

National University of Singapore  
School of Computing

**IT5003 - Data Structures and Algorithms**  
**Final Assessment**

(Semester 1 AY2024/25)

Time Allowed: 2 hours

---

INSTRUCTIONS TO CANDIDATES:

1. Do **NOT** open this assessment paper until you are told to do so.
2. This assessment paper contains **THREE** (3) sections.  
It comprises **SIXTEEN** (16) printed pages, including this page.
3. This is an **Open Book Assessment**.  
Only non-programmable calculator is allowed in this assessment.
4. Answer **ALL** questions within the **boxed space** of the answer sheet (page 13-16).  
For Section A and B, shade the option in page 13 of the answer sheet (use 2B pencil).  
There are a few starred (\*) boxes: free 1 mark if left blank but 0 for wrong answer (no partial).  
The answer sheet is at page 13-16 but you will still need to hand over the entire paper.  
You can use either pen or pencil. Just make sure that you write **legibly!**
5. Important tips: Pace yourself! Do **not** spend too much time on one (hard) question.  
Read all the questions first! Some (subtask) questions might be easier than they appear.
6. You can use **pseudo-code** in your answer but beware of penalty marks for **ambiguous answer**.  
You can use **standard, non-modified** classic algorithm in your answer by just mentioning its name, e.g. run BFS on graph  $G$ , Dijkstra's on graph  $G'$ , etc.
7. The total marks is 100. All the best :)

## A MCQs ( $20 \times 2 = 40$ marks)

Select the **best unique** answer for each question.

Each correct answer worth 2 marks.

1. What is the *tightest worst-case* time complexity of the following Python code in terms of  $n$ ?

Notice the keyword ‘tightest’, i.e., if the best answer is  $x$  but you choose an option that is worse than  $x$  (which is still true in Big O notation, but not the tightest), you will be marked as wrong. Similarly if you choose an option that is better than  $x$  (impossible), you will be marked as wrong.

```
def foo(nums): # Disclaimer: generated by Chat-GPT
    n = len(nums)
    for i in range(n):
        for j in range(i + 1, n):
            sorted_pair = sorted([nums[i], nums[j]])
            if sorted_pair[0] == sorted_pair[1]:
                print("outcome_A")
        return
    print("outcome_B")
```

*# Example usage:*

```
foo([1, 3, 2, 4, 5, 1])
```

- a).  $O(n)$
  - b).  $O(n \log n)$
  - c).  $O(n^2)$
  - d).  $O(n^2 \log n)$
  - e).  $O(n^3 \log n)$
2. Which call of function `foo(nums)` will print “outcome B”?
- a). `foo([1, 3, 2, 4, 5, 1])`
  - b). `foo([1, 2, 3, 4, 5, 6, 7, 6, 5, 4, 3, 2, 1])`
  - c). `foo([7, 7])`
  - d). `foo([3, 4, 9, 11, 2, 7, 8, 10, 11, 1])`
  - e). None of the above
3. What is the *tightest worst-case* time complexity of the following Python code in terms of  $n$ ?

```
from math import log2
a = [i for i in range(n)]
b = [j for j in range(n*int(log2(n)))][:7] # assume n is a power of 2
```

```
c = []  
for ai in a:  
    for bi in b:  
        c.append(ai*bi)
```

- a).  $O(\log n)$
- b).  $O(n)$
- c).  $O(n \log n)$
- d).  $O(n^2)$
- e).  $O(n^2 \log n)$

The rest of page 3 are redacted.

Everything in page 4 are redacted.

Everything in page 5 are redacted.

Everything in page 6 are redacted.

Everything in page 7 are redacted.

## B Simpler Questions (10 marks)

### B.1 Big-O Time Complexity Analysis (5 marks)

There is an unknown algorithm that process a Python list  $L$  (containing  $X$  integers). This algorithm has been correctly analyzed to have time complexity of  $O(N^2)$ . The total number of individual operations performed by this algorithm is exactly 256 operations on a particular test case  $C$ . There is no randomized component in this unknown algorithm. For the five statements below, which statement(s) is/are **always True for all cases** given the information above?

1.  $X = N$ .
2.  $N = 16$ .
3. If we add more integers into  $L$  and thus  $X$  increases, then the algorithm will perform more than 256 operations for sure.
4. If we change Python list  $L$  into set  $S$ , the algorithm will now have  $O(N)$  time complexity.
5. If we re-run this unknown algorithm on test case  $C$  one more time, it will again perform 256 operations.

### B.2 A Sorting Algorithm (5 marks)

You are given the following Python code:

```
le = lambda l: [x for x in l[1:] if x <= l[0]]
ri = lambda l: [x for x in l if x > l[0]]
q = lambda l: q(le(l)) + [l[0]] + q(ri(l)) if l else []
```

For the five statements below, which statement(s) is/are **always True for all cases** for that code?

1. Function `q` takes in any list  $L$  and sorts  $L$  into non-decreasing order.
2. Function `q` can sort list  $L$  of strings.
3. Function `q` implements Randomized Quick Sort algorithm.
4. Function `q` runs in expected  $O(n \log n)$  time given list  $L$  of length  $n$ .
5. Function `q` runs in  $O(n)$  time given sorted list  $L$  (in non-decreasing order) of length  $n$ .



## C Applications (50 marks)

### C.1 Buddy System (15 marks)

There are  $N$  students lining up in a long list  $L = [L_0, L_1, \dots, L_{N-1}]$ . You, their teacher, are examining their heights one by one from the leftmost student (index 0) to the rightmost student (index  $N - 1$ ). You want to assign a buddy to each of your student using the following rule: The buddy of student  $i \in [1..N - 1]$  is the **highest indexed** student  $j \in [0..i - 1]$  that is **strictly shorter** than student  $i$ . If there is no such student  $j$ , then student  $i$  has no buddy. Your job is to compute a checksum  $S$ : the sum of heights of the buddies of all  $N$  students.

For example, if  $N = 7$  and  $L = [1, 3, 4, 2, 7, 1, 0]$ , then the buddies of each of the students are:

- student 0 (height 1) has no buddy (virtual height 0 that does not contribute to the checksum),
- student 1 (height 3) has student 0 as buddy (height 1),
- student 2 (height 4) has student 1 as buddy (height 3),
- student 3 (height 2) has student 0 as buddy (height 1),
- student 4 (height 7) has student 3 as buddy (height 2),
- student 5 (height 1) has no buddy (virtual height 0 too),
- student 6 (height 0) has no buddy (virtual height 0 too).

The checksum  $S$  that you have to output is thus  $0 + 1 + 3 + 1 + 2 + 0 + 0 = 7$ .

#### C.1.1 Check Your Understanding (4 marks)

What are the checksums ( $S_1$  and  $S_2$ ) if the inputs are:

1.  $N = 7, L = [1, 2, 3, 4, 5, 6, 7]$ .
2.  $N = 7, L = [8, 3, 9, 5, 2, 6, 9]$ .

#### C.1.2 Subtask 1: Naïve Algorithm (5 marks)

Design a naïve  $O(N^2)$  algorithm that directly implements the given buddy assignment rule above. If you are confident that you can solve this in  $O(N)$ , you can skip this Subtask 1.

#### C.1.3 Solve The Full Problem (6\* marks)

Design an  $O(N)$  algorithm that correctly implements the given buddy assignment rule above. Notice the (\*) marking scheme.

## C.2 BST Merging (15 marks)

You are given two Binary Search Tree (BST):  $A$  and  $B$ , containing integers. The BSTs are not necessarily balanced and have height  $H_A$  and  $H_B$ , respectively. The BSTs are not necessarily of similar size and have size  $N_A$  and  $N_B$ , respectively. You are given a guarantee that all integer values in  $A$  are strictly smaller than  $B$ . Design an algorithm to merge the two BSTs into one BST (either a new BST  $C$  or reuse  $A$  or  $B$ ).

### C.2.1 Subtask 1: Naïve Algorithm (8 marks)

Design a naïve  $O(N_A + N_B)$  algorithm to do this BST merging.

If you are confident that you can solve this in  $O(\min(H_A, H_B))$ , you can skip this Subtask 1.

### C.2.2 Solve The Full Problem (7\* marks)

Design an  $O(\min(H_A, H_B))$  algorithm that correctly solves the BST merging problem above.

Notice the (\*) marking scheme.

## C.3 Binge Reading Competitive Programming Book (20 marks)

Prof Halim had written the Competitive Programming book (the latest available version is CP4), which may or may not be useful to solve this task.

Some sections in CP4 discuss algorithmic topic that require the reader to read at least one (or more) earlier section(s), e.g., before one can understand section 4.2.3. about Breadth First Search (BFS) – 1 page, one has to read section 2.2.5. about Linked Data Structures (Queue ADT) – 2 pages and section 2.4.1. about Graph Data Structure – 5 pages. Some sections are considered “ultimate topic” as they have no other section depending on them.

You bought CP4 the night before this final assessment. Each section has page count. You want to read (and master) *just two* “ultimate topic” sections to improve your chance of doing well in this paper. Note that reading a section that has pre-requisite section(s) requires you to also read that pre-requisite section(s) (and this can be recursive).

Your task is to output the minimum number of pages to read so that you achieve your objective of reading (and mastering) two “ultimate topic” sections of CP4.

For example: There are four sections  $A$  (10 pages),  $B$  (7 pages),  $C$  (5 pages), and  $D$  (2 pages). Section  $C$  and  $D$  both depend on section  $A$ . Section  $B$  does not depend on any other section. So there are three “ultimate topic” sections:  $B$ ,  $C$ , and  $D$ . It is best to concentrate on reading (and mastering) sections  $C$  and  $D$  (we also need to read section  $A$ ), totalling  $10 + 5 + 2 = 17$  pages read.

### C.3.1 Store the Input Information ( $5 \times 1 = 5$ marks)

The input is given in  $m + 2$  lines, as follows:

- The first line containing two integers  $n$  and  $m$  where  $n$  ( $2 \leq n \leq 450$ ) indicates the number of sections (conveniently indexed from 1 to  $n$  instead of 2.2.5., 2.4.1., or 4.2.2. as mentioned above) and  $m$  ( $0 \leq m < \min(1000, n \times (n - 1)/2)$ ) indicates the number of section dependencies.

- The second line contains  $n$  positive integers (each positive integer is not more than 777), indicating the number of pages in each section.
- Finally, there are  $m$  lines each containing two integers  $u$  and  $v$  ( $1 \leq u < v \leq n$ ) indicating that section  $u$  must be read before section  $v$ .

We guarantee that there are at least two sections that classify as “ultimate topic” sections. Answer the five short fill-in-the-blanks questions in the answer sheet.

1. The graph described in the input is a special graph called a \_\_\_\_\_.
2. A section is an “ultimate topic” section if and only if \_\_\_\_\_.
3. Hint: It is important to reverse all edge directions of graph  $G$  given in the input. A graph  $G'$  with the same vertices as  $G$  but all its directed edges reversed is called \_\_\_\_\_ graph of  $G$ .
4. For the purpose of this problem, it is probably best to store the graph  $G'$  above using \_\_\_\_\_ graph data structure.
5. We store the information about the number of pages of each section in a/an \_\_\_\_\_ data structure.

### C.3.2 Check Your Understanding (2 marks)

There are five sections  $A$  (10 pages),  $B$  (7 pages),  $C$  (4 pages),  $D$  (6 pages), and  $E$  (15 pages). Section  $D$  depends on section  $A$ . Section  $C$  depends on section  $B$ . Section  $E$  does not depend on any other section. Determine which sections are the “ultimate topic” sections (1 mark) and what is the minimum number of pages to read to master two of the “ultimate topic” sections (1 mark)?

### C.3.3 Subtask 1: Just one “ultimate topic” (4 marks)

If you decide to master just one “ultimate topic”: section  $t$ , how many pages that you have to read in total? Design a correct  $O(n + m)$  algorithm that runs on  $G'$  for this Subtask 1.

### C.3.4 Subtask 2: Two specific “ultimate topics” (4 marks)

If you decide to master two specific “ultimate topic”: section  $t_1$  and  $t_2$  (not necessarily the best two “ultimate topic” sections), how many pages that you have to read in total? To solve this subtask, you (probably) need to correctly solve Subtask 1 first. Now analyze the time complexity of your algorithm.

### C.3.5 Solve the Problem (5\* marks)

Now we don't tell you which two specific “ultimate topic” sections to concentrate on.

Design the full solution and analyze its time complexity.

Notice the (\*) marking scheme.

This page is empty.

# The Answer Sheet

Write your Student Number in the box below using **(2B) pencil**.

**Do NOT write your name.**

STUDENT NUMBER											
A											
U	<input type="radio"/>	0	0	0	0	0	0	0	0	A	N
A	<input checked="" type="radio"/>	1	1	1	1	1	1	1	1	B	R
HT	<input type="radio"/>	2	2	2	2	2	2	2	2	E	U
NT	<input type="radio"/>	3	3	3	3	3	3	3	3	H	W
		4	4	4	4	4	4	4	4	J	X
		5	5	5	5	5	5	5	5	L	Y
		6	6	6	6	6	6	6	6	H	
		7	7	7	7	7	7	7	7		
		8	8	8	8	8	8	8	8		
		9	9	9	9	9	9	9	9		

Write your MCQ answers in the special MCQ answer box below for automatic grading.

We do not manually check your answer.

Shade your answer properly (use (2B) pencil, fully enclose the circle; select just one circle).

No	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
B	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
C	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
D	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
E	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

No	16	17	18	19	20
A	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
B	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
C	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
D	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
E	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

No	B.1.1	B.1.2	B.1.3	B.1.4	B.1.5	No	B.2.1	B.2.2	B.2.3	B.2.4	B.2.5
T	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	T	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
F	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	F	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

---

Box C.1.1 Check Your Understanding (write two integers  $S_1$  and  $S_2$ )

Box C.1.2 Design a naïve  $O(N^2)$  algorithm

Box C.1.3\* (1 if blank, 0 if wrong) Design an  $O(N)$  algorithm

Box C.2.1 Design a naïve  $O(N_A + N_B)$  algorithm

Box C.2.2\* (1 if blank, 0 if wrong) Design an  $O(\min(H_A, H_B))$  algorithm

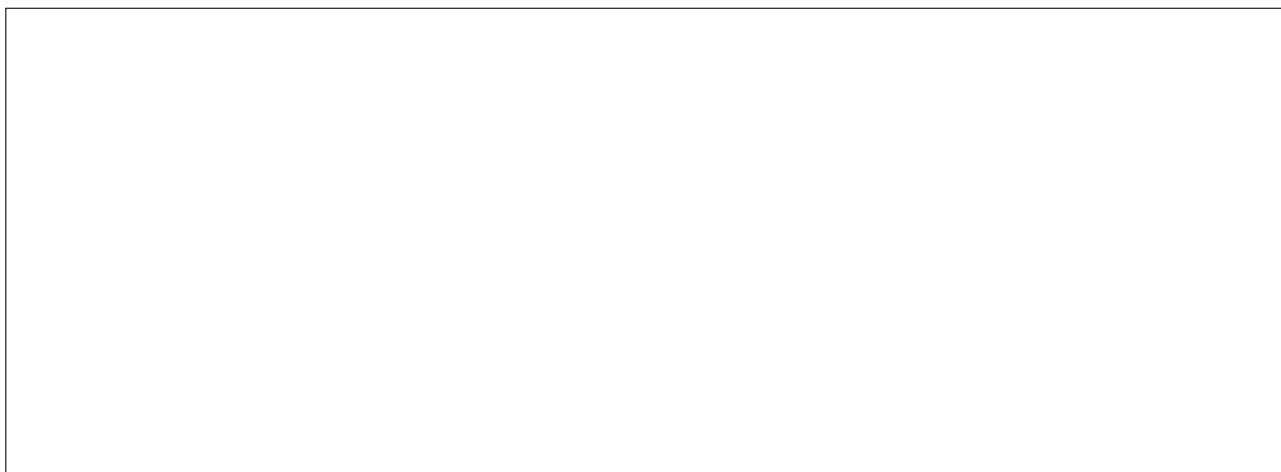
Box C.3.1. Fill-in-the-blanks

1. The graph described in the input is a special graph called a .....
2. A section is an “ultimate topic” section if and only if .....
3. Hint: It is important to reverse all edge directions of graph  $G$  given in the input. A graph  $G'$  with the same vertices as  $G$  but all its directed edges reversed is called ..... graph of  $G$ .
4. For the purpose of this problem, it is probably best to store the graph  $G'$  above using .....  
..... graph data structure.
5. We store the information about the number of pages of each section in a/an .....  
data structure.

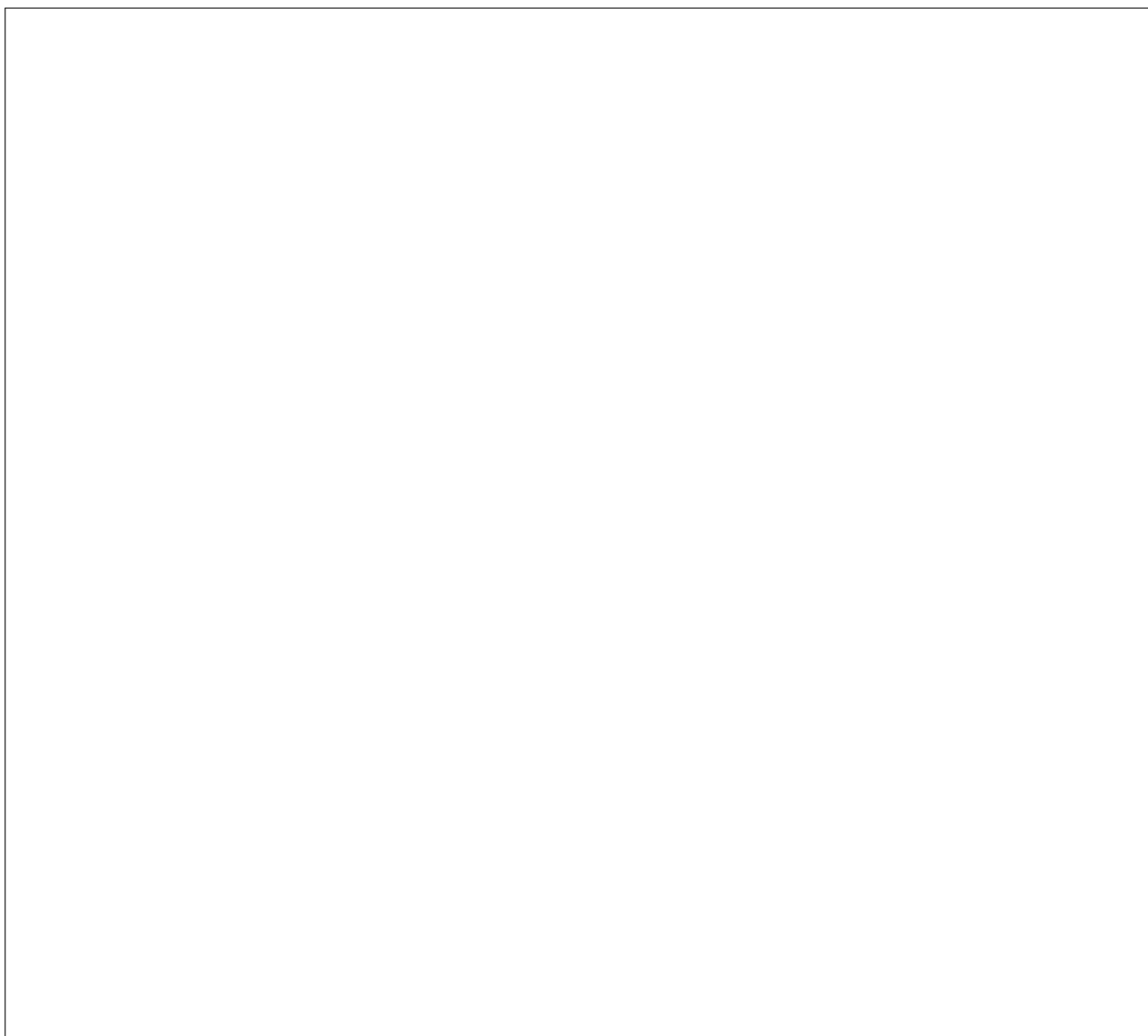
Box C.3.2 Check Your Understanding (write “ultimate topic” sections and the final answer)

Box C.3.3 Just one “ultimate topic”

Box C.3.4 Two specific “ultimate topics”



Box C.3.5\* (1 if blank, 0 if wrong) Solve the Problem



– END OF PAPER; All the Best –