

National University of Singapore

School of Computing

IT5003 - Data Structures and Algorithms

Final Assessment

(Semester 2 AY2023/24)

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, 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 ($25 \times 2 = 50$ marks)

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

Each correct answer worth 2 marks.

- Analyze the following Python code implementation of Binary Search algorithm:

```
def binary_search(lst, v): # Disclaimer: generated by Chat-GPT 3.5
    if len(lst) == 0:
        return False
    else:
        mid = len(lst) // 2
        if lst[mid] == v:
            return True
        else:
            if v < lst[mid]:
                return binary_search(lst[:mid], v)
            else:
                return binary_search(lst[mid+1:], v)

# assume you have a sorted list L of n Integers
print(binary_search(L, 7))
```

What is the worst-case time complexity of that implementation in terms of n ?

- a). $O(\log n)$
 - b). $O(n)$
 - c). $O(n \log n)$
 - d). $O(n^2)$
 - e). None of the above; as that binary search implementation is buggy
- What is the *worst-case* time complexity of the following Python code in terms of n ?

```
def magic(n):
    counter = 0
    i = 7
    while i < n:
        for j in range(0, n, 2):
            for k in range(2):
                counter += 1
        i = i * 7
    return counter
```

a). $O(\log n)$

b). $O(n)$

c). $O(n \log n)$

d). $O(n^2)$

e). $O(n^2 \log n)$

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

```
a = [i for i in range(n)]
b = [j for j in range(n**2)]
c = sorted(a+b[:n])
```

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.

Everything in page 8 are redacted.

23. The current version (as of final assessment day of Sem2 AY23/24) of VisuAlgo /heap page allows duplicate Integer keys to be inserted. If value v is already inserted before and another copy of v is going to be inserted again, the visualization of /heap simply inserts the duplicate like with any other new key. Which statement is **correct**?
- The time complexity of key basic Binary (Max) Heap operations (enqueue and dequeue) remain the same: $O(\log n)$
 - Insertion of n copies of the same values into an initially empty Binary (Max) Heap runs in $O(n^2)$
 - Creating a max heap of n copies of the same value using the faster Create-Heap routine (check Max Heap property from the last internal vertex back to root) now runs in $O(n \log n)$ instead of $O(n)$
 - Heap Sort now runs in $O(n^2)$
 - Having duplicates can cause a bug involving the ordering of equal-value Integers
24. The current version (as of final assessment day of Sem2 AY23/24) of VisuAlgo /hashtable page allows duplicate Integer keys to be inserted. If value v is already inserted before and another copy of v is going to be inserted again, the visualization of /hashtable simply inserts the duplicate like with any other new key. In class, we have discussed that this is not the best way to handle duplicates and will soon be addressed in the future. Why?
- In Open Addressing: Linear Probing, this causes a severe primary clustering problem
 - In Open Addressing: Quadratic Probing, this causes a severe secondary clustering problem
 - In Open Addressing: Double Hashing, this causes a severe (unnamed) clustering problem
 - In Separate Chaining, this causes duplicates to form long chains
 - All of the above
25. One of the applications below is **not** what we discussed in class as potential application of Data Structures and Algorithms in real-life. Select the impostor!
- We should sort the SGD notes in our wallet in either increasing (2, 2, ..., 2, 5, 5, ..., 5, 10, 10, ..., 50, ...) or decreasing order to speed up our daily cash payments
 - When there is a queue in front of a food stall, we should disregard the First-In-First-Out (FIFO) order and slot ourself closer to the front
 - When we see a ‘priority seat’ in MRT is currently unoccupied, we can choose to sit on it, but give the seat to other who belong to the ‘priority group’ when appropriate
 - We should organize the items in our bag using a deterministic ‘hash function’ so that we can quickly retrieve any item in $O(1)$ time (i.e., no ‘search’ needed)
 - When exploring a totally new city, it is generally a good idea to follow the path shown by our favorite map software to reach our new destination faster

B Simpler Questions (30 marks)

B.1 Feedback on IT5001 ($2 \times 2 = 4$ marks)

After taking IT5003, what will you say to prospective students (or your past-self) in IT5001 that you wish you could know earlier in IT5001? Give two logical sentences (2 marks each).

B.2 Real-Life Application of DSA ($3 \times 2 = 6$ marks)

After taking IT5003, what Data Structures and Algorithms (DSA) principles/ideas that you will apply in your own real-life situations? Give three logical sentences (2 marks each). See MCQ 25 for at least 4 inspirations (but neither can be used as your own answer verbatim).

B.3 Triple-Ended Queue (10 marks)

In class, we learn about deque (double-ended queue) ADT and its possible data structure implementations. deque ADT allows for efficient pushing and popping of elements from either the front or the back of the queue. Now, let's bring this ADT up to the next level, the teque (triple-ended queue)! The teque ADT supports the following operations:

- `append(x)`: insert the element x into the back of the teque.
- `appendleft(x)`: insert the element x into the front of the teque.
- NEW: `insertmiddle(x)`: insert the element x into the middle of the teque. The inserted element x now becomes the new middle element of the teque. If n is the size of the teque before the insertion, the insertion index for x is $\lfloor (n + 1)/2 \rfloor$ (using 0-based indexing).
- `printteque()`: list out the content of teque from the front to the back.

Propose how to implement this teque ADT so that the first three operations especially `insertmiddle(x)` all run in $O(1)$ (6 marks); only the `printteque()` operation can be in $O(n)$ where n is the size of the teque (4 marks).

B.4 Stability of Binary (Max) Heap With Duplicates (10 marks)

In class, we learn about Binary (Max) Heap data structure. The version that we learn from recent semester is the version that allows duplicates. To be specific, notice the bold part of the generalized Binary Max Heap property: The parent of each vertex - except the root - contains value greater than (**or equal to - we now allow duplicates**) the value of that vertex.

In class, we also learn about stability of sorting algorithm: A sorting algorithm is called stable if the relative order of elements with the same key value is preserved by the algorithm after sorting is performed.

B.4.1 Show Your Understanding! (3 marks)

Insert three Integers $\{7_1, 7_2, 7_3\}$ one after another (the subscripts 1, 2, 3 are only shown to help us differentiate the otherwise three equal Integers 7, what we actually insert are $\{7, 7, 7\}$) into an initially empty Binary (Max) Heap. Then do three ExtractMax() operations. Do you get $\{7_1$ (the first 7 that we insert), $7_2, 7_3$ (the last 7 that we insert) $\}$ in that order (stable), or do you get any other non-stable order? Give a short explanation!

B.4.2 Make Heap Sort a Stable Sort! (7* marks)

Show how to make Heap Sort algorithm ($O(N \log N)$) or $O(N)$ Create Binary (Max) Heap (of Integers) and then N calls of ExtractMax()) is declared a stable sorting algorithm even with the presence of duplicate Integers! Note that answers along the line of: ‘use Python’s Tim Sort, Merge Sort, (or any other known stable-sort algorithm) instead of Heap Sort’ is not acceptable. Your answer must utilize Binary (Max) Heap Complete Binary Tree and/or Max Heap properties. Your answer must generally maintain stability on any future enqueue / dequeue operations involving duplicates. Hint: We can convert Python `heapq` (Min Heap) of numbers into a Max Heap by transforming the input (inserting the negation of the numbers instead of the original numbers). Use a similar idea for this question.

C Applications (20 marks)

C.1 Graph Traversal (20 marks)

You are given a connected graph $G = (V, E)$ with N ($1 \leq N \leq 10^5$) vertices (labeled from $[1..N]$) and $N - 1$ undirected weighted edges (the weight is a positive Integer not more than 10^3). Traversing an edge gives you the profit that is equals to the weight of the edge. However, traversing the same edge more than once does not give you additional profit. For this problem, we have an important additional constraint: each edge can only be traversed up to k ($1 \leq k \leq 10^9$) times.

Your task is to start from any source vertex in G and end at any other vertex in G , traverse the graph in such a way to maximize your total profit.

This problem is solvable with (graph) algorithm(s) that we discussed in class.

C.1.1 Check Your Understanding (2 marks)

1. The graph mentioned in this problem is special and it has a name. What is it?
2. Is this a *Single-Source Shortest Paths* (SSSP) problem on weighted graph?

C.1.2 Subtask 1: $k \geq 2$ (3 marks)

Read the problem carefully and design a correct algorithm to fully solve this problem when $k \geq 2$. Analyze the worst-case time complexity of your proposed algorithm.

C.1.3 Subtask 2: $k = 1$ and you must start from vertex 1 (5 marks)

Read the problem carefully and design a correct algorithm to fully solve this problem when $k = 1$ and you must start from vertex 1. Remember that the edges can have varying weight of positive Integers not more than 10^3 . Analyze the worst-case time complexity of your proposed algorithm.

C.1.4 Sample Test Cases (3 × 1 = 3 marks)

Given this graph with $N = 6$ and $N - 1 = 5$ undirected weighted edges in format (u and v , weight w): $\{(u : 1, v : 2, w : 3), (u : 2, v : 3, w : 1), (u : 1, v : 4, w : 7), (u : 1, v : 5, w : 9), (u : 5, v : 6, w : 5)\}$.

What is the solution if:

- a). $k = 7$ (see C.1.2),
- b). if $k = 1$ and you must start from vertex 1 (see C.1.3), and
- c). $k = 1$ and you can start from any vertex (see C.1.5)?

C.1.5 Solve The Full Problem (7* marks)

Use all the insights above and design a correct algorithm to fully solve this problem ($1 \leq N \leq 10^5$; $1 \leq k \leq 10^9$, weighted edges, and you can start/end from any vertex in G).

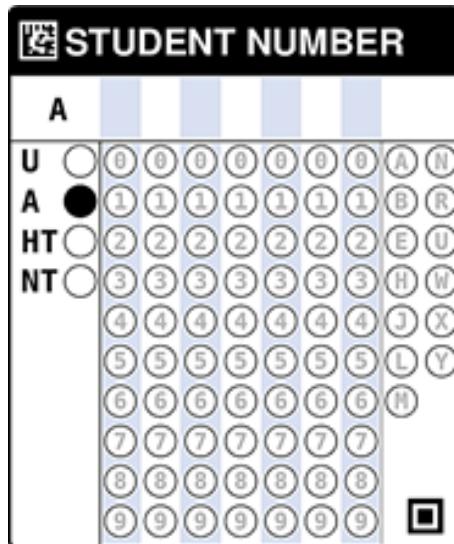
Analyze the worst-case time complexity of your proposed algorithm.

To get full 7 marks, your algorithm must be the fastest possible.

The Answer Sheet

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

Do NOT write your name.



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"/>														
B	<input type="radio"/>														
C	<input type="radio"/>														
D	<input type="radio"/>														
E	<input type="radio"/>														

No	16	17	18	19	20	21	22	23	24	25
A	<input type="radio"/>									
B	<input type="radio"/>									
C	<input type="radio"/>									
D	<input type="radio"/>									
E	<input type="radio"/>									

Box B.1. Two sentences about IT5001

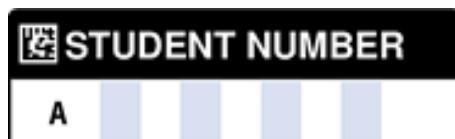
Box B.2. Three sentences about real-life application of DSA

Box B.3. Triple-Ended Queue Implementation:

Box B.4.1 Show Your Understanding!

Box B.4.2.* (1 if blank, 0 if wrong) Make Heap Sort a Stable Sort! (you can use top box of page 15)

In case this sheet is detached from page 13-14, re-write your Student **Number** again:



Box C.1.1. Check Your Understanding

1. The special graph mentioned in this problem is a _____.
2. Is this a *Single-Source* Shortest Paths (SSSP) problem on weighted graph? Select one: Yes/No.

Box C.1.2. Subtask 1: $k \geq 2$

Box C.1.3. Subtask 2: $k = 1$ and you must start from vertex 1

Box C.1.4. Sample Test Cases

Box C.1.5.* (1 if blank, 0 if wrong, 2 for the second best solution) Solve The Full Problem

– END OF PAPER; All the Best –