Complexity TD Series Za3ma

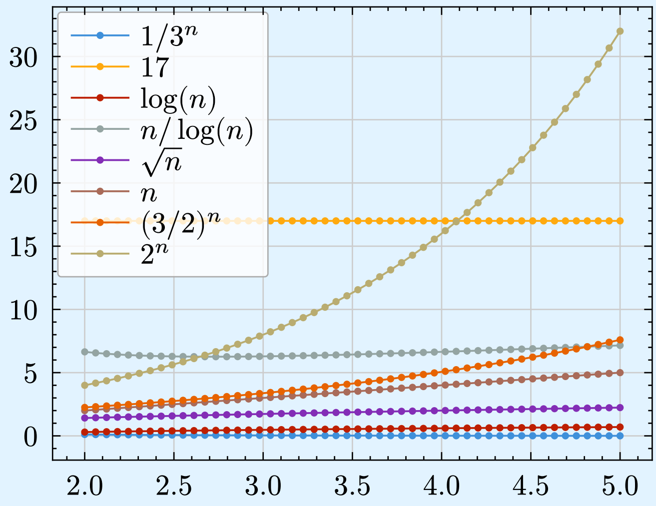
Exercise 1: *Asymptotic complexity of the functions*

- $6n^3 + 10n^2 + 5n + 2 \to \mathrm{O}(n^3)$.
- $3\log_2(n) + 4 \to \mathrm{O}(\log_2(n))$.
- $2^n + 6n^2 + 7n \to \mathrm{O}(2^n)$.
- $7k + 2 \to \mathrm{O}(1)$.
- $4\log(n) + n \to \mathrm{O}(n)$.
- $2\log_{10}(k) + kn^2 \to n^2$

Exercise 2: *Compare the functions near infinity, for checking just use the fact that $|f(n)/g(n)|$ is bounded near infinity, then $f(n) = \mathrm{O}(g(n))$.*

$$\left(\frac{1}{3}\right)^n < 17 < \log(n) < \frac{n}{\log(n)} <$$

$$\sqrt{n} < n < \left(\frac{3}{2}\right)^n < 2^n$$



Exercise 3: *Find number of operations and complexity, given that we have no way to know what should be counted as a step or not, I will consider additions, declarations and evaluations to be elementary steps.*

- *A:* $1 + 2n^2 \to \mathrm{O}(n^2)$.
- *B:* $1 + 2 \cdot n \cdot 2n \to \mathrm{O}(n^2)$.
- *C:* $2n^2(n+1)/2 \to \mathrm{O}(n^3)$.

Exercise 4: *Find number of operations and complexity, same here, just rewriting the program because of the extremely bad idententation.*

- f_1*:*

```
x = 0;

for i = 0 to n - 1 do
  for j = 0 to i - 1 do
    x = x + 1;
  end
end
```

$$1 + 2n(n+1)/2 \to \mathrm{O}(n^2)$$

- f_2*:*

```
x = 0;
i = n;

while i > 1 do
  x = x + 1;
  i = i / 2;
end
```

$$2 + 4\log_2(n) \to \mathrm{O}(\log_2(n))$$

$\log_2(n)$ in this case from the fact that in each iteration, we halve how many steps we go through, thus, we need how many 2 divides n times to pass through all n, which is $\log_2(n)$ in this case ($2^k = n \Rightarrow k = \log_2(n)$).

- f_3*:*

```
x = 0;
i = n;

while i > 1 do
  for j = 0 to n - 1 do
    x = x + 1;
  end
  i = i / 2;
end
```

$$2 + \log_2(n)(2 + 2n) \to \mathrm{O}(n\log_2(n))$$

same reasoning here

- f_4*:*

```
x = 0;
i = n;

while i > 1 do
  for j = 0 to i - 1 do
    x = x + 1;
  end
  i = i / 2;
end
```

$$2 + \sum_{j=0}^{\lceil\log_2(n)\rceil} 4\frac{n}{2^j} \to \mathrm{O}(n))$$

You can try it for $i = 2^k$ for some k, you can see that at each iteration, there would be 2^k evaluations and additions, thus giving the result.

Exercise 5:

1.

```
S := 1
for i = 2 to n do
  S = S * i
```

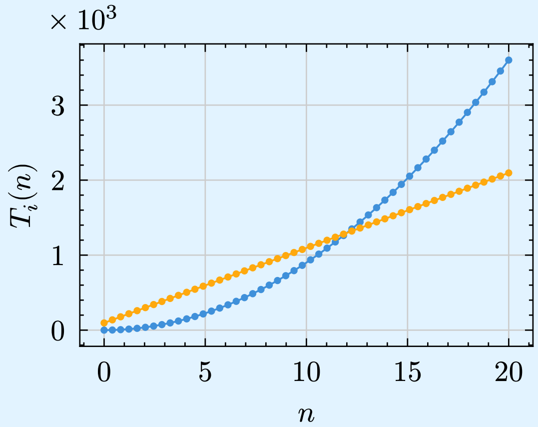
```
end
return S
```

2. *The complexity is* $1 + 2(n-1) = \mathrm{O}(n)$.

Exercise 6:

1. 1. $T_1(n) = 9n^2 = \mathrm{O}(n^2)$.
   2. $T_2(n) = 100n + 96 = \mathrm{O}(n)$.
2. 1. *For* T_1*:* $(n_0, c) = (1, 9)$ *clearly.*
   2. *For* T_2*:* $(n_0, c) = (1, 200)$ *since* $200n = 100n + 100n > 100n + 100 > 100n + 96$.
3.

$T_i \setminus n$	1	3	5	10	14
T_1	9	81	225	900	1764
T_2	196	396	596	1096	1496

4. 

5. *for* $n < 12$*:* T_1 *is better, otherwise* T_2 *is better.*
6. *calling the algorithms with complexity* T_1 *and* T_2 *we get an algorithm of complexity* $T_1 + T_2 + c$ *where* c *is a constant, and its complexity would be* $\mathrm{O}(n^2)$.

Exercise 7:

1.

```
input:
  - n: size
  - A: matrix of size n
  - B: matrix of size n
output:
  - C: matrix product of A*B

C = [][]
for i = 0 to n-1
  for j = 0 to n-1
    for k = 0 to n-1
      C[i][j] += A[i][k] * B[k][j]
    end
  end
end
return C
```

2. $T(n) = 3(n-1)^3 \to \mathrm{O}(n^3)$.