1. Diberikan suatu barisan bilangan bulat $a_1, a_2, \dots, a_n$ dengan $n \geq 2$.
   a. Buatlah algoritma untuk mencari bilangan bulat terbesar kedua dari baris di atas
   b. Tentukan jumlah operasi perbandingan pada algoritma pada soal (a) sebagai suatu fungsi dalam n, misalnya T(n). Selanjutnya, tentukan big-Oh dari T(n) tersebut.

==========

a. Input: $a = a_1, a_2, \dots, a_n$

	T	Jumlah
Max1 = max(a_1, a_2)	t1	1
Max2 = min(a_1, a_2)	t2	1
for i = 3 to n	t3	n - 2
if $a_i >$ Max2	t4	n - 2
if $a_i >$ Max1	t5	n - 2
Max2 = Max1	t6	n - 2
Max1 = a_i	t7	n - 2
else		
Max2 = a_i	t8	n - 2
endif		
endif		
endfor		
Output: Max2	t9	1

b. Maka, running time atau T(n) dari algoritma tersebut:
   T(n) = $(t3 + t4 + t5 + t6 + t7 + t8)(n - 2) + (t1 + t2 + t9)$

   Untuk menentukan big-O dari T(n), perhatikan bahwa terjadi kasus terburuk apabila a_i selalu lebih besar dari Max1, di mana jika itu terjadi maka terjadi 5 proses $(n - 2)$. Kemudian jumlahkan itu dengan operasi basic sebanyak 3.
   O(T(n)) = 5(n – 2) + 3 = 5n – 10 + 3 = 5n – 7 = O(n)

   $\therefore T(n) \in O(n)$

2. Trace dari suatu matrix bujur sangkar A didefinisikan sebagai jumlah dari semua elemen diagonal dari A.
   a. Buatlah suatu algoritma untuk menentukan trace dari suatu matrix bujur sangkar A berukuran nxn.
   b. Tentukan big-Oh dari algoritma yang anda rancang.

==========

a. Input: $A = [[a_{11}, a_{12}, \dots, a_{1n}], [a_{21}, a_{22}, \dots, a_{2n}], \dots, [a_{n1}, a_{n2}, \dots, a_{nn}]]$

	T	**Jumlah**
Sum = 0	t1	1
for i = 1 to n	t2	n
Sum = Sum + $A[i, i]$	t3	n
endfor		
Output: Sum	t4	1

b. O(T(n)) = 2(n) + 2 = 2n + 2 = O(n)

$\therefore T(n) \in O(n)$

3. Gunakan definisi big-Oh untuk menunjukan bing-Oh dari fungsi-fungsi berikut ini :
   a. T(n) = 1^3+ 2^3+….+n^3 adalah O(n^4)
   b. T(n) = (6n-4n^5-4)/(7n^2-3) adalah O(n^3)
   c. T(n) = $(x+2)^2$ log (x^2+1)+ ^{2}log (x^3+1) adalah O(x^2 log x)

==========

a. $\exists c, n_0 \ni \forall n > n_0 : T(n) \leq c \,.\, n^4$
   Perhatikan bahwa $1^3 + 2^3 + \cdots + n^3 \leq n^3 + n^3 + \cdots + n^3 = n(n^3) = n^4$
   Ambil $n_0 = 1$ dan $c = 1$, maka terpenuhi, sehingga $T(n) \in O(n^4)$

b. $\exists c, n_0 \ni \forall n > n_0 : T(n) \leq c \,.\, n^3$
   Ambil $n_0 = \frac{1}{4}$ dan $c = 1$, maka terpenuhi, sehingga $T(n) \in O(n^3)$

c. Mengasumsikan maksud soal adalah T(x), maka dapat dikerjakan sebagai berikut.

$$
\begin{aligned}
(x+2)^2 \log(x^2+1) + {}^2\log(x^3+1) &= (x+2)^2 \log(x^2+1) + \frac{({}^2\log 10)}{({}^2\log 10)} \times {}^2\log(x^3+1) \\
&= (x+2)^2 \log(x^2+1) + {}^2\log 10 \times \log(x^3+1) \\
&\leq (x+x)^2 \log(x^2+x^2) + {}^2\log 10 \times \log(x^3+x\text{^}3) \\
&\leq 4x^2(\log 2 + 2\log x) + 4(\log 2 + 3\log x) \\
&\leq 12x^2 \log x + 16 \log x \\
&\leq 12x^2 \log x + 16x^2 \log x \\
&\leq 28x^2 \log x
\end{aligned}
$$

   Menggunakan definisi:
   $\exists c, x_0 \ni \forall x > x_0 : T(x) \leq c \,.\, x^2 \log x$
   Ambil $x_0 = 2$ dan $c = 28$, maka terpenuhi, sehingga $T(x) \in O(x^2 \log x)$

4. Buatlah suatu algoritma untuk menemukan p-norm dari suatu vector x berukuran n yang didefinisikan sbb :

$$\left|\left|x\right|\right|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}}$$

==========

Input:
- $x = [x_1, x_2, \dots, x_n]$
- p

Sum = 0
for i = 1 to n
    Sum = Sum + $|x_i|^p$
endfor
Norm = $(\text{Sum})^{\frac{1}{p}}$

Output: Norm

5. Algoritma binary search adalah algoritma pencarian (search) suatu nilai pada suatu barisan bilangan bulat terurut dengan cara membagi dua daerah pencarian pada setiap iterasi.
   a. Tunjukan bagaimana algoritma binary search mencari nilai 27 dari barisan berikut: 5,6,8,12,15,21,25,31.
   b. Tentukan jumlah operasi perbandingan pada algoritma binary search sebagai suatu fungsi dari banyaknya bilangan pada barisan, missal T(n). Selanjutnya, tentukan bing-Oh dari T(n) tersebut.

==========

a. Iterasi 1:
   Data terbagi menjadi 5, 6, 8, 12 dan 15, 21, 25, 31
   Algoritma kemudian melihat nilai terakhir dari barisan pertama
   Karena 27 > 12, maka barisan yang digunakan adalah 15, 21, 25, 31

   Iterasi 2:
   Data terbagi menjadi 15, 21 dan 25, 31
   Algoritma kemudian melihat nilai terakhir dari barisan pertama
   Karena 27 > 21, maka barisan yang digunakan adalah 25, 31

   Iterasi 3:
   Data terbagi menjadi 25 dan 31
   Algoritma kemudian melihat nilai terakhir dari barisan pertama
   Karena 27 > 25, maka barisan yang digunakan adalah 31

   Data hanya memiliki 1 elemen dan karena elemen tersebut ≠ 27, maka binary search telah selesai melakukan iterasi dan tidak mengeluarkan output

   Output: *None*

b. $T(n) = (t1 + t2 + t3)(\lfloor \log_2 n \rfloor) + t4$
   $O(T(n)) = O(\log_2 n)$

   $\therefore T(n) \in O(\log_2 n)$

6. Untuk x ≥ 0, fungsi sin(x) dapat diaproksimasikan dengan menggunakan:

$$\sin(x) \approx x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots + (-1) * \frac{x^{2n-1}}{(2n-1)!}$$

==========

Terdapat kesalahan dalam approksimasi yang digunakan untuk sin(x) jika (-1) karena itu berarti setiap suku adalah negatif, sehingga dilakukan pengubahan di bawah asumsi menjadi:

$$\sin(x) \approx x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots + (-1)^{n-1} * \frac{x^{2n-1}}{(2n-1)!}$$

Input: x, n

Sin = 0
for $i = 1$ to n

$$\text{Sin} = \text{Sin} + (-1)^{i-1} \frac{x^{2i-1}}{(2i-1)!}$$

endfor

Output: Sin

7. Diberikan algoritma berikut :

```
Procedure AMAL (A, n)
Integer I,j,n, A[j]
For i ← 1 to n
    J ← n
    While I < j do
            If A [j] > A [j-1] then
                    S ← A [j]
                    A [j] ← A [j-1]
                    A [j-1] ← S
            End if
    End while
Next i
End procedure.
```

Telusuri prosedur diatas untuk mendapatkan hasilnya pada iterasi ke 3, jika nilai n = 7 dan array A memuat data sebagai berikut :

5	2	4	1	3	6	9

==========

Dalam algoritma ini terjadi infinite looping sehingga tidak bisa dijawab sebagaimana ada, maka dilakukan sedikit modifikasi pada algoritma yang diberikan sebagai asumsi algoritma yang dimaksudkan.

```
Procedure AMAL (A, n)
Integer I,j,n, A[j]
for i ← 1 to n
    J ← n
    while I < j do
            If A [j] > A [j-1] then
                    S ← A [j]
                    A [j] ← A [j-1]
                    A [j-1] ← S
                endif
            Next I
    endwhile
endfor
endprocedure
```

Proses yang terjadi berlangsung sebagai berikut.

Iterasi 1:
- 2 5 4 1 3 6 9
- 2 4 5 1 3 6 9
- 2 4 1 5 3 6 9
- 2 4 1 3 5 6 9

Iterasi 2:
- 2 1 4 3 5 6 9
- 2 1 3 4 5 6 9

Iterasi 3:
- 1 2 3 4 5 6 9

Output: 1 2 3 4 5 6 9

8. Buatlah algoritma untuk mengurangi matriks P dan matriks Q yang hasilnya di simpan pada matriks R.

==========

Input:
- $P = [[p_{11}, p_{12}, \dots, p_{1n}], [p_{21}, p_{22}, \dots, p_{2n}], \dots, [p_{k1}, p_{k2}, \dots, p_{kn}]]$
- $Q = [[q_{11}, q_{12}, \dots, q_{1n}], [q_{21}, q_{22}, \dots, q_{2n}], \dots, [q_{k1}, q_{k2}, \dots, q_{kn}]]$

```
for i = 1 to k
    for j = 1 to n
            R[i, j] = P[i, j] – Q[i, j]
    endfor
endfor
```

Output: R

Afterword

Pembuatan dokumen ini dibantu oleh:

1. Takamori37 (Matematika UI 2016, D4 Akuntansi PKN STAN 2017)