T G G T G A G T C C T T C T G T A C A G A C C G
T G A A A T C G C A A T T T C A A A C C T T T T T T T G A G G A G G G T G T A A A

C C T T T T T A C C T T T T T C T T A G C C A A T C T C G C C G T G C A C A G T
A T A A C C T T T T T A T G A G T C G C G C A A T C G T C A A C C T T T T T T T
A C T A C T G A T A C A A A C C T T C T G T T C C T C T T T C A G A G G C T T G
T T G T A G G T A C T G T T C C T C C T C C G A C C T G T T C C T A A G T C C G
A C A A G T C T G T T C C T C C C T G C T G T T C C T T C C T G A C A A G T C C
G A C C T G T T C T G T T C C T G A C A A C T G T T C C T G T C C T G T T C C T
G C T T T C C G A C A A G T C C C T G T T C C T T C C T G T C C G A C A A G T C
T C C T G C T C C T G C T G T T C C T T C C G A C A A G T C C G A C A A G T C C
T C C T G C T C C T T C C T A C A A G C T G T T C T C C T G C T G T C C G A T C
C T G C T G C T T C T C C T G C T G A C A A G T C C G A C A T C C T G C T C C T
G C T T T C C T A C A A C T G T T C C T C C T C C T C C T C C T G C T C C T T C C
T G C T T T C C T C T G T T C C T C T G T T C C T C T G T T C C T T C C T G C T
T C C T G C T T C C T G C T T C C T G C T T C C T G C T T C C T G C T T C C T G
T C T C C T G T C C T G C T G C T T C C T G C T T C C T G C T T C C T G T C C T
G C T T C C T G C T T C C T G C T T C C T G C T G T T A G T A G T T A G T A G T
T A G T A G T T A G T A G T T A G T A G T T A G T A G T T A G T A G T T A G T A
G T T A G T A G T T A G T A G T T A G T A G T T A G T A G T T A G T A G T T A G
T A G T T A G T A G T T A G T A G T T A G T A G T T A G T A G T T A G T A G T T
A G T A G T T A G T A G T T A G T A G T T A G T A G T T A G T A G T T A G T A G
T T A G T A G T T A G T A G T T A G T A G T T A G T A G T T A G T A G T T A G T
A G T T A G T A G T T A G T A G T T A G T A G T T A G T A G T T A G T A G T T A
G T A G T T A G T A

1.7 6

Buatlah fungsi feed-forward!

[ ]:
```
import math

''' Fungsi aktivasi '''
def aktivasi(x):
    return 1/ (1+ math.exp(-x))

'''
Mendapatkan nilai W di setiap list sesuai dengan index.
Mapping W menjadi satu dimensi
'''
def WTi(W, i):
    return list(map( lambda w:w[i], W))

'''
Menampung WTi sesuai index dan membuat menjadi satu list
'''
def WT(W):
    return list( map( lambda i : WTi(W, i), range(len(W[0])) ) )

'''
```

```
Nilai yang masuk ke neuron di hidden layer adalah penjumlahan antara perkalian␣
 ↪weight dengan
nilai yang masuk pada input neuron.
'''
def XW(X,W):
    return map( lambda w: ft.reduce( lambda a,b:a+b, map( lambda xx,vw: xx*vw,␣
 ↪X, w), 0), WT(W) )

''' Mengaktivasi nilai yang didapat pada XW '''
def input_to_hidden(X, W):
    return list( map( lambda x:aktivasi(x) , XW(X, W) ) )

''' membuat feed-forward dari fungsi yang sudah dibuat di atas, supaya modular␣
 ↪'''
def feed_forward(X, W, M):
    return input_to_hidden(input_to_hidden(X, W), M)
```

[ ]:
```
X = [ 9, 10, -4 ]
W = [ [ 0.5, 0.4 ] , [ 0.3, 0.7 ] , [ 0.25, 0.9 ] ]
M = [ [ 0.34 ], [0.45] ]

feed_forward(X, W, M)
```

[ ]:
```
[0.6876336740661236]
```

LATIHAN

Exercise 9

May 16, 2022

1 Exercise 9 PBF

1.1 1

Buat program untuk menghitung deret bilangan prima dari 2 hingga N menggunakan HOF filter dan map.

Contoh **primes(100)**:

2 3 5 7 11 13 17 .... 83 89 97

[1]:
```
factor = lambda n: list(filter( lambda i: n % i == 0, range(1, n+1)))
primes = lambda n: list(filter( lambda i: len(factor(i)) == 2, range(1, n+1)))

print( *primes(100) )
```

```
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```

1.2 2

```
employee = {
    'Nagao':35,
    'Ishii':30,
    'Kazutomo':20,
    'Saito':25,
    'Hidemi':29
}
```

Terdapat dictionary employee berisi nama dan umur pegawai, lakukan filter untuk mengetahui pegawai yang berumur > 25 tahun!

[2]:
```
employee = {
    'Nagao':35,
    'Ishii':30,
    'Kazutomo':20,
    'Saito':25,
    'Hidemi':29
}

print( employee.items() )
```

```
dict_items([('Nagao', 35), ('Ishii', 30), ('Kazutomo', 20), ('Saito', 25),
('Hidemi', 29)])
```

[3]:
```
print( *filter( lambda x: x[1] > 25, employee.items() ))
```

```
('Nagao', 35) ('Ishii', 30) ('Hidemi', 29)
```

[4]:
```
filter_by_age = lambda age, employee: list( filter(lambda x: x[1] > 25,␣
 ↪employee.items() ) )
print(*map( lambda x: x[0], filter_by_age(25, employee)))
```

```
Nagao Ishii Hidemi
```

Exercise 10

May 16, 2022

1 1

Buat fungsi mencari jumlah bilangan genap dari list L.

Contoh:

L = [2,1,9,10,3,90,15] -> 3

[7]:
```
import functools as ft
L = [2,1,9,10,3,90,15]

# function count the even numbers from a list L using reduce without for
def even_count(L):
    return ft.reduce(lambda x,y: x+1 if y%2==0 else x, L, 0)

print(even_count(L))
```

```
3
```

2 2

Buat fungsi untuk menghitung n! menggunakan reduce

[15]:
```
facto = lambda n: ft.reduce( lambda a, b: a*b if b > 1 else 1, range(1,n+1), 1)
print(facto(5))
```

```
120
```

[18]:
```
n = 10
for i in range(0, n+1):
    print( str(i) + '! = ' + str(facto(i)) )
```

```
0! = 1
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
```

```
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
```

3 3

Hitung euclidian distance dari dua vektor berikut menggunakan higher order function!

X = [2,5,6,7,10]

Y = [-2,9,2,-1,10]

[23]:
```
X = [2,5,6,7,10]
Y = [-2,9,2,-1,10]

euclid = lambda X, Y: ft.reduce( lambda a,c: a + c, map(lambda x,y: (x-y)**2,␣
 ↪X, Y) )**0.5
print(euclid(X,Y))
```

```
10.583005244258363
```

3.1 4

```
employee = {
    'Nagao':35,
    'Ishii':30,
    'Kazutomo':20,
    'Saito':25,
    'Hidemi':29
}
```

Terdapat dictionary employee berisi nama dan umur pegawai, lakukan filter untuk mengetahui pegawai yang berumur > 25 tahun!

[27]:
```
employee = {
    'Nagao':35,
    'Ishii':30,
    'Kazutomo':20,
    'Saito':25,
    'Hidemi':29
}

cnt_emp = lambda lim, employee: ft.reduce(lambda x,y: x+1 if y[1]> lim else x,␣
 ↪employee.items(), 0)
cnt_emp(25, employee)
```

[27]:
```
3
```

4 5

Buatlah deret fibonacci menggunakan higher order function!

[32]:
```
fibo = lambda n: ft.reduce( lambda a, b: a if b[0] <= 1 else a + [ a[ b[0]-1 ]␣
↪+ a[ b[0]-2 ] ] ,
                    enumerate( [0,1] + list(range(1, n))) , [0,1] ) if n > 0␣
↪else [0]
```

[34]:
```
for i in range(10):
    print('Fibonacci of ' + str(i) + ' = ' + str(fibo(i)))
```

```
Fibonacci of 0 = [0]
Fibonacci of 1 = [0, 1]
Fibonacci of 2 = [0, 1, 1]
Fibonacci of 3 = [0, 1, 1, 2]
Fibonacci of 4 = [0, 1, 1, 2, 3]
Fibonacci of 5 = [0, 1, 1, 2, 3, 5]
Fibonacci of 6 = [0, 1, 1, 2, 3, 5, 8]
Fibonacci of 7 = [0, 1, 1, 2, 3, 5, 8, 13]
Fibonacci of 8 = [0, 1, 1, 2, 3, 5, 8, 13, 21]
Fibonacci of 9 = [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

[36]:
```
# Recursive fibonacci using lambda
fibo_rec = lambda n: 0 if n == 0 else 1 if (n == 1 or n == 2) else␣
↪fibo_rec(n-1) + fibo_rec(n-2)
deret_fibo = lambda n: list( map( lambda x: fibo_rec(x), range(n+1) ) )
deret_fibo(10)
```

[36]:
```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

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/

Exercise 12

May 16, 2022

1 1

Ubah fungsiku menjadi pure function!

[6]:
```
def fungsiku(L):
    def check_genap(l):
        return l % 2 == 0
    for i in range(len(L)):
        if check_genap(L[i]):
            L[i] = L[i] / 2
        else:
            L[i] = L[i] * n + 1
    return L
```

[16]:
```
n = 3
L = [5,6,7,8]
print(fungsiku(L))
print(L)
```

```
[16, 3.0, 22, 4.0]
[16, 3.0, 22, 4.0]
```

[18]:
```
# change function fungsiku to pure function using map
def fungsiku_map(L):
    def check_genap(l):
        return l % 2 == 0
    return map(lambda x: x / 2 if check_genap(x) else x * n + 1, L)
```

[19]:
```
n = 3
L = [5,6,7,8]
print(list(fungsiku_map(L)))
```

```
[16, 3.0, 22, 4.0]
```

2 2

Ubah fungsiku2 menjadi pure function!

[8]:
```
def fungsiku2(L):
    def check_faktor(l):
        return l % n == 0
    for i in range(len(L)):
        if check_faktor(L[i]):
            L[i] = L[i] / 2
        else:
            L[i] = L[i] * n + 1
    return L
```

[9]:
```
n = 3
L = [5,6,7,8]
print(fungsiku2(L))
print(L)
```

```
[16, 3.0, 22, 25]
[16, 3.0, 22, 25]
```

[10]:
```
# change function fungsiku2 to pure function using map
def fungsiku2_map(L):
    def check_faktor(l):
        return l % n == 0
    return map(lambda x: x / 2 if check_faktor(x) else x * n + 1, L)
```

[13]:
```
n = 3
L = [5,6,7,8]
print(list(fungsiku2_map(L)))
```

```
[16, 3.0, 22, 25]
```

3 3

Apakah isi dalam tupel tup ada yang dapat diubah?

tup = ([3, 4, 5], 'myname')

[ ]:
```
tup = ([3, 4, 5], 'myname')
```

Tuple mirip dengan list. Bedanya, tuple bersifat immutable, sehingga anggotanya tidak bisa diubah

Exercise 13

May 16, 2022

1 1

Addku = lambda x: x + 10

Powku = lambda x: x**2

Kurku = lambda x: x - 2 * x

A. Buatlah fungsi komposisi menggunakan 3 fungsi diatas yang melakukan hal sebagai berikut secara berurut: 1. Menjumlahkan input dengan nilai 10 2. Mengurangi input dengan 2 kali input nya 3. Mengeluarkan nilai kuadrat dari input nya

B. Buatlah fungsi invers nya!

[1]:
```
addku = lambda x: x + 10
powku = lambda x: x**2
kurku = lambda x: x - 2 * x

f_komp = lambda f,g: lambda x: f(g(x))

my_f_kom = f_komp(kurku, f_komp(powku, addku))

my_f_kom(10)
```

[1]:
```
-400
```

[3]:
```
# invers
inv_addku = lambda x: x - 10
inv_powku = lambda x: x**0.5
inv_kurku = lambda x: -1 * x

my_f_kom_inv = f_komp(inv_addku, f_komp(inv_powku, inv_kurku))
my_f_kom_inv(-400)
```

[3]:
```
10.0
```

2 2

2.1 IPK

[9]:
```
from functools import reduce as r

# Define function composition
mycompose = lambda *funcs: r( lambda f, g: lambda x: f(g(x)), reversed(funcs),␣
↪lambda x:x )
```

[10]:
```
# Ketentuan jumlah tanggungan
def skor1(jtg):
    return 1 if jtg >= 5 else 5-jtg
```

[11]:
```
# Ketentuan token listrik
def skor2(X):
    def rata(X):
        return sum(X)/len(X)

    def l_cond_1(X):
        return [X, [X>100000] ]

    def l_cond_2(X):
        return [X[0], X[1] + [ X[0] >= 50000 ] ]

    def to_score2(X):
        return r( lambda a,b: a+ (1 if b == True else 0), X[1], 1)

    compose_cond = mycompose(rata, l_cond_1, l_cond_2, to_score2)
    return compose_cond(X)

# skor2([50000, 50000, 50000])
```

[12]:
```
# Ketentuan gaji

def con_1(X):
    return [X[0], 1, X[2], [ X[0] > X[2][X[1]] ] ]

def con_2_to_n(X):
    return [X[0], X[1]+1, X[2], X[3] + [ X[0] > X[2][X[1]] ] ]

def to_score(X):
    return r( lambda a,b: a+ (1 if b == True else 0), X[-1], 2)

def prep(gj):
    return [gj, 0, list(map( lambda x: x*1000000, list(range(10,3,-1)) + [3]) )]

def skor3(gaji):
```

```
    commpy = mycompose(prep, con_1, *(con_2_to_n for i in range(4)), to_score)
    return commpy(gaji)
```

[13]:
```
# Ketentuan KIP K
def skor4(X=True):
    return 1 if X else 5
```

[14]:
```
def combineskor(X):
    return X + [map( lambda f,x: f(x), X[1], X[0] )]

def boboti(X):
    return r( lambda a,b: a+b, map(lambda x,y: x*y, X[-1], [0.2, 0.3, 0.2, 0.
 ↪3]) )

def toUKT(X):
    return 750000 + X*500000
```

[15]:
```
mhs = [3,
    [120000, 75000, 50000],
    5.5 * 10**6,
    False
]

datas = [mhs, [skor1, skor2, skor3, skor4] ]
compose_fin = mycompose(combineskor, boboti, toUKT)
compose_fin(datas)
```

[15]:
```
2200000.0
```

3 3

3.1 Turunan polinom

dat = '-3x^5 + 2x^2 -4x +5'

output -> '-15.0x^4 + 4.0x - 4.0'

[16]:
```
# Turnan polinom

def split(dat):
    return dat.replace(' ', '').replace('-','+-').split('+')

def chdepan(dat):
    return dat[1:] if dat[0] == '' else dat

def eqkan(dat):
```

```
    return map( lambda x: x if '^' in x else x+ '^1' if 'x' in x else x+ 'x^0',␣
 ↪dat)

def toarr2d(dat):
    return r( lambda a, b: a + [[float(hurf) for hurf in b.split('x^')]] , dat,␣
 ↪[])

def sortdesc(dat):
    return sorted(dat, key=lambda x: x[1], reverse=True)

def calctur(dat):
    return map( lambda x: [0,0] if x[1] == 0 else [x[1]*x[0], x[1]-1], dat)

def tostr(dat):
    return map( lambda x: '0' if x[0] == 0 else str(x[0]) if x[1]==0 else␣
 ↪str(x[0]) + 'x^' + str(x[1]), dat)

def prettykan(dat):
    return r( lambda a,b: a+'+' + b if b != '0' else a, dat, '')

def prettysign(dat):
    return dat.replace('+-', ' -').replace('+', '+ ')
```

[20]:
```
dat = '-3x^5 + 2x^2 -4x +5'
fss = (split, chdepan, eqkan, toarr2d, sortdesc, calctur, tostr, prettykan,␣
 ↪prettysign)
my_turunan = mycompose(*fss)
my_turunan(dat)
```

[20]:
```
' -15.0x^4.0+ 4.0x^1.0 -4.0'
```

4 4

Buatlah fungsi untuk menghitung biaya yang harus dibayar customer pada suatu e-commerce menggunakan higher order function. Buatlah decorator untuk mengeluarkan harga sebelum pajak dan sesudah pajak (pajak = 11%) ! Gunakan decorator untuk menambahkan perhitungan waktu eksekusi!

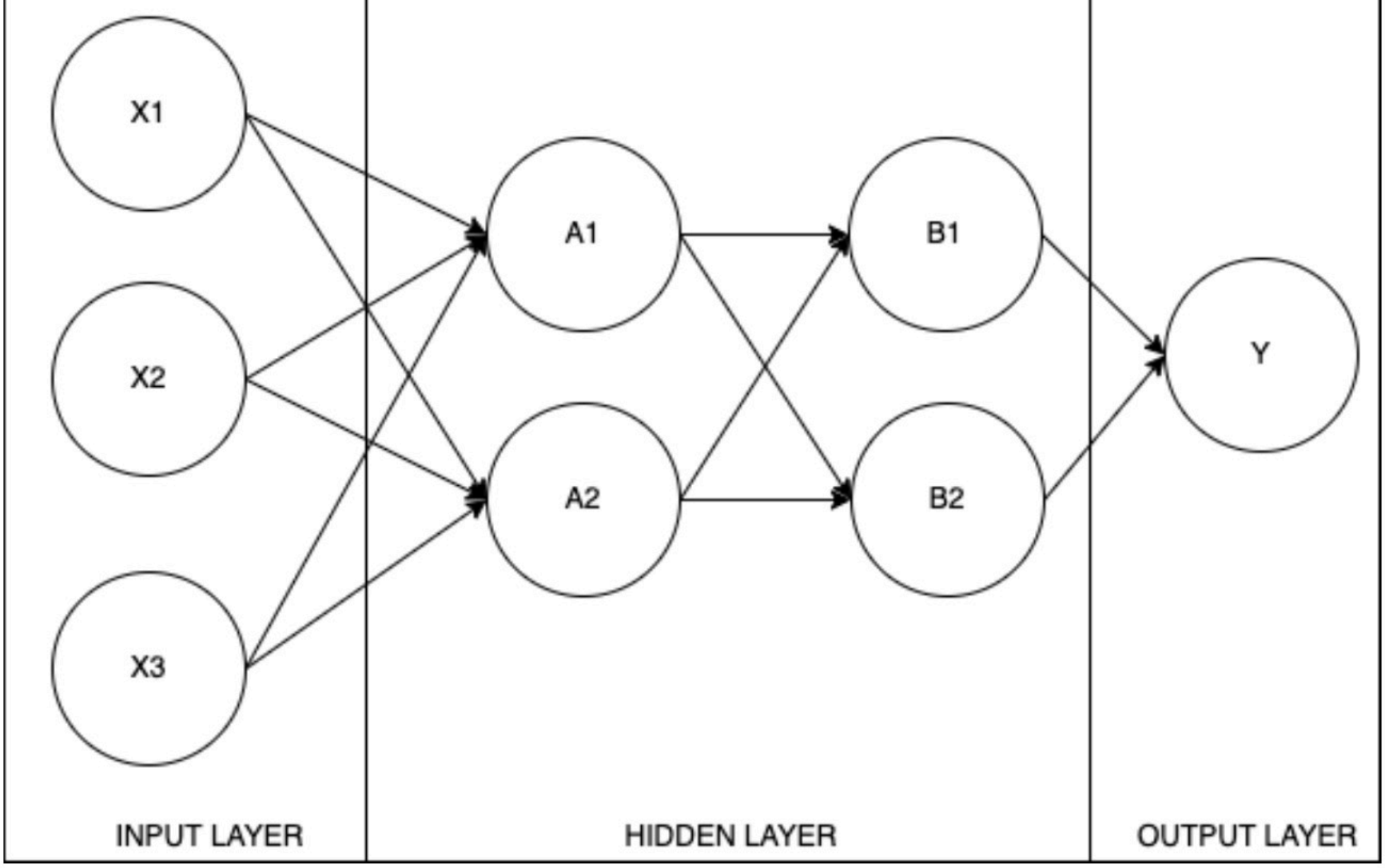
[4]:
```
from functools import reduce as r
keranjang = [
    {'Jumlah_Barang': 5, 'Harga:': 10 },
    {'Jumlah_Barang': 7, 'Harga:': 20 },
    {'Jumlah_Barang': 20, 'Harga:': 4.5 }
]

def pajak_decorator(func):
```

```
    def inner(*args, **kwargs):
        res = func(*args, **kwargs)
        print('Sub Total: ', res)
        print('Pajak: ', res * 0.01)
        print('Total: ', res + res * 0.01)
        return res
    return inner

import time

def calc_time_decorator(func):
    def inner(*args, **kwargs):
        start = time.time()
        res = func(*args, **kwargs)
        end = time.time()
        print('Time: ', end - start)
        return res
    return inner
```

[6]:
```
@calc_time_decorator
@pajak_decorator
def hitung_pembayaran_1(keranjang):
    return r( lambda a,b: a + (b['Jumlah_Barang'] * b['Harga:']), keranjang, 0)␣
 ↪* 1000

hitung_pembayaran_1(keranjang)
```

```
Sub Total:  280000.0
Pajak:  2800.0
Total:  282800.0
Time:  0.0009989738464355469
```

[6]: 280000.0



Weight antara Input Layer dengan Hidden Layer dinotasikan dengan w_{ij} dengan i adalah index neuron input dan j adalah index neuron di hidden layer 1.

Weight antara Hidden layer 1 dan Hidden Layer 2 dinotasikan dengan m_{ij} dengan i adalah index neuron di hidden layer 1 dan j adalah index neuron di hidden layer 2.

Weight antara Hidden layer 1 dan Hidden Layer 2 dinotasikan dengan h_{ij} dengan i adalah index neuron di hidden layer 2 dan j adalah index neuron di output layer.

a. Buatlah fungsi untuk membangkitkan matriks w , m , h secara random dengan batas bawah -1 dan batas atas 1 !

b. Buatlah fungsi untuk melakukan forward propagation dengan:

   Fungsi aktivasi $f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$ dan X1 , X2 , X3 bernilai 3 digit terakhir nim anda secara berurutan!

c. Buatlah decorator untuk menambahkan perhitungan running time pada fungsi forward propagation tersebut!

Latihan UTS PBF

May 16, 2022

1 Latihan UAS

Buatlah fungsi untuk membangkitkan matriks w, m, h secara random dengan batas bawah -1 dan batas atas 1.

[6]:
```
import numpy as np

gen_rand = lambda a, b, c: np.random.uniform(low = a, high = b, size = c)
W = gen_rand(-1, 1, (3,2))
M = gen_rand(-1, 1, (2,2))
H = gen_rand(-1, 1, (1,1))
```

Fungsi aktivasi untuk forward propagation

[4]:
```
# code
import math
import functools as ft

''' Fungsi aktivasi '''
def aktivasi(x):
    return (math.exp(x)-math.exp(-x))/(math.exp(x)+math.exp(-x))

'''
Mendapatkan nilai W di setiap list sesuai dengan index.
Mapping W menjadi satu dimensi
'''
def WTi(W, i):
    return list(map( lambda w:w[i], W))

'''
Menampung WTi sesuai index dan membuat menjadi satu list
'''
def WT(W):
    return list( map( lambda i : WTi(W, i), range(len(W[0])) ) )

'''
Nilai yang masuk ke neuron di hidden layer adalah penjumlahan antara perkalian␣
 ↪weight dengan
```

```
nilai yang masuk pada input neuron.
'''
def XW(X,W):
    return map( lambda w: ft.reduce( lambda a,b:a+b, map( lambda xx,vw: xx*vw,␣
 ↪X, w), 0), WT(W) )

''' Mengaktivasi nilai yang didapat pada XW '''
def input_to_hidden(X, W):
    return list( map( lambda x:aktivasi(x) , XW(X, W) ) )

''' membuat feed-forward dari fungsi yang sudah dibuat di atas, supaya modular␣
 ↪'''
@calc_time_decorator
def feed_forward(X, W, M, H):
    return input_to_hidden(input_to_hidden(input_to_hidden(X, W), M), H)
```

[5]:
```
X = [ 0, 8, 1]
forw_prop = feed_forward(X, W, M, H)
print(f'Output: {forw_prop}')
```

```
Time:  0.00011444091796875
Output: [-0.31466743954210885]
```

Buatlah decorator untuk menambahkan perhitungan running time

[3]:
```
import time

def calc_time_decorator(func):
    def inner(*args, **kwargs):
        start = time.time()
        res = func(*args, **kwargs)
        end = time.time()
        print('Time: ', end - start)
        return res
    return inner
```

[ ]:
```
@calc_time_decorator
```

2 Jurnal - Abaikan saja

[ ]:
```
import math
import functools as ft

''' Fungsi aktivasi '''
def aktivasi(x):
    return 1/ (1+ math.exp(-x))
```

```
'''
Mendapatkan nilai W di setiap list sesuai dengan index.
Mapping W menjadi satu dimensi
'''
def WTi(W, i):
    return list(map( lambda w:w[i], W))

'''
Menampung WTi sesuai index dan membuat menjadi satu list
'''
def WT(W):
    return list( map( lambda i : WTi(W, i), range(len(W[0])) ) )

'''
Nilai yang masuk ke neuron di hidden layer adalah penjumlahan antara perkalian␣
 ↪weight dengan
nilai yang masuk pada input neuron.
'''
def XW(X,W):
    return map( lambda w: ft.reduce( lambda a,b:a+b, map( lambda xx,vw: xx*vw,␣
 ↪X, w), 0), WT(W) )

''' Mengaktivasi nilai yang didapat pada XW '''
def input_to_hidden(X, W):
    return list( map( lambda x:aktivasi(x) , XW(X, W) ) )

''' membuat feed-forward dari fungsi yang sudah dibuat di atas, supaya modular␣
 ↪'''
def feed_forward(X, W, M):
    return input_to_hidden(input_to_hidden(X, W), M)
```

[ ]:
```
X = [ 9, 10, -4 ]
W = [ [ 0.5, 0.4 ] , [ 0.3, 0.7 ] , [ 0.25, 0.9 ] ]
M = [ [ 0.34 ], [0.45] ]

feed_forward(X, W, M)
```

[ ]:
```
[0.6876336740661236]
```

[ ]:
```
input_to_hidden(X, W)

print('1')
print(list(XW(X,W)))
print(input_to_hidden(X,W))

print('2')
print(list(XW(input_to_hidden(X,W),W)))
```

```
print(input_to_hidden(input_to_hidden(X, W), M))
```

```
1
[6.5, 7.0]
[0.998498817743263, 0.9990889488055994]
2
[0.7989760935133112, 1.0987617912612246]
[0.6876336740661236]
```