National University of Singapore

School of Computing

CS2040S - Data Structures and Algorithms

Final Assessment

(Semester 1 AY2025/26)

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 TWO (2) sections.
   It comprises SIXTEEN (16) printed pages, including this page.
3. This is an **Open Book Assessment**.
   You cannot use any electronic device except one non-programmable calculator.
4. 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 (sub-)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 Inorder Traversal on BST T, BFS on graph G, Dijkstra's on graph G', etc.
7. The total marks is 60. All the best :)

A MCQs ($6 \times 1 = 6$ marks)

Select the **best unique** answer for each question. Each correct answer is worth 1 mark.

1. What is the *tightest worst-case* time complexity of `answerQueries` in terms of n and m? 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.

```java
private int upperBound(int[] arr, int target) {
    int lo = 0, hi = arr.length;
    while (lo < hi) {
        int mid = (lo+hi) / 2;
        if (arr[mid] <= target)
            lo = mid+1;
        else
            hi = mid;
    }
    return lo;
}

public int[] answerQueries(int[] nums, int[] queries) {
    int n = nums.length; // there are n integers in nums
    Arrays.sort(nums); // please analyze this as O(n log n)
    int[] prefix = new int[n];
    prefix[0] = nums[0];
    for (int i = 1; i < n; ++i)
        prefix[i] = prefix[i-1] + nums[i];
    int m = queries.length; // there are m integer queries
    int[] result = new int[m];
    for (int i = 0; i < m; ++i) {
        int idx = upperBound(prefix, queries[i]); // see above
        result[i] = idx;
    }
    return result;
}
```

a). $O(m \log n)$

b). $O((n+m) \log n)$

c). $O(n+m)$

d). $O(nm)$

e). $O(\infty)$, i.e., an infinite loop

2. Which of the following statements about Singly Linked List (SLL) with both head and tail pointers is correct?

   a). Accessing the middle element of an SLL takes $O(1)$ time
   b). An SLL uses more memory per vertex than a Doubly Linked List (DLL)
   c). You can insert a new vertex at index $i + 1$ in $O(1)$ time if we have the pointer to vertex i
   d). In an SLL, you can easily traverse the list in both directions
   e). Deleting the last vertex in an SLL always takes $O(1)$ time

3. Which of the following statements about Binary Max-Heap (1-based compact array) is correct?

   a). Finding the element with the minimum value takes $O(1)$ time
   b). The parent of a vertex at array index i is always located at index $2 \cdot i$.
   c). To add a new element v, we first place v at the root, and then a `shift-down` operation is performed to restore the heap property
   d). Every vertex's value is greater than or equal to the value of its parent vertex
   e). Deleting the maximum element involves replacing the root with the last element in the max-heap and then performing a `shift-down` operation

4. Which of the following statements is correct about hash tables that use Separate Chaining (SC) to resolve collisions?

   a). The worst-case time complexity for a successful search is $O(1)$ regardless of the number of keys inserted
   b). Deletion is more difficult in SC than in Open Addressing (OA) method because it requires rehashing all keys
   c). A perfect hash function can always be created to guarantee that no collision will occur
   d). If all n keys hash to the same hash value, the hash table effectively becomes a linked list with $O(n)$ operations
   e). SC requires the hash table array to be resized more often than OA does

5. Which of the following statements about Binary Search Trees (BSTs) is always true for *any* valid BST (with no duplicate keys) with $n \geq 1$ unique integer keys?

   a). The inorder traversal of the BST produces the keys in decreasing order
   b). The resulting BST will have height $\lfloor \log_2 n \rfloor$ if the keys are inserted in ascending order
   c). If two BSTs have the same inorder traversal sequence, they must be structurally identical
   d). If a BST contains only odd integers, then the number of internal vertices is always odd
   e). Deleting a vertex with two children always decreases the total number of vertices in the BST by exactly one and never affects the inorder traversal order of the rest

6. The performance and/or the correctness of the famous Dijkstra's algorithm depends on the implementation details. Here is one correct and fast, $O((n+m)\log n)$ Dijkstra's algorithm.

```
// (Modified) Dijkstra's algorithm
PriorityQueue<IntegerPair> pq = new PriorityQueue<>();
pq.offer(new IntegerPair(0, s));
while (!pq.isEmpty()) {
  IntegerPair top = pq.poll();
  int d = top.first(), u = top.second();
  if (d > dist.get(u)) continue; // [THIS LINE]
  for (IntegerPair v_w : AL.get(u)) {
    int v = v_w.first(), w = v_w.second();
    if (dist.get(u)+w >= dist.get(v)) continue;
    dist.set(v, dist.get(u)+w);
    pq.offer(new IntegerPair(dist.get(v), v));
  }
}
```

What happen if the line labeled as "// **[THIS LINE]**" is **REMOVED**?

a). The implementation remains correct but runs slower than $O((n+m)\log n)$

b). The implementation remains correct and still runs in $O((n+m)\log n)$

c). The implementation remains correct but runs faster than $O((n+m)\log n)$

d). The implementation becomes wrong but still runs in $O((n+m)\log n)$

e). The implementation becomes wrong and runs slower than $O((n+m)\log n)$

B Application Questions ($9 + 4 \times 10 + 5 = 54$ marks)

B.1 Earliest Time to (at least) Pass CS2040S (9 marks)

The default passing mark to pass a course (with a D) in NUS, without any moderation, is 40 out of 100 marks. Suppose that you have scored 39 CA marks prior to taking this 60 marks final assessment paper[1].

There are n ($1 \leq n \leq 20$) questions in this paper. Since the minimum mark of any question is 1, scoring *any* single question will guarantee that you pass this course.

Each question i has two properties `understood[i]` – how long does it take to read and understand question i and `solve[i]` – how long does it take to fully solve and write down the answer for question i (after you had read and understand it). Both properties are positive integers with value at most 120 minute(s).

What is the earliest time (in terms of minute(s) since this paper starts) that you know that you have (at least) pass CS2040S if you luckily choose the best question to start? See a few examples in Table 1 below:

[1] No student is at perfect 40 out of 40 CA marks, i.e., at the start, nobody has passed, yet.

No	Input n	*understood*	*solve*	Output	Explanation
1	2	$[4, 5]$	$[5, 2]$	7	$4 + 5 = 9$ vs $5 + 2 = 7$
2	3	$[8, 8, 8]$	$[60, 20, 30]$	28	solve the second question: $8 + 20 = 28$
3	4	$[15, 20, 18, 16]$	$[20, 20, 20, 20]$	35	solve the first question: $15 + 20 = 35$

Table 1: A Few Example Scenarios

Use any data structure and/or algorithm that you need to solve this task.
As n is very small, there are many possible ways to solve this problem.

B.2 Fourth Power (10 marks)

You are given an `int[]` array A that is *already sorted in non-decreasing order*. There are n elements in A, and each integer lies in the range $[-30\,000..30\,000]$. Your task is to return an array of the *fourth power* of each integer (use Java `long` datatype), but the resulting array must now be in *non-increasing order*. See a few examples in Table 2 below:

No	Input `int[]` array A non-**decreasing**	Output `long[]` array non-**increasing**	Explanation
1	`[-7, -7, 2, 9]`	`[6561, 2401, 2401, 16]`	$9^4 = 6561$, $(-7)^4 = 2401$, $2^4 = 16$
2	`[1, 2, 2, 4, 5]`	`[625, 256, 16, 16, 1]`	$5^4 = 625$, $4^4 = 256$
3	`[-3, -2, -1]`	`[81, 16, 1]`	$(-3)^4 = 81$

Table 2: A Few Examples for the Fourth Power Problem

Design any correct algorithm that solves this task for 6 marks.
For full 10 marks, the correct algorithm must run in $O(n)$ and uses only a few $O(1)$ additional variables.
The output array of size $O(n)$ has been created for you and is not part of the additional variables.

B.3 Robot Movement (10 marks)

You are a robot currently located at the origin: cell $(7, 7)$ of an infinite 2D grid. You execute a simple robot movement command (a String C of n characters). Each command in C is either a 'U' (Up), 'R' (Right), 'D' (Down), or 'L' (Left) by one integer unit. You wonder if you ever revisit any cell that you have been to before. Output `true` if you do, or `false` otherwise. Check a few examples below, where the cell that you revisit is underlined:

No	Input String C	Output	Explanation
1	`"URDL"`	`true`	$(7, 7) \to (7, 8) \to (8, 8) \to (8, 7) \to \underline{(7, 7)}$
2	`"URDLL"`	`true`	$(7, 7) \to (7, 8) \to (8, 8) \to (8, 7) \to \underline{(7, 7)} \to (6, 7)$
3	`"UURR"`	`false`	$(7, 7) \to (7, 8) \to (7, 9) \to (8, 9) \to \underline{(9, 9)}$
4	`"DUDUDUDU"`	`true`	You will revisit $(7, 7)$ and $(7, 6)$ multiple times

Table 3: A Few Examples for The Robot Movement Problem

Design any correct algorithm that solves this task in $O(n^2)$ for 6 marks.
For 8 marks, that correct algorithm must run in $O(n \log n)$.

For full 10 marks, that correct algorithm must run in $O(n)$ and use no more than $O(n)$ space.

B.4 Special Binary Tree (10 marks)

You are given the root of a Binary Search Tree (BST) T with n vertices (each vertex is an integer $\in [1..10^9]$). From each vertex, you can access its left and right children, if any. (Note that the value of each vertex is not important for this problem.) Output an integer denoting the size of the *largest perfect binary subtree*.

Let us now formalize the definition: A perfect binary tree is a tree in which all leaves are on the same level and every internal vertex has exactly two children. See Figure 1 for illustration. The roots of all perfect binary subtrees *that are not just leaves* are highlighted. Note that each leaf is also a perfect binary subtree of size 1.

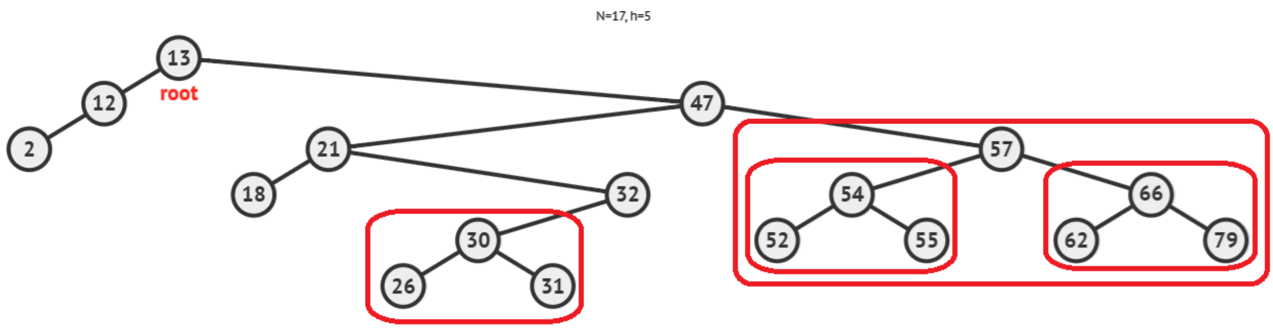


Figure 1: An example BST T (note that each integer can be as large as 10^9)

No	Input BST T	Output	Explanation
1	As in Figure 1	7	Subtree rooted at 57 is a perfect binary tree of size 7
2	As in Figure 1 but vertex 79 is deleted	3	Subtree rooted at 57 is no longer a perfect binary tree. largest perfect binary subtrees are now rooted at 30 and 54, both of size 3
3	A skewed right/left BST	1	Only the leaf vertices are perfect binary subtrees of size 1

Table 4: A few examples for the Special Binary Tree problem

Design any correct algorithm that solves this task in $O(n^2)$ for 6 marks.
For full 10 marks, the correct algorithm must run in $O(n \log n)$ or faster.
Especially for this question, please answer in pseudo-code.

B.5 Superstar (10 marks)

You are given an *undirected simple graph* G consisting of n vertices (labeled $[0..n-1]$) in form of m edges stored in an Edge List ($0 \leq m \leq n \cdot (n-1)/2$). Each edge connects two *distinct* vertices bidirectionally. To spice things up, you are also given an integer array *value* of length n, where $value[u]$ denotes the value of vertex u (this value ranges $\in [-10^5..10^5]$). This array represents the *vertex weights*.

In this problem, we focus on a special kind of graph called a *star graph*. A star graph has a central vertex connected to zero or more neighbors.

The input graph G may contain many subgraphs that form star graphs. (For example, each individual vertex on its own can be considered a trivial star graph of size 1.) We define the *star-value* of a star (sub)graph of G as the *sum* of the values of the vertices in that star (sub)graph.

Your task is as follows: Given another integer k, return the largest star-value among all star (sub)graphs of G that contain *at most* k edges. See Figure 2 and Table 5 below for several examples.

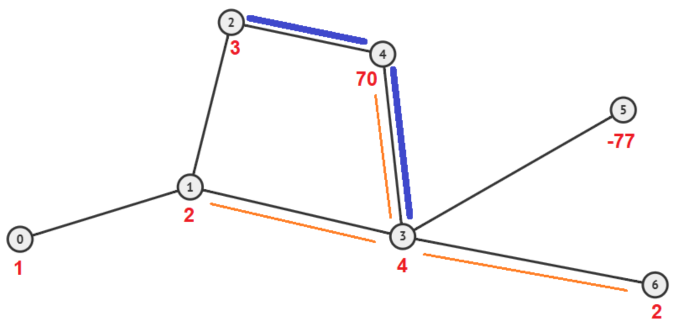


Figure 2: An example graph G with two star (sub)graphs highlighted

No	Input Graph G	k	Output	Explanation
1	As in Figure 2	2	77	Subgraph centered at 4 with neighbors 2 and 3: star-value: $70 + 3 + 4 = 77$
2	As in Figure 2	6	78	Subgraph centered at 3 with neighbors 1, 4, and 6: star-value: $4 + 2 + 70 + 2 = 78$ notice that we do not use all $k = 6$ edges
3	As in Figure 2	1	74	Subgraph centered at 3 with neighbor 4: star-value: $4 + 70 = 74$ or alternatively, subgraph centered at 4 with neighbor 3: star-value: $70 + 4 = 74$
4	As in Figure 2	0	70	Vertex 4 is a single-vertex star with star-value: 70

Table 5: A few examples for the Superstar problem

For full 10 marks, solve the complete problem correctly and analyze its time complexity.
You will be given marks between [7..10] depending on the time complexity of the correct solution.
However, if you are not sure how to solve the full version, consider just solving this special case that admits an easier $O(n^2)$ (or faster) solution for 6 marks (and indicate this clearly in your answer):

> In the **simplified** version of the problem, $k = n - 1$ at all time — see Test Case No. 2 in Table 5 as an illustration.

B.6 Completing Errands (5* marks) - This is a STARRED task

You are given a 2D integer matrix of size $m \times n$ ($m \geq 1$ and $n \geq 1$) where a cell value of 0 means the cell is blocked (you cannot pass through it), a value of 1 means an empty cell that can be freely walked through, and any other integer greater than 1 (up to 10^9) represents a special cell that must be visited in a specific order (but can be temporarily passed through, see Example 3). The values in the matrix greater than 1 are *distinct*. There are k such special cells with values greater than 1, where $1 \leq k \leq \sqrt{m \times n}$. In another words, many cells in the matrix contains 0s and 1s.

Your movement is constrained as follows: in one unit step, you may move to a neighboring cell to the North, East, South, or West, provided that the cell is within the matrix boundaries and is not blocked (i.e., not zero). You can revisit a cell several times (highlighted in Example 3).

You start at the top-left corner cell (index $(0, 0)$). Your goal is to visit all cells with values greater than 1 in strictly ascending order, using the minimum total number of steps. If it is impossible to visit all such cells in order (due to blocked cells), output ∞.

See Figure 3 for example grids and the table below for example inputs and outputs.

1	0	0	0	0	0
2	0	5	1	6	0
3	1	4	1	7	0
1	1	1	1	1	0
1	1	1	1	1	0
0	0	0	0	0	0

Grid 1, 6x6

1	1	1
7	0	77
1	0	1
1	1	1

Grid 2, 4x3

1	1	1
7	0	77
99	0	1
1	1	1

Grid 3, 4x3

1	0	1
7	0	77
1	0	1
1	0	1

Grid 4, 4x3

Figure 3: Example 2D matrices

No	Input Grid	Output	Explanation
1	Grid 1 of Figure 3	8	Path visits cells in order: $1 \to 2 \to 3 \to (1) \to 4 \to 5 \to (1) \to 6 \to 7$
2	Grid 2 of Figure 3	5	Moves down first to 7, then moves back up to visit 1, them move right twice to visit the other two $1 \to 1$, then go down to visit 77
3	Grid 3 of Figure 3	10	Similar to example 2, but additionally requires 5 more steps to reach 99 after 77, i.e., from 77, go up, left, left, down (revisit 7), down to reach 99 (this example shows that we can revisit the cells)
4	Grid 4 of Figure 3	∞	Moves down first to 7 but then cannot reach 77 (blocked)

Table 6: Examples for the Completing Errands Problem

Design any correct algorithm that solves this task for 5* marks.

~~You are not told the target optimal time complexity (in terms of m, n, and/or k).~~

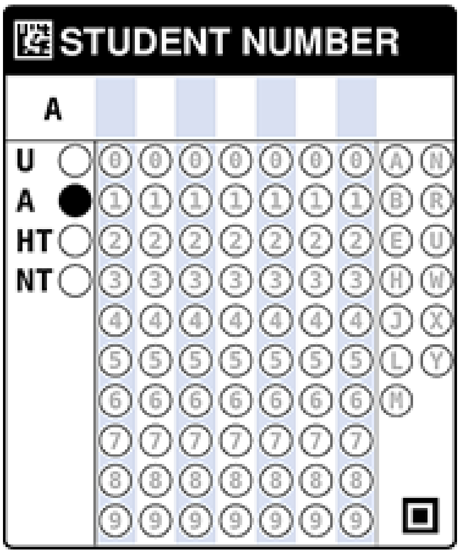
You need to design an $O(m^{1.5}n^{1.5})$ algorithm.

If you leave this task blank, you will receive free 0.5 marks.

However, if you submit an incorrect or inefficient solution, you will not receive any mark.

The Answer Sheet for Semester 1 AY2025/26

Write your Student Number in the box below using **(2B) pencil**.
Do NOT write your name.



This portion is for examiner's use only

Section	Maximum Marks	Your Marks	Grading Remarks
A	6		
B	54		
Total	60		

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
A	◯	◯	◯	◯	◯	◯
B	◯	◯	◯	◯	◯	◯
C	◯	◯	◯	◯	◯	◯
D	◯	◯	◯	◯	◯	◯
E	◯	◯	◯	◯	◯	◯

Box B.1. Earliest Time to (at least) Pass CS2040S

I claim that my solution runs in $O($____________________$)$.

Box B.2. Fourth Power

I claim that my solution runs in $O(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)$.

Box B.3. Robot Movement

I claim that my solution runs in $O(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)$.

Box B.4. Special Binary Tree

I claim that my solution runs in $O(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)$.

Box B.5. Superstar (indicate clearly whether you are attempting the simplified or the full version)

I claim that my solution runs in $O($____________________$)$.

Box B.6.* Completing Errands (This is a STARRED task, only answer it if you are sure)

I claim that my solution runs in $O(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)$.

You can use this page as an extra answer sheet
but clearly add this in the answer box: '**PLEASE READ THE LAST PAGE**'

– END OF PAPER; All the Best –