2021-09-17 - Pearls of Computer Science Core - Algorithmics

Course: B-CS-MOD01-1A-202001021/202001022 B-CS Pearls of Computer Science Module - 202001021/202001022

Contents:

Pages:

- A. Front page **1**
- B. Questions.. **7**
- C. Correction model **11**

Duration:	1 hour
Generated on:	Sep 22, 2021

2021-09-17 - Pearls of Computer Science Core - Algorithmics

Course: B-CS-MOD01-1A-202001021/202001022 B-CS Pearls of Computer Science Module - 202001021/202001022

Welcome to the digital exam for Pearl 001 Algorithmics.

- You may use **1 A4 sheet with your own notes** for this test, as well as a simple calculator
- **Scientific or graphical calculators, laptops, mobile phones, books etc. are not allowed. Put those in your bag now (with the sound switched off)**
- Please use the **BBB/Canvas chat for any question** during the exam.

Total number of points: 100

1
5 pt.

Question 1a

Suppose you execute the following assignments in Python

```
clubs = ["Dance", "Theatre", "Sports"]
members = [68, 162, 92]
```

Here *clubs* is a list of club names at the University of Twente, and *members* is a list of the corresponding (fictional) number of active members in each club.

Write a Python condition (*not* an **if** statement) that tests whether the "Dance" club is the smallest of the three clubs.

2
5 pt.

Question 1b

Suppose you execute the following assignments in Python

```
clubs = ["Dance", "Theatre", "Sports"]
members = [68, 162, 92]
```

Here *clubs* is a list of club names at the University of Twente, and *members* is a list of the corresponding (fictional) number of active members in each club.

Assign to a new list '*smallest*' both the name and the number of active members of the club with the fewest active members.

3
5 pt.

Question 1c

Suppose you execute the following assignments in Python

```
clubs = ["Dance", "Theatre", "Sports"]
members = [68, 162, 92]
```

Here *clubs* is a list of club names at the University of Twente, and *members* is a list of the corresponding (fictional) number of active members in each club.

Write a sequence of assignments that is **as short as possible**, resulting in a change to '*members*' after which the number of active members are ordered from lowest to highest.

(NB: It is *not* correct to assign an entirely new value to *members*. You *must* modify the list by swapping elements.)

4 Question 2

Consider the following Python function:

```
def compute(data)
   i, j = 0, 0
   result = data[i]

   while i < len(data):
       if data[i] < result:
           result = data[i]

       while j < i:
           if data[j] < result:
               result = data[j]
           j = j + 1

       i = i + 1

   return result
```

5 pt. **a.** What is the **return value** upon calling compute([99, -8, 8, 20, 0])?

5 pt. **b.** What is the **return value** upon calling compute(["magic", "tanja", "hamster", "companion"])?

5 pt. **c.** Using your own words, what does the algorithm do?

Hint: Do *not* list line-by-line what the algorithm does, but state what the function *compute* does as a whole.

5
10 pt.

Question 3

Consider again the Python function given in Question 2:

```
def compute(data)
    i, j = 0, 0
    result = data[i]

    while i < len(data):
        if data[i] < result:
            result = data[i]

        while j < i:
            if data[j] < result:
                result = data[j]
            j = j + 1

        i = i + 1

    return result
```

Assume we input a list *data* of length n . How many steps does *compute* need to finish?

1. Approximately n

2. Approximately n^2

3. Approximately $n \cdot log_2 n$

4. Approximately $\sqrt{n}$

Motivate your answer.

6

Question 4

Consider again the Python function given in Question 2:

```
01. def compute(data)
02.    i, j = 0, 0
03.    result = data[i]
04.
05.    while i < len(data):
06.        if data[i] < result:
07.            result = data[i]
08.
09.        while j < i:
10.            if data[j] < result:
11.                result = data[j]
12.            j = j + 1
13.
14.        i = i + 1
15.
16.    return result
```

5 pt. **a.** How does the **return value** of the algorithm change if we replace **each** occurrence of "<" with a ">" instead, ie. if we replace **each** occurrence of "smaller than", with a "larger than"?

5 pt. **b.** How does the **return value** of the algorithm change if we delete lines 09 to 12?

5 pt. **c.** How does the **complexity** of the algorithm change if we delete lines 09 to 12? Motivate your answer.

7
10 pt.

Question 5a

Consider the following list

```
[8, 20, -4, 4, -18, 6]
```

Show how bubble sort sorts this list by writing down the list after every single modification.

8
10 pt.

Question 5b

Consider the following list

```
[8, 20, -4, 4, -18, 6]
```

Show how merge sort sorts this list by presenting how the list is split and zipped back together.

Write down every change the algorithm makes to the list(s) in a new line.

9
10 pt.

Question 6

After having studied all week for the Algorithmics test you lie awake one night and wonder if one could get an even faster search algorithm than binary search.

Then it dawns on you: *ternary search!* Rather than dividing a list into two equal parts, *ternary search* divides the list into *three* equal parts (left, middle, right) and then determines in which of these three parts the searched-for value lies.

Can this idea be realized, ie. could such an algorithm work in practice?

If not, why not?

If so, what would be the complexity of *ternary* search -- In particular, would it be fundamentally faster than binary search? Explain your answer.

10

Question 7

After an exciting first 10 weeks of studying at the University of Twente you accumulated a pile of books. You stacked them such that the largest book is at the very bottom, and the smallest at the very top -- A pyramid of books!

One fine day you decide to shift the entire book-pyramid to another location in your room -- But alas! The pile is too heavy.

There is no other solution: You must re-locate the books one-by-one. To make for a stable pile of books after the re-location, the biggest book needs to be at the bottom once again.

10 pt. **a.** (a) Write an algorithm in natural language with **unambiguous** and **numbered** instructions that re-locates all the books to another place one-by-one, such that the final stack has the same shape as before.

You may assume that there is enough space in your room to spread out books and sort them all you like.

*Do **not** give an answer in Python!*

5 pt. **b.** (b) What is the complexity of your algorithm in (a) in terms of the number of books n .

Thank you, your answers were saved. We'll inform you about your result once every test was corrected.

Correction model

1.
5 pt.

Correction criterion	Points
-2 if an "if" is included -2 for missing 'and' -3 for the wrong list (clubs vs members) -3 if the largest club was used instead of the smallest (< vs >) -1 if the same element was used for comparison and everything else is correct, eg. members[0] < members[1] and members[0] < members[1] -2 for syntax mistakes, eg. =< instead of <= Answer: members[0] < members[1] and members[0] < members[2]	5 points
Total points:	*5 points*

2.
5 pt.

Correction criterion	Points
This often yields 0 points if it is not quite correct. -4 if 'append' or 'extend' are used -3 for wrong or missing brackets -1 for one missing bracket -3 for absence of any index Answer: smallest = [clubs[0], members[0]]	5 points
Total points:	*5 points*

3.
5 pt.

Correction criterion	Points
0 if capacity is assigned a new list -2 for every additional assignment beyond the first -2 for wrong order -3 for wrong list 'clubs' -2 missing comma(s) Answer: members[1], members[2] = members[2], members[1]	5 points
Total points:	*5 points*

4.
15 pt.

a.

Correction criterion	Points
Answer: -8	5 points
Total points:	*5 points*

b.

Correction criterion	Points
Answer: companion NB: Adding " " around the answer can be ignored. In particular, the answer "an error occurs" is not correct. This should be clear from exercise 2.6 of the course manual.	5 points
Total points:	*5 points*

c.

Correction criterion	Points
Answer: The algorithm returns (one of) the minimum value(s) of a list. Any answer that is syntactically equivalent is acceptable for full points here, eg. "returns the smallest value" or "returns the minimum" Notably, the algorithm works on integers, as well as strings (or any mutually comparable objects).	5 points
Total points:	*5 points*

5.
10 pt.

Correction criterion	**Points**
NB: The intention of the question was to have an algorithm that runs with a complexity of n^2 by having a nested while-loop, where both conditions depend on len(data). However, due to a missing reset of the inner counter variable j the algorithm runs in linear complexity n. This is a mistake on the side of the creator of the exam. For the correction of this exam we decided to treat both answers (n and n^2) as correct *if* the explanation was sufficient. In particular, that means explanations akin to * The complexity is n^2 because of the nested loop depending on len(data) * The complexity is n, because the outer while loop depends on len(data), while the inner loop only ever does 1 additional step for j < i, which happens once per iteration * Note in particular that the answer 2n is "approximately" n as it was defined in the course notes, ie. the complexity is the behaviour of the function in the limit Stating a complexity without proper motiviation often yields 0 points. Similarl applies for stating that the algorithm executes every line exactly once, hence the complexity is n. The motivation needs to indicate that the outer while loop (with counter i) depends on len(data), while the inner loop (with counter j) does not without the reset j=0.	10 points
Total points:	*10 points*

6.
15 pt.

a.

Correction criterion	**Points**
The algorithm always returns the first entry of the list 'data', since *each* occurrence means the while loop of line 5 is skipped. In particular, the algorithm does *not* return the maximum instead.	5 points
Total points:	*5 points*

b.

Correction criterion	**Points**
Note that a wrong motivation deducts up to 3 points, even if the answer is correct. Answer: The return value does not change. The deleted lines are superfluous.	5 points
Total points:	*5 points*

c.

Correction criterion	**Points**
Answer: The complexity goes from "approximately n^2" to "approximately n". This is due to deleting the nested while loop that depended on len(data). NB: Due to the missing reset of the counter variable j=0 inside the first while loop (with variable i as counter) answers akin to: * the complexity of the algorithm is still n * the complexity of the algorithm will be reduced, but still be linear are also correct *if* question 3 was answered appropriately. Please double check the correction model or question 3. Additionally: * it is not sufficient to say "the function will be less complex because 3 lines of code are removed". This results in 0 points * it is not sufficient to say "the complexity will be halved". This results in 3/5 points	5 points
Total points:	*5 points*

7.
10 pt.

Correction criterion	**Points**
The question is assessed strictly: -3 for every mistake, eg. skipping a step up to -8 for a systematic error [8, 20, -4, 4, -18, 6] [8, -4, 20, 4, -18, 6] [8, -4, 4, 20, -18, 6] [8, -4, 4, -18, 20, 6] [8, -4, 4, -18, 6, 20] [-4, 8, 4, -18, 6, 20] [-4, 4, 8, -18, 6, 20] [-4, 4, -18, 8, 6, 20] [-4, 4, -18, 6, 8, 20] [-4, -18, 4, 6, 8, 20] [-18, -4, 4, 6, 8, 20]	10 points
Total points:	*10 points*

8.
10 pt.

Correction criterion	**Points**
The question is assessed strictly. -3 if pairs are not split -5 if zipping is not included -5 if lists are mixed up -3 if the call stack is resolved erroneously, eg. merging elements 3 and 4 up to -8 for systematic mistakes Answer: The differing recursion depth is indicated by * and - [8, 20, -4, 4, -18, 6] is split into [8, 20, -4] and [4, -18, 6] *[8, 20, -4] is split into [8] and [20, -4] ** [20, -4] is split into [20] and [-4] ** [20] and [-4] are merged to [-4, 20] * [8] and [-4, 20] are merged to [-4, 8, 20] - [4, 6, -18] is split into [4] and [6, -18] -- [6, -18] is split into [6] and [-18] -- [6] and [-18] are merged to [-18, 6] - [4] and [-18, 6] are merged to [-18, 4, 6] [-4, 8, 20] and [-18, 4, 6] are merged to [-18, -4, 4, 6, 8, 20]	10 points
Total points:	*10 points*

9.
10 pt.

Correction criterion	**Points**
Points are given for the explanation rather than the precise answer. Points are awarded leniently as long as the argument makes sense (ie. no precise calculation like below is necessary). A sensible explanation may yield up to 8 points even if the answer is not correct. Conversely, just writing "yes" or "no" does not yield more than 2 points, even if it is correct. Answer: Yes, this algorithm can be made to work. It's complexity would be c_3*log_3(n) in the length n of the list, compared to c_2*log_2(n) for binary search. Here c_2 and c_3 are constants. Even though the algorithm exists, it is not fundamentally faster than binary search. Going from binary to ternary the speedup is (c_2 * log_2(n))/ (c_3 * log_3(n)) = (c_2/c_3) log_2(3) = C Which is a constant (as the ratio of two logarithms with different bases is constant). Depending on the exact constants it is even very likely to be smaller than 1, implying that ternary is actually a slowdow. Determining if the searched-for value lies in left, middle or right requires two comparisons in the worst case. With the same number of comparisons binary search always divides the list into four compartments.	10 points
Total points:	*10 points*

10.
15 pt.

a.

Correction criterion	**Points**
-2 if the algorithm is not numbered. -2 to -6 if the algorithm is ambiguous -8 (to -10) if more than one book is used in the instructions, eg. "1. I am strong and therefore can lift the entire stack from A to B" Answer: Many solutions exist. One example-algorithm could look like this: 1. Take the topmost book from the original pile 2. Place the book from step 1 on an intermediate pile 3. If there are still books on the original pile go to step 1, else go to step 4 4. Take the topmost book from the intermediate pile 5. Place the book from step 4 on the target pile 6. If there are still books on the intermediate pile go to step 4, else we are done NB: With the additional restriction that the space is limited and that only smaller books can sit on top of larger books this puzzle becomes the famous puzzle "Tower of Hanoi" which is often used to explain recursion.	10 points
Total points:	*10 points*

b.

Correction criterion	**Points**
Points are given if the answer reflects the steps of the algorithm in (a) correctly, even if (a) does not accomplish the task originally asked for. Answer: For example, the algorithm above takes 6n steps, as each book needs to be taken away from the pyramid until the largest book at the bottom is moved, and then re-stacked together. n is the number of books and there are 6 instructions to follow.	5 points
Total points:	*5 points*

Caesura

Points scored	Grade
100	10
99	9.9
98	9.8
97	9.7
96	9.6
95	9.5
94	9.4
93	9.3
92	9.2
91	9.1
90	9.0
89	8.9
88	8.8
87	8.7
86	8.6
85	8.5
84	8.4
83	8.3
82	8.2
81	8.1
80	8.0
79	7.9
78	7.8
77	7.7
76	7.6
75	7.5
74	7.4
73	7.3
72	7.2
71	7.1
70	7.0
69	6.9
68	6.8
67	6.7
66	6.6
65	6.5
64	6.4
63	6.3
62	6.2
61	6.1
60	6.0
59	5.9
58	5.8
57	5.7
56	5.6
55	5.5
54	5.4
53	5.3
52	5.3
51	5.2
50	5.1
49	5.0
48	4.9
47	4.8
46	4.8
45	4.7
44	4.6
43	4.5
42	4.4
41	4.4
40	4.3

39	4.2
38	4.1
37	4.0
36	3.9
35	3.9
34	3.8
33	3.7
32	3.6
31	3.5
30	3.5
29	3.4
28	3.3
27	3.2
26	3.1
25	3.0
24	3.0
23	2.9
22	2.8
21	2.7
20	2.6
19	2.6
18	2.5
17	2.4
16	2.3
15	2.2
14	2.1
13	2.1
12	2.0
11	1.9
10	1.8
9	1.7
8	1.7
7	1.6
6	1.5
5	1.4
4	1.3
3	1.2
2	1.2
1	1.1
0	1.0

Question identifiers

These identifiers can be used to track the exact origin of the question. Use these identifiers together with the identifier of this document when sending in comments about the questions, so that your comment can be connected precisely with the question you are referring to.

Document identifier: 6454-8174

Question number	Question identifier	Version identifier
1	69019	b893a1f8-49f3-e81d-4b3a-9b6edbb162c9
2	69024	bd98a2d4-a1bf-71dc-b4f0-f57a270294f2
3	69029	2426ba42-402f-52b9-b73d-365ff8870e03
4	69054	82add0f7-792a-8ffa-85c2-6c293bd8138a
5	69059	2770dbc8-bcc7-f808-5895-bfc9109b4f74
6	69064	8276b730-5cc5-bd08-c1ae-c745b9e3b7cb
7	69034	a1e505a9-5082-f1a2-cc5a-cd33e85e4504
8	69039	03d31c58-8c8e-3802-d7a1-23336bc13e3d
9	69044	b2881345-2db0-1365-5dc0-76a2b54b02b3
10	69049	aaa95ff7-dc47-c39d-4d20-2aec198bf5e0