Introduction to algorithms Fall 2017

IFT2125-6001 TA: Maëlle Zimmermann

Demonstration 2

1

Question: Let $0 < \varepsilon < 1$. Use the relations $\subset$ and $=$ to order the O of functions and demonstrate the solution (using the theorems on the limits, see demonstration 1).

$$n \log n \quad n^{8} \quad n^{1+\varepsilon} \quad (1+\varepsilon)^{n} \quad n^{2}/\log n \quad (n^{2}-n+1)^{4}$$

Solution: The solution is

$$O(n \log n) \subset O(n^{1+\varepsilon}) \subset O(n^{2}/\log n) \subset O(n^{8}) = O((n^{2}-n+1)^{4}) \subset O((1+\varepsilon)^{n})$$

Each inclusion can be demonstrated point by point using the theorems seen on the limits and the rules of The Hospital.

(at)

$$\lim_{n\to\infty} \frac{n^{1+\varepsilon}}{n \log n} = \lim_{n\to\infty} \frac{n^{\varepsilon}}{\log n} = \lim_{n\to\infty} \frac{\varepsilon n^{\varepsilon - 1}}{1/n} = \lim_{n\to\infty} \underbrace{\varepsilon n^{\varepsilon}}_{\text{because } \varepsilon > 0} = +\infty .$$

(b)

$$\lim_{n\to\infty} \frac{n^{2}/\log n}{n^{1+\varepsilon}} = \lim_{n\to\infty} \frac{n^{1-\varepsilon}}{\log n} = \lim_{n\to\infty} \frac{(1-\varepsilon)\, n^{-\varepsilon}}{1/n} = \lim_{n\to\infty} \underbrace{(1-\varepsilon)\, n^{1-\varepsilon}}_{\text{because } 1-\varepsilon > 0} = +\infty .$$

(c)

$$\lim_{n\to\infty} \frac{\text{No.}^{8}}{n^{2}/\log n} = \lim_{n\to\infty} n^{6} \log n = +\infty .$$

(d) Instead of using the calculation of limits which would require
the definition of O (f (n)).

(n 2 - n + 1) n = 4 8 - 7 4n + 10n 6 - 16n, 19n + 5 4 - 16n 3 + 10n 2 - 4n + 1
8 ≤ n + 10n + 19n 4 + 6 10n 2 + 1
≤ n 8 + 8 + 10n 19n 10n + 8 8 + n 8
41n = 8.

And by definition (n 2 - n + 1) 4 ∈ O (n 8). Furthermore,

(n 2 - n + 1) n = 4 8 - 7 4n + 10n 6 - 16n, 19n + 5 4 - 16n 3 + 10n 2 - 4n + 1
N ≥ 8 - (4n + 7 5 + 16n 16n 3 + 4n)
N ≥ 8 - 40n 7
N ≥ 8 - (1/2) n 8 80n for n ≤ 7 8 ∀n ≥ 80
= (1/2) n 8

And 8 ≤ n 2 (n 2 - n + 1) 4 ≥ n 0 ∀ n = 80, n 8 and therefore ∈ O ((n 2 - n + 1) 4).

(e) Let b = 1 + ε.

lim n → ∞ b n / No. 8 = lim n → ∞ lnb · b n / 8n 7
= lim n → ∞ (LNB) 2 · b n / 8 · 6 7n
= ...
= lim n → ∞ (LNB) 8 · b n / 8!
= + ∞

2

Question: Let 0 <ε <1. Use the relations ⊂ and = to order the O of
functions and demonstrate the solution (using the theorems on
the limits, see demonstration 1).

not! (n + 1)! 2 n 2 n + 1 2 2n n n √ notnot n log n

Solution: The solution is

O (n log n) ⊂ O (n √ n) ⊂ O (2 n) = O (2 n + 1) ⊂ O (2n 2) ⊂ O (n!) ⊂ O ((n + 1)!) ⊂ O (n n)

A Parte: It is useful to show first that $\forall d, c \in \mathbb{R}_{>0}, {}_0 \exists n \in \mathbb{N}: \forall n \geq n_{0,} c (\log n) \leq {}_d$. Indeed, $\forall d, c \in \mathbb{R}_{>0}$ we have:

$$\lim_{n \to \infty} \frac{c \log n}{{}^d} = \lim_{n \to \infty} \frac{c / n}{dn^{d-1}} = \lim_{n \to \infty} \frac{c}{dn^d} = 0.$$

We can conclude (by the rigorous definition of a limit) that

$$\forall \varepsilon > 0 \ \exists n_0 \in \mathbb{N}: n \geq {}_0 \ \forall n, \quad \frac{c \log n}{d} < \varepsilon.$$

By taking $\varepsilon = 1$, we have proved the assertion.

Each inclusion is shown point by point as in the previous exercise.

(a) We have shown that $(\forall d, c \in \mathbb{R}_{>0}) \ \exists n_0 \in \mathbb{N}: \forall n \geq n_{0,} c (\log n) \leq {}_d$. So for sufficiently large n, we can limit log n by ${}_1$ $\sqrt[2]{}$ not:

$$0 \leq \frac{n^{\log n}}{\sqrt{\text{not}}\,\text{not}} \leq \frac{\text{not}^{\frac{1}{2}}\sqrt{\text{not}}}{\sqrt{\text{not}}\,\text{not}} = \frac{1}{\text{not}^{\frac{1}{2}}\sqrt{\text{not}}}$$

As

$$\lim_{n \to \infty} \frac{1}{\text{not}^{\frac{1}{2}}\sqrt{\text{not}}} = 0$$

we obtain that

$$\lim_{n \to \infty} \frac{n^{\log n}}{\sqrt{\text{not}}\,\text{not}} = 0$$

(b) For n sufficiently large:

$$\sqrt{\text{not}}\, n = 2 \log_2 n^{\sqrt{\text{not}}}$$

$$C = 2^{\log n \sqrt{\text{not}}} \qquad \text{because } \log_2 n = \log n / \log 2$$

$$\leq 2^{n^{\frac{3}{4}}} \qquad \text{because } c \log n \leq n^{1/4} \text{ for n suff. great}$$

Thus for n sufficiently large:

$$0 \leq \frac{\sqrt{\text{not}}\,\text{not}}{2^n} \leq \frac{2^{n^{\frac{3}{4}}}}{2^n} = \frac{2^{n^{\frac{3}{4}}}}{2^{n^{\frac{3}{4}} n^{\frac{1}{4}}}} = \frac{1}{2^{n^{\frac{3}{4}}(n^{\frac{1}{4}} - 1)}}$$

As

$$\lim_{n \to \infty} \frac{1}{2^{n^{\frac{3}{4}}(n^{\frac{1}{4}} - 1)}} = 0$$

we obtain that

$$\lim_{n \to \infty} \frac{\sqrt{\text{not}}\,\text{not}}{2^n} = 0.$$

(c) We proved this in the proof.

(d)

$$\lim_{n \to \infty} \frac{2^{2n}}{2^n} = \lim_{n \to \infty} \frac{4^n}{2^n} = \lim_{n \to \infty} \left(\frac{4}{2}\right)^{\text{No}} = \lim_{n \to \infty} 2^n = +\infty.$$

(e) For $n \geq 4$, we have

$$\frac{\text{not}!}{4^n} = \frac{n \cdot n - 1 \ldots 2 \cdot 1}{4 \cdot 4 \ldots 4 \cdot 4} \geq \frac{\text{not}}{4} \cdot \frac{3}{4} \cdot \frac{2}{4} \cdot \frac{1}{4}$$

since k / 4 ≥ 1 for 4 ≤ k ≤ n. As

$$\lim_{n \to \infty} \frac{not}{4} \cdot \frac{3}{4} \cdot \frac{2}{4} \cdot \frac{1}{4} = +\infty,$$

we obtain

$$\lim_{n \to \infty} \frac{not!}{4^n} = +\infty.$$

(f)

$$\lim_{n \to \infty} \frac{(n+1)!}{not!} = \lim_{n \to \infty} \frac{(n+1) \cdot n \cdot (n-1) \dots 2 \cdot 1}{n \cdot (n-1) \dots 2 \cdot 1} = \lim_{n \to \infty} n + 1 = +\infty.$$

(g) We have

$$\frac{n^n}{(n+1)!} = \frac{not}{2} \cdot \frac{not}{3} \cdots \frac{not}{n+1} \geq \frac{not}{4}$$

because n / k ≥ 1 for 3 ≤ k ≤ n and n / (n + 1) ≥ 1/2. As

$$\lim_{n \to \infty} \frac{not}{4} = +\infty$$

we obtain

$$\lim_{n \to \infty} \frac{n^n}{(n+1)!} = +\infty$$

3

Question: Give evidence explicitly with two function f, g: $N \to R_{\geq 0}$ such as f / ∈ O (g) and g / ∈ O (f).

Solution:

$$f(n) = \begin{cases} n & \text{if n is even} \\ 1 & \text{otherwise} \end{cases} \quad \text{and} \quad g(n) = \begin{cases} 1 & \text{if n is even} \\ n & \text{otherwise} \end{cases}$$

Suppose that f ∈ O (g (n)). Then there exists $n \in N_0$, $c \in R_{>0}$ such that f (n) ≤ cg (n), ∀ $n \geq n_0$. Let k = 2 max (d $_0$, ⌈c⌉). Then as k is even

$$f(k) = 2\,k = \max(d_0, \lceil c \rceil) > cg = c(k).$$

This is a contradiction because $k \geq n_0$. We conclude that f / ∈ O (g).

In a symmetric way, one can prove that g / ∈ O (f).

4

Question: Execute the intelligent permutation algorithm to determine if (1345) belongs to the set of permutations generated by {(12), (12345)}.

Solution: Here is the trace of the execution of the algorithm.

initialization

	1	2	3	4	5
1	()	()	()	()	()
2	()	()	()	()	()
3	()	()	()	()	()
4	()	()	()	()	()
5	()	()	()	()	()

Initial sieving

The permutations (12) and (12345) are sieved.

(12)	insert in [1,2]
(12345)	(12) already in [1,2]
(12345) (21) = (2345)	insert in [2,3]

	1	2	3	4	5
1	()	(12)	()	()	()
2	()	()	(2345)	()	()
3	()	()	()	()	()
4	()	()	()	()	()
5	()	()	()	()	()

Iteration 1

The products of all pairs of permutations in the table are sieved.

(12) (12) = id	nothing to do
(12) (2345) = (13452)	insert in [1,3]
(2345) (12) = (12345)	already sifted
(2345) (2345) = (24) (35)	insert in [2,4]

	1	2	3	4	5
1	()	(12)	(13452)	()	()
2	()	()	(2345)	(24) (35)	()
3	()	()	()	()	()
4	()	()	()	()	()
5	()	()	()	()	()

Iteration 2

The new permutations in the table are (24) (35) and (13452). Note: there are two versions of the algorithm. Either we sift the products of the new possible pairs between two permutations of the table. Either we tamisons the form p p_i · products or p_i · po`up is a new permutation table and an initial permutation $p_{i.}$
In the iteration that follows it is the second version which is applied because it makes less than products. Note also that it is the same to consider (12345) or (2345) as the initial permutation because it is the same table that results from it after sieving initial.

(12) (13452) = (2345) already in the table
(13452) (12) = (1345) (13452) already in [1,3]
(1345) (25431) = (25) insert in [2,5]
(12) (24) (35) = (142) (35) insert in [1,4]
(24) (35) (12) = (124) (35) (12) already in [1,2]
(124) (35) (21) = (24) (35) already in the table
(2345) (13452) = (135) (24) (13452) already in [1,3]
(135) (24) (25431) = (2345) already in the table
(13452) (2345) = (142) (35) already in the table
(2345) (24) (35) = (2543) (25) already in [2.5]
(2543) (52) = (345) insert in [3,5]
(24) (35) (2345) = (2543) already sifted

	1	2	3	4	5
1	()	(12)	(13452)	(142) (35)	()
2	()	()	(2345)	(24) (35)	(25)
3	()	()	()	()	(354)
4	()	()	()	()	()
5	()	()	()	()	()

Iteration 3

(12) (25) = (152) Insert into [1,5]
(24) (35) (25) = (2453) (24) (35)
(2453) (53) (42) = (34) insert in [3,4]
(25) (24) (35) = (2354) (2345) already in [2,3]
(2354) (5432) = (345) (34) already in [3,4]
(345) (43) = (45) insert in [4,5]

	1	2	3	4	5
1	()	(12)	(13452)	(142) (35)	(152)
2	()	()	(2345)	(24) (35)	(25)
3	()	()	()	(34)	(354)
4	()	()	()	()	(45)
5	()	()	()	()	()

No box at the top of the diagonal contains the identity, there is no more box to rem- and it is therefore unnecessary to continue. We conclude that {(12) (12345)} generates S $_5$. Here we can easily deduce that (1345) belongs to the generated set by {(12), (12345)}. In cases where the table is not fully filled, it would be necessary to sieve the permutation in the table to answer the question: if sieve the permu- tation modifies the array, it does not belong to the set generated by the permutations initials.

We can implement the algorithm in python as follows.

```
def product (p, q):
    return tuple (q [p [i] - 1] for i in range (len (p)))

reverse def (p):
    q = [0] * len (p)
    for i in range (len (p)):
        q [p [i-1]] = i + 1
    q return

def sift (Table, p):
    IDENTITY = tuple (range (1, len (p) +1))
```



```
    q = p

    while q! = IDENTITY
        i = min (x for x in range (len (q))) if q [x]! = x + 1)
        j = q [i] - 1
        if array [i] [j] == IDENTITY:
            array [i] [j] = q
            q return
        else:
            q = product (q, inverse (array [i] [j]))
    return None

def appartenance_intelligent (permutations, r):
    IDENTITY = tuple (range (1, len (r) +1))
    table = [[IDENTITY] * len (r) for _ in range (len (r))]

    # Initial sifting / Initial sift
    for p in permutations:
        sift (table, p)

    # Fill table / Fill table
    to_sift = [(p, q) for p in permutations for q in permutations]

    while len (to_sift)> 0:
        p, q = to_sift.pop ()
        q = sift (array, product (p, q))

        if q is not None:
        # q is a new permutation added to the array
            to_sift.extend ([(p, q) for p in permutations])
            to_sift.extend ([(q, p) for p in permutations])

    # Genere r? / Generates r?
    sift return (table, r) is None

# Example / Example
a = tuple ([2, 1, 3, 4, 5]) # (12) (3) (4) (5)
b = tuple ([2, 3, 4, 5, 1]) # (12345)
tuple r = ([2, 1, 4, 5, 3]) # (12) (345)

print (appartenance_intelligent (set ([a, b]), r))
```