CS2040S — Data Structures and Algorithms
School of Computing
National University of Singapore

Final Assessment

26 Apr 2023 **Time allowed:** 2 hours

Instructions — please read carefully:

1. Do not open the final until you are directed to do so.
2. Read **all** the instructions first.
3. The quiz is closed book. You may bring one double-sided sheet of A4 paper to the quiz. (You may not bring any magnification equipment!) You may NOT use a calculator, your mobile phone, or any other electronic device.
4. The **QUESTION SET** comprises **SIX (6) questions** and **EIGHTEEN (18) pages**, and the **ANSWER SHEET** comprises of **TWELVE (12) pages**.
5. The time allowed for solving this test is **2 hours**.
6. The maximum score of this test is **100 marks**. The weight of each question is given in square brackets beside the question number.
7. All questions must be answered correctly for the maximum score to be attained.
8. All questions must be answered in the space provided in the **ANSWER SHEET**; no extra sheets will be accepted as answers.
9. You must submit only the **ANSWER SHEET** and no other documents. The question set may be used as scratch paper.
10. An excerpt of the question may be provided above the answer box. It is to aid you to answer in the correct box and is not the exact question. You should refer to the original question in the question booklet.
11. You are allowed to use pencils, ball-pens or fountain pens, as you like as long as it is legible (no red color, please).
12. Unless otherwise stated in the question, when we ask for the worst-case big-O running time of an algorithm we always mean to give the tightest possible answer.
13. Unless otherwise stated in the question, we will assume that operators behave as per Java (e.g., 5/2 will evaluate to 2). However, pseudocode does not necessarily satisfy Java syntax (unless stated otherwise) and things that do not compile as legal Java are not necessarily bugs (as long as they are clear pseudocode).
14. Unless otherwise stated in the question, we are not concerned with overflow errors, e.g., storing the value $2^{245,546,983}$ in an integer.

GOOD LUCK!

This page is intentionally left blank.

It may be used as scratch paper.

Question 1: Hash Hash [16 marks]

A. Consider a hash table with 50 slots. Assuming collisions are resolved by chaining, and the hash function maps each key to a bucket chosen uniformly at random. (Ignore the fact that hash functions are deterministic.) What is the probability that the last five buckets of the table are empty after three insertions?

[2 marks]

1. $49/50$
2. $(9/10)^3$
3. $(45/50) \times (44/50) \times (43/50)$
4. $(47/50)^3$

B. What is the worst-case time complexity of deleting an element in a hash table with chaining? Let n be the number of keys in the hash table.

[2 marks]

1. $\Theta(1)$
2. $\Theta(\log n)$
3. $\Theta(\sqrt{n})$
4. $\Theta(n)$
5. $\Theta(n \log n)$
6. None of the above.

C. Which of the following statements about HashMap in Java is <u>FALSE</u>?

[2 marks]

1. HashMap stores key-value pairs and allows expected constant-time access to values based on their keys under the simple uniform hashing assumption.
2. If two keys have the same hash code, they are stored in the same bucket and accessed using chaining.
3. Iterating over the elements in a HashMap returns the elements in the order they were inserted.
4. When adding elements to a HashMap, if the load factor exceeds a certain threshold, the size of the HashMap is increased to maintain a good balance between space and time complexity.

D. You are given n keys. You want to hash these keys <u>perfectly</u> to m slots, where $m \geq n$. This means that there should be no collisions between the keys. What is the probability that a random function from the n keys to the m slots achieves this?

[2 marks]

1. $(n/m)^n$
2. $(1 - 1/m)^n$
3. $m!/m^n$
4. $m!/(m^n \cdot (m-n)!)$
5. None of the above.

E. If you choose a random hash function from n keys to m slots, what is the expected number of pairs of distinct keys that collide?

[2 marks]

1. n/m
2. n^2/m
3. $n(n-1)/(2m)$
4. n/m^2
5. None of the above

F. Suppose you are hashing n keys to a hash table with m slots. Under the simple uniform hashing assumption, the probability that the 1st, 10th, and n'th keys hash into the same bucket is: [2 marks]

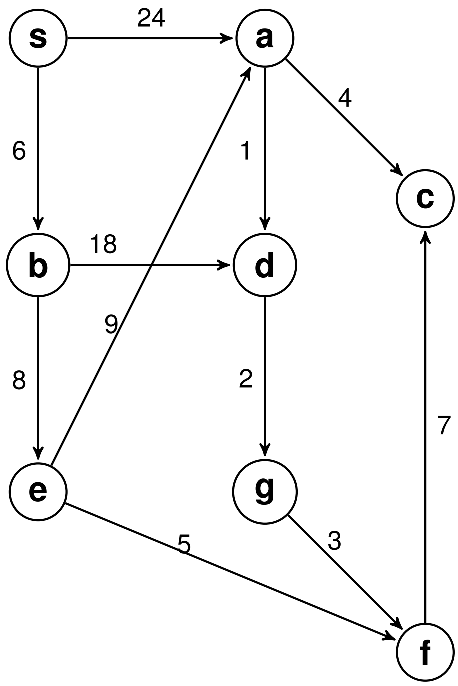
1. $1/n^2$
2. $1/m^2$
3. $1/n^3$
4. $1/m^3$
5. $3/m$

G. Given an array A of n distinct positive integers, we want an algorithm to decide if there exist indices i, j, k, ℓ such that $A[\ell] = A[i] \cdot A[j] \cdot A[k]$. Here, i, j, k, and ℓ need not be distinct. Which of the following yields a correct algorithm with the best expected running time? (Make the simple uniform hashing assumption if necessary.) [4 marks]

1. Create a new array T which stores $A[i] \cdot A[j] \cdot A[k]$ for each triple of indices i, j, k. Sort A and T, and use the merge procedure from MergeSort to check if there's a common element in T and A.
2. Create a new array T which stores $A[i] \cdot A[j] \cdot A[k]$ for each triple of indices i, j, k. Insert the elements of P into a hash table H of size $O(n^2)$. Look up each element of A in H.
3. Create a new array T which stores $A[i] \cdot A[j] \cdot A[k]$ for each triple of indices i, j, k. Insert the elements of P into a hash table H of size $O(n^3)$. Look up each element of A in H.
4. Create a new array P which stores $A[i] \cdot A[j]$ for each pair of indices i, j, k. Insert each element of P into a hash table of size $O(n^2)$, checking each time if there's a collision.
5. Create a new array P which stores $A[i] \cdot A[j]$ for each pair of indices i, j, and a new array Q which stores $A[i]/A[j]$ for each pair of indices i, j. Sort P and Q, and use the merge procedure from MergeSort to check if there's a common element in P and Q.
6. Create a new array P which stores $A[i] \cdot A[j]$ for each pair of indices i, j, and a new array Q which stores $A[i]/A[j]$ for each pair of indices i, j. Insert the elements of P into a hash table H of size $O(n)$. Look up each element of Q in H.
7. Create a new array P which stores $A[i] \cdot A[j]$ for each pair of indices i, j, and a new array Q which stores $A[i]/A[j]$ for each pair of indices i, j. Insert the elements of P into a hash table H of size $O(n^2)$. Look up each element of Q in H.

Question 2: Graph Walking [19 marks]

Consider the following weighted directed graph. The source node is s.



Note: The graph is stored in an adjacency list format, where the adjacency list for each vertex is in an arbitrary, unknown order

A. Which of the following may be the sequence of vertices visited by BFS? [2 marks]

1. s,a,c,d,g,f,e,b
2. s,b,a,e,d,c,f,g
3. s,b,a,e,c,d,g,f
4. s,b,e,f,c,a,d,g
5. s,b,d,g,f,e,a,c
6. s,b,e,f,a,d,c,g

B. Which of the following may be the sequence of vertices visited by (pre-order) DFS? [2 marks]

1. s,a,c,d,g,f,e,b
2. s,b,a,e,d,c,f,g
3. s,b,a,c,d,e,g,f
4. s,b,e,f,c,a,d,g
5. s,b,d,g,f,e,a,c
6. s,b,e,f,a,d,c,g

C. Which of the following may be the sequence of vertices extracted from the priority queue by Dijkstra's algorithm? [2 marks]

1. s,a,c,d,g,f,e,b
2. s,b,a,e,d,c,f,g
3. s,b,a,c,d,e,g,f
4. s,b,e,f,c,a,d,g
5. s,b,d,g,f,e,a,c
6. s,b,e,f,a,d,c,g

D. What is the smallest number of iterations after which the Bellman-Ford algorithm is guaranteed to have the correct distance estimates at each node (no matter what the sequence in which the edges are relaxed in each iteration)? [2 marks]

1. 2 iterations
2. 3 iterations
3. 4 iterations
4. 5 iterations
5. 6 iterations
6. 7 iterations

E. Which of the following is a topological sort? [2 marks]

1. s,a,c,d,g,f,e,b
2. s,b,a,e,d,c,f,g
3. s,b,e,a,d,g,f,c
4. s,b,e,f,a,d,g,c
5. s,b,d,g,f,e,a,c
6. s,b,e,f,a,d,c,g

F. Which of the edges is **not** present in a shortest path tree? [3 marks]

1. (s,b)
2. (a,c)
3. (e,a)
4. (b,e)
5. (f,c)
6. (e,f)

G. Ignore the orientations of the edges to get a weighted undirected graph. What is the total weight of its minimum spanning tree? [3 marks]

1. 25
2. 29
3. 32
4. 35
5. 38
6. None of the above.

H. Ignore the orientations of the edges to get a weighted undirected graph. If we ran Kruskal's algorithm on this graph, which is the <u>last</u> edge that would be added to the MST? [3 marks]

1. (s,a)
2. (s,b)
3. (b,e)
4. (a,e)
5. (b,d)
6. None of the above.

Question 3: Shuffling Cards [15 marks]

Howdy Harrini has a standard deck of 52 cards. Each time he shuffles the cards, he either moves the top card in the deck to the bottom, or he moves the bottom card in the deck to the top. For example, if the deck is $[A\heartsuit, 10\spadesuit, 9\clubsuit, K\heartsuit, \ldots, 5\diamondsuit]$, after one shuffle, he might end up with the deck $[10\spadesuit, 9\clubsuit, K\heartsuit, \ldots, 5\diamondsuit, A\heartsuit]$, where the "…" part is the same in both decks.

He asks you how good his shuffling technique is. You decide to model the shuffling dynamics using a graph. Construct a graph G where each vertex is a possible state of the deck. For two vertices A and B, there is an edge (A, B) if one of Howdy's shuffles can change the deck in state A to the deck in state B.

A. You want your graphical formulation to be as simple as possible (meaning, least number of nodes, edges, and weights). How should you think about G?

[3 marks]

1. A directed acyclic graph
2. A tree
3. An unweighted undirected graph
4. An unweighted directed graph
5. A weighted undirected graph
6. A weighted directed graph

B. How many connected components does G have? [3 marks]

1. 52!
2. 51!
3. 50!
4. 50
5. 51
6. 52
7. None of the above

C. What is the <u>diameter</u> of each connected component of G, i.e., the largest distance between any pair of vertices in a connected component? [3 marks]

1. 1
2. 25
3. 26
4. 50
5. 52
6. None of the above

Howdy decides to change his shuffling method. Now, in each shuffle, he will pull an arbitrary card from the deck and place it on top. Let H be the graph for this new shuffling method. Again, each vertex corresponds to a possible state of the deck, and there is an edge (A, B) if the deck in state A changes to the deck in state B by one shuffle.

D. You want your graphical formulation to be as simple as possible (meaning, least number of nodes, edges, and weights). How should you think about H? [3 marks]

1. A directed acyclic graph
2. A tree
3. An unweighted undirected graph
4. An unweighted directed graph
5. A weighted undirected graph
6. A weighted directed graph

E. Which of the following is true for the graph H?

[3 marks]

(I) There exist vertices A and B such that there is no path from A to B and no path from B to A.

(II) For all vertices A and B, either there is a path from A to B but no path from B to A, or there is a path from B to A but no path from A to B.

(III) There exist vertices A and B such that there is a path from A to B but no path from B to A.

(IV) For all vertices A and B, there is a path from A to B, and there is a path from B to A.

(V) None of the above.

Question 4: Travel Scheduler [21 marks]

You launch an enormously successful startup after graduating from NUS, make a bunch of money, and decide to spend your time traveling the world. Each month, you ask GuPTa (your personal assistant AI) to plan the itinerary for your next vacation.

Information about flights and their costs is represented as a directed graph G, where each vertex is an airport, and where there is an edge (x, y) with cost c_{xy} if and only if there is a flight from airport x to y with cost c_{xy} dollars. (Obviously, costs cannot be negative.) As usual, the cost of a path is the sum of the costs of the edges along the path.

A. Suppose you want GuPTa to solve the single-source shortest path problem on G. If GuPTa runs Dijkstra's algorithm using an AVL tree implementation of priority queues, what is the worst case running time in terms of n, the number of airports? [3 marks]

1. $O(n)$
2. $O(n \log n)$
3. $O(n^2)$
4. $O(n^2 \log n)$
5. $O(n^2 \log^2 n)$
6. None of the above.

B. You ask GuPTa to find the minimum cost path in G from a source vertex s to a destination vertex d that uses at most two edges (i.e., at most one stopover). Which of the following is a suitable algorithm to run? [4 marks]

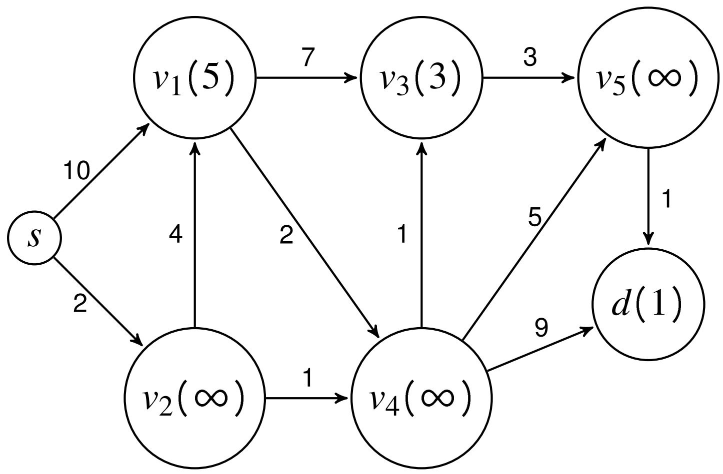
1. Run Dijkstra on G. If in the returned tree, there is a path with at most 2 edges from s to d, return it. Otherwise, report that none exists.
2. Construct a new graph H as follows. For every vertex v in G, create two vertices $v^{(1)}$ and $v^{(2)}$ in H. If there is an edge (u, v) in G with cost c_{uv}, add in H the edge $(u^{(1)}, v^{(2)})$ with cost c_{uv}. Run Dijkstra on H to find the minimum cost path from $s^{(1)}$ to $d^{(2)}$.
3. Construct a new graph H as follows. For every vertex v in G, create two vertices $v^{(1)}$ and $v^{(2)}$ in H. If there is an edge (u, v) in G with cost c_{uv}, add in H the three edges $(u^{(1)}, v^{(1)}), (u^{(1)}, v^{(2)}), (u^{(2)}, v^{(2)})$ with cost c_{uv}. Run Dijkstra on H to find the minimum cost path from $s^{(1)}$ to $d^{(2)}$.
4. Construct a new graph H as follows. For every vertex v in G, create three vertices $v^{(1)}, v^{(2)}, v^{(3)}$ in H. If there is an edge (u, v) in G with cost c_{uv}, add in H the two edges $(u^{(1)}, v^{(2)}), (u^{(2)}, v^{(3)})$ with cost c_{uv}. Run Dijkstra on H to find the minimum cost path from $s^{(1)}$ to $d^{(3)}$.
5. Construct a new graph H as follows. For every vertex v in G, create three vertices $v^{(1)}, v^{(2)}, v^{(3)}$ in H. If there is an edge (u, v) in G with cost c_{uv}, add in H the two edges

$(u^{(1)}, v^{(2)}), (u^{(2)}, v^{(3)})$ with cost c_{uv}. Also, for each node v in G, add in H the two edges $(v^{(1)}, v^{(2)})$ and $(v^{(2)}, v^{(3)})$ with cost 0. Run Dijkstra on H to find the minimum cost path from $s^{(1)}$ to $d^{(3)}$.

6. Construct a new graph H as follows. For every vertex v in G, create three vertices $v^{(1)}, v^{(2)}, v^{(3)}$ in H. If there is an edge (u, v) in G with cost c_{uv}, add in H the five edges $(u^{(1)}, v^{(1)}), (u^{(2)}, v^{(2)}), (u^{(3)}, v^{(3)}), (u^{(1)}, v^{(2)}), (u^{(2)}, v^{(3)})$ with cost c_{uv}. Run Dijkstra on H to find the minimum cost path from $s^{(1)}$ to $d^{(3)}$.

7. None of the above.

You find that with only one stopover, you are not able to reach some of the exotic destinations you want to go to. Hence, you remove the restriction in part **A** about having only one stopover. Instead, you require that there should not be more than two consecutive flight legs without an intervening hotel stay. Assume that each node in G, other than the source node s, is also labeled with the hotel stay cost at the location, shown inside parentheses in the example below.



Here, the path $s \to v_2 \to v_4 \to v_5$ has infinite cost because if you didn't make a rest stop at v_2 or v_4, you can't take the third flight from v_4 to v_5. On the other hand, the path $s \to v_1 \to v_4 \to v_5$ with hotel stay at v_1 has cost $10 + 2 + 5 = 17$ for the three flights and cost 5 for the stop at v_1, for a total cost of 22.

C. For the example above, what is the minimum total cost of a path from s to v_5, respecting the condition about hotel stays? [3 marks]

1. 7
2. 10
3. 13
4. 18
5. 22
6. None of the above.

D. For the example above, what is the minimum total cost of a path from s to d, respecting the condition about hotel stays? [3 marks]

1. 9
2. 12
3. 17
4. 21
5. 24
6. 25

E. In general, suppose we have a graph G as above, but in addition to having each edge (x, y) having cost c_{xy}, each vertex x is also labeled with the cost h_x for the hotel stay cost at the location of x. What algorithm should GuPTa run to find the minimum total cost path from a source vertex s to a destination vertex d respecting the conditions above? [4 marks]

1. Construct a new graph H as follows. For every vertex v in G, create two vertices $v^{(1)}$ and $v^{(2)}$ in H. If there is an edge (u, v) in G, add in H the edge $(u^{(1)}, v^{(2)})$ with cost $c_{uv} + h_u$. Run Dijkstra on H to find the minimum cost path from $s^{(1)}$ to $d^{(2)}$.
2. Construct a new graph H as follows. For every vertex v in G, create two vertices $v^{(1)}$ and $v^{(2)}$ in H. If there is an edge (u, v) in G, add in H the edges $(u^{(1)}, v^{(2)})$ with cost c_{uv} and $(u^{(2)}, v^{(1)})$ with cost h_u. Run Dijkstra on H to find the minimum cost path from $s^{(1)}$ to $d^{(1)}$.
3. Construct a new graph H as follows. For every vertex v in G, create three vertices $v^{(1)}, v^{(2)}, v^{(3)}$ in H. If there is an edge (u, v) in G, add in H the two edges $(u^{(1)}, v^{(2)})$ and $(u^{(2)}, v^{(3)})$ with costs c_{uv} and h_u respectively. Run Dijkstra on H with source $s^{(1)}$, and return the minimum of the distances to $d^{(1)}, d^{(2)}$, and $d^{(3)}$.
4. Construct a new graph H as follows. For every vertex v in G, create three vertices $v^{(1)}, v^{(2)}, v^{(3)}$ in H. If there is an edge (u, v) in G, add in H the two edges $(u^{(1)}, v^{(2)})$ and $(u^{(2)}, v^{(3)})$ with costs c_{uv} each. Also, for each node v in G, add in H the two edges $(v^{(2)}, v^{(1)})$ and $(v^{(3)}, v^{(1)})$ with costs h_v. Run Dijkstra on H with source $s^{(1)}$, and return the minimum of the distances to $d^{(1)}, d^{(2)}$, and $d^{(3)}$.
5. Construct a new graph H as follows. For every vertex v in G, create three vertices $v^{(1)}, v^{(2)}, v^{(3)}$ in H. If there is an edge (u, v) in G with cost c_{uv}, add in H the two edges $(u^{(1)}, v^{(2)})$ and $(u^{(2)}, v^{(3)})$ with costs c_{uv} each. Also, for each node v in G, add in H the two edges $(v^{(2)}, v^{(1)})$ and $(v^{(3)}, v^{(2)})$ with costs h_v each. Run Dijkstra on H with source $s^{(1)}$, and return the minimum of the distances to $d^{(1)}, d^{(2)}$, and $d^{(3)}$.
6. None of the above.

You become more conscious about your impact on the environment, and you decide to put a budget on your carbon footprint. Each month, you ask GuPTa to find a minimum-cost flight itinerary from a source s to a destination d with total carbon footprint at most B. (You remove your other preferences about stopovers.)

GuPTa possesses a graph G as above, but where each edge (x, y) is now annotated with a pair (c_{xy}, f_{xy}) where c_{xy} is the cost and f_{xy} is the carbon footprint of the flight from airport x to y.

F. Which of the following modifications of Dijkstra solves the problem of finding a minimum cost path in G from a source s to a destination d with total carbon footprint at most B?

[4 marks]

1. Run Dijkstra as usual. Follow the parent pointers back from d to s to check if total carbon footprint is $\leq B$. If yes, output the path; otherwise, report that no solution exists.
2. Maintain a priority queue as before, prioritized by by the estimated distances `distTo[v]`, but with each node `v`, store also a carbon footprint value `foot[v]` in the priority queue. Initialize all `foot[v]` to ∞, except `foot[s]`$= 0$. When relaxing edge (u, v), if both `distTo[v]` > `distTo[u]` $+c_{uv}$ and `foot[u]` $+f_{uv} \leq B$, use `decreaseKey` to update `distTo`$[v]$ and also update `foot[v]` to `foot[u]` $+ f_{uv}$. After d is extracted, if `foot[d]` $\leq B$, report path as usual, and otherwise, report no solution exists.
3. Maintain a priority queue as before, prioritized by by the estimated distances `distTo[v]`, but with each node `v`, store also a pointer `foot[v]` to a set of carbon footprint values. When relaxing edge (u, v), use `decreaseKey` to update `distTo[v]` and also add to the set at `foot[v]` the values $f + f_{uv}$ for each f in the linked list at `foot[u]`. At the end, check whether `foot[d]` contains a value less than or equal to B.
4. Maintain a priority queue, prioritized by carbon footprint values `foot[v]` instead of estimated distances `distTo[v]`. Initialize all `foot[v]` to ∞, except `foot[s]`$= 0$. When relaxing edge (u, v), if both `distTo[v]` > `distTo[u]` $+c_{uv}$ and `foot[u]` $+f_{uv} \leq \min(B,$ `foot[v]`$)$, use `decreaseKey` to update `foot[v]` to `foot[u]` $+f_{uv}$ and also update `distTo[v]`. After d is extracted, if `foot[d]` $\leq B$, report path as usual, and otherwise, report no solution exists.
5. None of the above.

Question 5: Longest Sawtooth Subsequence [14 marks]

A sawtooth sequence is a sequence of numbers $s_1, \ldots, s_m$, where for every $i \le m-2$, either $s_i < s_{i+1} > s_{i+2}$ or $s_i > s_{i+1} < s_{i+2}$. All sequences of length 1 and 2 are trivially sawtooth.

We want to design a dynamic program that given an input sequence $x_1, \ldots, x_n$ of distinct numbers, finds the length of the longest sawtooth subsequence of $x_1, \ldots, x_n$. To this end, define:

- $A(i)$ to be the length of the longest sawtooth subsequence in the sequence $x_1, \ldots, x_i$ and whose last pair is ascending, and
- $D(i)$ to be the length of the longest sawtooth subsequence in the sequence $x_1, \ldots, x_i$ and whose last pair is descending.

$A(i)$ and $D(i)$ are always at least 1, because a length-1 subsequence is assumed to trivially satisfy the conditions. Consider the example sequence:

$$x_1 = 11, \qquad x_2 = 100, \qquad x_3 = 80, \qquad x_4 = 42, \qquad x_5 = 60, \qquad x_6 = 1$$

Here, $A(3) = 2$ because there are length-2 sawtooth subsequences (e.g, x_1, x_3) of x_1, x_2, x_3 that end on an ascending pair, while $D(3) = 3$ because there are length-3 sawtooth subsequences (x_1, x_2, x_3) that end on a descending pair. Similarly, $A(4) = 2$, while $D(4) = 3$.

A. What are $A(5)$ and $D(5)$ for the example? [2 marks]

1. 2 and 3
2. 3 and 2
3. 4 and 2
4. 4 and 3
5. 4 and 4
6. None of the above.

B. What are $A(6)$ and $D(6)$ for the example? [2 marks]

1. 1 and 4
2. 3 and 2
3. 4 and 2
4. 4 and 3
5. 4 and 4
6. None of the above.

C. What is a recurrence relation to compute $A(i)$ and $D(i)$? [3 marks]

1. $A(i) = A(i-1)$ if $x_i < x_{i-1}$, else $A(i) = 1 + A(i-1)$
   $D(i) = D(i-1)$ if $x_i > x_{i-1}$, else $D(i) = 1 + D(i-1)$.
2. $A(i) = A(i-1)$ if $x_i < x_{i-1}$, else $A(i) = 1 + D(i-1)$
   $D(i) = D(i-1)$ if $x_i > x_{i-1}$, else $D(i) = 1 + A(i-1)$.
3. $A(i) = 1 + \max\{A(j) : 1 \le j < i, x_j < x_i\}$,
   $D(i) = 1 + \max\{D(j) : 1 \le j < i, x_j > x_i\}$,
4. $A(i) = 1 + \max\{D(j) : 1 \le j < i, x_j < x_i\}$,
   $D(i) = 1 + \max\{A(j) : 1 \le j < i, x_j > x_i\}$,
5. None of the above.

D. What are the base cases of the recurrence relation? [3 marks]

1. $A(1) = 1, D(1) = 1$
2. $A(1) = 1, D(1) = 1$; $A(2) = 1$ if $x_2 > x_1$, and $A(2) = 2$ otherwise; $D(2) = 1$ if $x_2 < x_1$, and $D(2) = 2$ otherwise.
3. For all i, $A(i) = 1$ if $x_i = \min(x_1, \ldots, x_i)$, and $D(i) = 1$ if $x_i = \max(x_1, \ldots, x_i)$.
4. For all i, $A(i) = 1$ if $x_i = \max(x_1, \ldots, x_i)$, and $D(i) = 1$ if $x_i = \min(x_1, \ldots, x_i)$.
5. None of the above.

E. The length of the longest sawtooth subsequence of $x_1, x_2, \ldots, x_n$ is equal to $\max(A(n), D(n))$. [2 marks]

1. True
2. False

F. Use your answer to part **C** to design a dynamic programming algorithm to compute the values $A(1), \ldots, A(n), D(1), \ldots, D(n)$. Which of the following is the tightest bound on the running time of this algorithm? [2 marks]

1. $O(\log n)$
2. $O(n)$
3. $O(n \log n)$
4. $O(n^2)$
5. $O(n^3)$
6. None of the above.

Question 6: Assorted Selections [15 marks]

A. Choose the tightest possible bound from the available options for the following function:

$$T(n) = n^{1+1/\log n}$$

[2 marks]

1. $O(\log n)$
2. $O(n)$
3. $O(n \log n)$
4. $O(n^2)$
5. $O(n^3)$
6. $O(2^n)$

B. Choose the tightest possible worst-case bound from the available options for the following recurrence, assuming that $T(1) = 1$:

$$T(n) = \begin{cases} T(\sqrt{n}) + 1, & \text{if } n \text{ is a power of 2} \\ T(n-1) + 1, & \text{otherwise.} \end{cases}$$

[2 marks]

1. $O(\log \log n)$
2. $O(\log n)$
3. $O(\sqrt{n})$
4. $O(n)$
5. $O(n^2)$
6. $O(2^n)$

C. Suppose you have a perfectly balanced binary search tree with n nodes, where n is a power of 2. You run an algorithm `Process` on each of the n nodes, invoking it first on the leaves, then on nodes one level above, then on nodes two levels above, and so on, ending at the root. The cost of running `Process` on a node x is the number of descendants of x. What is the amortized cost of `Process`? [3 marks]

1. $\Theta(1)$
2. $\Theta(\log \log n)$
3. $\Theta(\log n)$
4. $\Theta(\sqrt{n})$
5. $\Theta(n)$
6. $\Theta(n \log n)$

D. Suppose you have a perfectly balanced binary search tree with n nodes, where n is a power of 2. You run an algorithm `Process2` on each of the n nodes, invoking it first on the leaves, then on nodes one level above, then on nodes two levels above, and so on, ending at the root. The cost of running `Process2` on a node x is the height of x. What is the amortized cost of `Process2`? [3 marks]

1. $\Theta(1)$
2. $\Theta(\log \log n)$
3. $\Theta(\sqrt{\log n})$
4. $\Theta(\log n)$
5. $\Theta(n)$
6. $\Theta(n \log n)$

E. You are the maintainer of the *CS2040S Fan Club*. You store the members in an AVL tree, indexed by their names. Each member is added exactly once to the tree on their *membership day*. In addition to supporting searching, addition and deletion of members by their name, you want to support a `deleteOldest` operation that removes the member who has the earliest membership day. What would be your recommended solution? [3 marks]

1. Augment each AVL tree node x with the membership date of x.
2. Augment each AVL tree node x with the latest membership date for any node in the subtree rooted at x.
3. Augment each AVL tree node x with the earliest membership date for any node in the subtree rooted at x.
4. Maintain a separate queue with the members in the order they were inserted (oldest at the front).
5. Maintain a separate stack with the members in the order they were inserted (oldest at the back).
6. None of the above.

F. Recall the disjoint set data structure described in class, where the `union` operation is implemented as a weighted union (the larger tree's root is the parent of the smaller tree's root). Suppose you have an arbitrary sequence of n many `union` and `find` operations on k nodes $x_1, \ldots, x_k$. Suppose $n > 2k$. Initially, all k nodes are disjoint singleton sets. On which of the following sequences is the total running time going to be $\Omega(n \log k)$? [2 marks]

1. $\texttt{find}(x_1), \texttt{find}(x_1), \ldots, \texttt{find}(x_1), \texttt{union}(x_1, x_2), \texttt{union}(x_1, x_3), \ldots, \texttt{union}(x_1, x_k)$
2. $\texttt{find}(x_k), \texttt{find}(x_k), \ldots, \texttt{find}(x_k), \texttt{union}(x_1, x_2), \texttt{union}(x_1, x_3), \ldots, \texttt{union}(x_1, x_k)$
3. $\texttt{union}(x_1, x_2), \texttt{union}(x_1, x_3), \ldots, \texttt{union}(x_1, x_k), \texttt{find}(x_1), \texttt{find}(x_1), \ldots, \texttt{find}(x_1)$
4. $\texttt{union}(x_1, x_2), \texttt{union}(x_1, x_3), \ldots, \texttt{union}(x_1, x_k), \texttt{find}(x_k), \texttt{find}(x_k), \ldots, \texttt{find}(x_k)$
5. None of the above.

Question 7: Shortcutting (Just for Fun!) [0 marks]

Consider the directed path P_n on n vertices: the vertex set V is $\{1,\ldots,n\}$ and the edge set E is $\{(i,i+1) : 1 \le i < n\}$. A d-diameter shortcut set is a set S of additional "shortcut edges" such that, if H is the graph on $\{1,\ldots,n\}$ with edge set $E \cup S$, then:

(i) for any $j > i$, there is a path of length at most d from i to j in H, and

(ii) for any shortcut edge $(i,j) \in S$, $j > i$.

[0 mark]

A. Find a set of $O(n \log n)$ shortcut edges that form a 2-diameter shortcut set.

B. What is the smallest 3-diameter shortcut set that you can find? What about d-diameter for a general d?

— END OF PAPER —

Question 1A Probability of last five 5 buckets empty after 3 insertions [2 marks]

- ◯ $49/50$
- ◯ $(45/50)\times(44/50)\times(43/50)$
- ◯ $(9/10)^3$
- ◯ $(47/50)^3$

($(9/10)^3$)

Question 1B Complexity of hash table deletion [2 marks]

- ◯ $\Theta(1)$
- ◯ $\Theta(n\log n)$
- ◯ $\Theta(\log n)$
- ◯ $\Theta(n)$
- ◯ $\Theta(n)$
- ◯ $\Theta(n\log n)$

($\Theta(n)$. Either of the two circles would be marked correct.)

Question 1C False statement for Java HashMap [2 marks]

- ◯ HashMap stores key-value pairs and allows expected constant-time access to values based on their keys under the simple uniform hashing assumption.
- ◯ If two keys have the same hash code, they are stored in the same bucket and accessed using chaining.
- ◯ Iterating over the elements in a HashMap returns the elements in the order they were inserted.
- ◯ When adding elements to a HashMap, if the load factor exceeds a certain threshold, the size of the HashMap is increased to maintain a good balance between space and time complexity.

(Iterating over the elements in a HashMap returns the elements in the order they were inserted.

Question 1D Probability of perfect hashing for random hash function [2 marks]

- ◯ $(n/m)^n$
- ◯ $m!/(m^n\cdot(m-n)!)$
- ◯ $(1-1/m)^n$
- ◯ $m!/m^n$
- ◯ None of the above.

($m!/(m^n\cdot(m-n)!)$)

Question 1E Expected number of collisions for random hash function [2 marks]

- ○ n/m
- ○ n^2/m
- ○ $n(n-1)/(2m)$
- ○ n/m^2
- ○ None of the above.

$(n(n-1)/(2m))$

Question 1F Probability of 3-way collision under SUHA [2 marks]

- ○ $1/n^2$
- ○ $1/m^2$
- ○ $1/n^3$
- ○ $1/m^3$
- ○ $3/m$

$(1/m^2)$

Question 1G How to find if there's a product of a triple in an array? [4 marks]

- ◯ Create a new array T which stores $A[i] \cdot A[j] \cdot A[k]$ for each triple of indices i, j, k. Sort A and T, and use the merge procedure from MergeSort to check if there's a common element in T and A.
- ◯ Create a new array T which stores $A[i] \cdot A[j] \cdot A[k]$ for each triple of indices i, j, k. Insert the elements of P into a hash table H of size $O(n^2)$. Look up each element of A in H.
- ◯ Create a new array T which stores $A[i] \cdot A[j] \cdot A[k]$ for each triple of indices i, j, k. Insert the elements of P into a hash table H of size $O(n^3)$. Look up each element of A in H.
- ◯ Create a new array P which stores $A[i] \cdot A[j]$ for each pair of indices i, j, k. Insert each element of P into a hash table of size $O(n^2)$, checking each time if there's a collision.
- ◯ Create a new array P which stores $A[i] \cdot A[j]$ for each pair of indices i, j, and a new array Q which stores $A[i]/A[j]$ for each pair of indices i, j. Sort P and Q, and use the merge procedure from MergeSort to check if there's a common element in P and Q.
- ◯ Create a new array P which stores $A[i] \cdot A[j]$ for each pair of indices i, j, and a new array Q which stores $A[i]/A[j]$ for each pair of indices i, j. Insert the elements of P into a hash table H of size $O(n)$. Look up each element of Q in H.
- ◯ Create a new array P which stores $A[i] \cdot A[j]$ for each pair of indices i, j, and a new array Q which stores $A[i]/A[j]$ for each pair of indices i, j. Insert the elements of P into a hash table H of size $O(n^2)$. Look up each element of Q in H.

(Create a new array P which stores $A[i] \cdot A[j]$ for each pair of indices i, j, and a new array Q which stores $A[i]/A[j]$ for each pair of indices i, j. Insert the elements of P into a hash table H of size $O(n^2)$. Look up each element of Q in H.)

Question 2A Visit sequence for BFS? [2 marks]

- ◯ s, a, c, d, g, f, e, b
- ◯ s, b, a, e, c, d, g, f
- ◯ s, b, d, g, f, e, a, c
- ◯ s, b, a, e, d, c, f, g
- ◯ s, b, e, f, c, a, d, g
- ◯ s, b, e, f, a, d, c, g

(s, b, a, e, d, c, f, g)

Question 2B Visit sequence for DFS? [2 marks]

- ◯ s, a, c, d, g, f, e, b
- ◯ s, b, e, f, c, a, d, g
- ◯ s, b, a, e, d, c, f, g
- ◯ s, b, d, g, f, e, a, c
- ◯ s, b, a, c, d, e, g, f
- ◯ s, b, e, f, a, d, c, g

(s, b, e, f, c, a, d, g)

Question 2C Visit sequence for Dijkstra? [2 marks]

◯ s,a,c,d,g,f,e,b	◯ s,b,a,c,d,e,g,f	◯ s,b,d,g,f,e,a,c
◯ s,b,a,e,d,c,f,g	◯ s,b,e,f,c,a,d,g	◯ s,b,e,f,a,d,c,g

(s,b,e,f,a,d,c,g)

Question 2D Bellman-Ford convergence [2 marks]

◯ 2 iterations	◯ 4 iterations	◯ 6 iterations
◯ 3 iterations	◯ 5 iterations	◯ 7 iterations

(4 iterations)

Question 2E Topological sort [2 marks]

◯ s,b,a,e,d,c,f,g	◯ s,b,e,a,d,g,f,c	◯ s,b,e,g,f,e,a,c
◯ s,a,c,d,g,f,e,b	◯ s,b,e,f,a,d,g,c	◯ s,b,e,f,a,d,c,g

(s,b,e,a,d,g,f,c)

Question 2F Not in shortest-path tree [3 marks]

◯ (s,b)	◯ (e,a)	◯ (f,c)
◯ (a,c)	◯ (b,e)	◯ (e,f)

$((a,c))$

Question 2G Weight of MST [3 marks]

◯ 25	◯ 32	◯ 38
◯ 29	◯ 35	◯ None of the above.

(29)

Question 2H Last edge added by Kruskal [3 marks]

- ◯ (s,a)
- ◯ (b,e)
- ◯ (b,d)
- ◯ (s,b)
- ◯ (a,e)
- ◯ None of the above.

((b,e))

Question 3A Graph model type [3 marks]

- ◯ A directed acyclic graph
- ◯ An unweighted directed graph
- ◯ A tree
- ◯ A weighted undirected graph
- ◯ An unweighted undirected graph
- ◯ A weighted directed graph

(An unweighted undirected graph)

Question 3B Number of connected components [3 marks]

- ◯ 52!
- ◯ 51
- ◯ 51!
- ◯ 52
- ◯ 50!
- ◯ 50
- ◯ None of the above

(51!)

Question 3C Diameter of connected component [3 marks]

- ◯ 1
- ◯ 50
- ◯ 25
- ◯ 52
- ◯ 26
- ◯ None of the above

(26)

Question 3D Graph model type for *H* [3 marks]

- ◯ A directed acyclic graph
- ◯ An unweighted directed graph
- ◯ A tree
- ◯ A weighted undirected graph
- ◯ An unweighted undirected graph
- ◯ A weighted directed graph

(An unweighted directed graph)

Question 3E Connectivity of H [3 marks]

- ◯ There exist vertices A and B such that there is no path from A to B and no path from B to A.
- ◯ For all vertices A and B, either there is a path from A to B but no path from B to A, or there is a path from B to A but no path from A to B.
- ◯ There exist vertices A and B such that there is a path from A to B but no path from B to A.
- ◯ For all vertices A and B, there is a path from A to B, and there is a path from B to A.
- ◯ None of the above.

(For all vertices A and B, there is a path from A to B, and there is a path from B to A.)

Question 4A Dijkstra runtime [3 marks]

- ◯ $O(n)$
- ◯ $O(n \log n)$
- ◯ $O(n^2)$
- ◯ $O(n^2 \log n)$
- ◯ $O(n^2 \log^2 n)$
- ◯ None of the above.

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

Question 4B Dijkstra with at most one stopover [4 marks]

- ◯ Run Dijkstra on G. If in the returned tree, there is a path with at most 2 edges from s to d, return it. Otherwise, report that none exists.
- ◯ Construct a new graph H as follows. For every vertex v in G, create two vertices $v^{(1)}$ and $v^{(2)}$ in H. If there is an edge (u,v) in G with cost c, add in H the edge $(u^{(1)},v^{(2)})$ with cost c. Run Dijkstra on H to find the minimum cost path from $s^{(1)}$ to $d^{(2)}$.
- ◯ Construct a new graph H as follows. For every vertex v in G, create two vertices $v^{(1)}$ and $v^{(2)}$ in H. If there is an edge (u,v) in G with cost c, add in H the three edges $(u^{(1)},v^{(1)}),(u^{(1)},v^{(2)}),(u^{(2)},v^{(2)})$ with cost c. Run Dijkstra on H to find the minimum cost path from $s^{(1)}$ to $d^{(2)}$.
- ◯ Construct a new graph H as follows. For every vertex v in G, create three vertices $v^{(1)},v^{(2)},v^{(3)}$ in H. If there is an edge (u,v) in G with cost c, add in H the two edges $(u^{(1)},v^{(2)}),(u^{(2)},v^{(3)})$ with cost c. Run Dijkstra on H to find the minimum cost path from $s^{(1)}$ to $d^{(3)}$.
- ◯ Construct a new graph H as follows. For every vertex v in G, create three vertices $v^{(1)},v^{(2)},v^{(3)}$ in H. If there is an edge (u,v) in G with cost c, add in H the two edges $(u^{(1)},v^{(2)}),(u^{(2)},v^{(3)})$ with cost c. Also, for each node v in G, add in H the two edges $(v^{(1)},v^{(2)})$ and $(v^{(2)},v^{(3)})$ with cost 0. Run Dijkstra on H to find the minimum cost path from $s^{(1)}$ to $d^{(3)}$.
- ◯ Construct a new graph H as follows. For every vertex v in G, create three vertices $v^{(1)},v^{(2)},v^{(3)}$ in H. If there is an edge (u,v) in G with cost c, add in H the five edges $(u^{(1)},v^{(1)}),(u^{(2)},v^{(2)}),(u^{(3)},v^{(3)}),(u^{(1)},v^{(2)}),(u^{(2)},v^{(3)})$ with cost c. Run Dijkstra on H to find the minimum cost path from $s^{(1)}$ to $d^{(3)}$.
- ◯ None of the above.

(Construct a new graph H as follows. For every vertex v in G, create three vertices $v^{(1)},v^{(2)},v^{(3)}$ in H. If there is an edge (u,v) in G with cost c, add in H the two edges $(u^{(1)},v^{(2)}),(u^{(2)},v^{(3)})$ with cost c. Also, for each node v in G, add in H the two edges $(v^{(1)},v^{(2)})$ and $(v^{(2)},v^{(3)})$ with cost 0. Run Dijkstra on H to find the minimum cost path from $s^{(1)}$ to $d^{(3)}$.)

Question 4C Minimum cost to v_5 with rest stops [3 marks]

◯ 7	◯ 13	◯ 22
◯ 10	◯ 18	◯ None of the above.

(18)

Question 4D Minimum cost to d with rest stops [3 marks]

- ◯ 9
- ◯ 17
- ◯ 24
- ◯ 12
- ◯ 21
- ◯ 25

(21)

Question 4E Dijkstra with hotel stay constraint [4 marks]

◯ Construct a new graph H as follows. For every vertex v in G, create two vertices $v^{(1)}$ and $v^{(2)}$ in H. If there is an edge (u,v) in G, add in H the edge $(u^{(1)}, v^{(2)})$ with cost $c_{uv} + h_u$. Run Dijkstra on H to find the minimum cost path from $s^{(1)}$ to $d^{(2)}$.

◯ Construct a new graph H as follows. For every vertex v in G, create two vertices $v^{(1)}$ and $v^{(2)}$ in H. If there is an edge (u,v) in G, add in H the edges $(u^{(1)}, v^{(2)})$ with cost c_{uv} and $(u^{(2)}, v^{(1)})$ with cost h_u. Run Dijkstra on H to find the minimum cost path from $s^{(1)}$ to $d^{(1)}$.

◯ Construct a new graph H as follows. For every vertex v in G, create three vertices $v^{(1)}, v^{(2)}, v^{(3)}$ in H. If there is an edge (u,v) in G, add in H the two edges $(u^{(1)}, v^{(2)})$ and $(u^{(2)}, v^{(3)})$ with costs c_{uv} and h_u respectively. Run Dijkstra on H with source $s^{(1)}$, and return the minimum of the distances to $d^{(1)}, d^{(2)}$, and $d^{(3)}$.

◯ Construct a new graph H as follows. For every vertex v in G, create three vertices $v^{(1)}, v^{(2)}, v^{(3)}$ in H. If there is an edge (u,v) in G, add in H the two edges $(u^{(1)}, v^{(2)})$ and $(u^{(2)}, v^{(3)})$ with costs c_{uv} each. Also, for each node v in G, add in H the two edges $(v^{(2)}, v^{(1)})$ and $(v^{(3)}, v^{(1)})$ with costs h_v. Run Dijkstra on H with source $s^{(1)}$, and return the minimum of the distances to $d^{(1)}, d^{(2)}$, and $d^{(3)}$.

◯ Construct a new graph H as follows. For every vertex v in G, create three vertices $v^{(1)}, v^{(2)}, v^{(3)}$ in H. If there is an edge (u,v) in G with cost c_{uv}, add in H the two edges $(u^{(1)}, v^{(2)})$ and $(u^{(2)}, v^{(3)})$ with costs c_{uv} each. Also, for each node v in G, add in H the two edges $(v^{(2)}, v^{(1)})$ and $(v^{(3)}, v^{(2)})$ with costs h_v each. Run Dijkstra on H with source $s^{(1)}$, and return the minimum of the distances to $d^{(1)}, d^{(2)}$, and $d^{(3)}$.

◯ None of the above.

(Construct a new graph H as follows. For every vertex v in G, create three vertices $v^{(1)}, v^{(2)}, v^{(3)}$ in H. If there is an edge (u,v) in G with cost c_{uv}, add in H the two edges $(u^{(1)}, v^{(2)})$ and $(u^{(2)}, v^{(3)})$ with costs c_{uv} each. Also, for each node v in G, add in H the two edges $(v^{(2)}, v^{(1)})$ and $(v^{(3)}, v^{(1)})$ with costs h_v each. Run Dijkstra on H with source $s^{(1)}$, and return the minimum of the distances to $d^{(1)}, d^{(2)}$, and $d^{(3)}$.)

Question 4F Dijkstra with carbon footprint constraint [4 marks]

- ◯ Run Dijkstra as usual. Follow the parent pointers back from d to s to check if total carbon footprint is $\leq B$. If yes, output the path; otherwise, report that no solution exists.
- ◯ Maintain a priority queue as before, prioritized by by the estimated distances `distTo[v]`, but with each node `v`, store also a carbon footprint value `foot[v]` in the priority queue. Initialize all `foot[v]` to ∞, except `foot[s]`$= 0$. When relaxing edge (u,v), if both `distTo`[v] > `distTo`[u] $+c_{uv}$ and `foot[u]` $+f_{uv} \leq B$, use `decreaseKey` to update `distTo`$[v]$ and also update `foot[v]` to `foot[u]` $+ f_{uv}$. After d is extracted, if `foot[d]` $\leq B$, report path as usual, and otherwise, report no solution exists.
- ◯ Maintain a priority queue as before, prioritized by by the estimated distances `distTo[v]`, but with each node `v`, store also a pointer `foot[v]` to a set of carbon footprint values. When relaxing edge (u,v), use `decreaseKey` to update `distTo`[v] and also add to the set at `foot[v]` the values $f + f_{uv}$ for each f in the linked list at `foot[u]`. At the end, check whether `foot`[d] contains a value less than or equal to B.
- ◯ Maintain a priority queue, prioritized by carbon footprint values `foot[v]` instead of estimated distances `distTo[v]`. Initialize all `foot[v]` to ∞, except `foot[s]`$= 0$. When relaxing edge (u,v), if both `distTo`[v] > `distTo`[u] $+c_{uv}$ and `foot`[u] $+f_{uv} \leq$ min(B, `foot`[v]), use `decreaseKey` to update `foot`[v] to `foot`[u] $+f_{uv}$ and also update `distTo[v]`. After d is extracted, if `foot[d]` $\leq B$, report path as usual, and otherwise, report no solution exists.
- ◯ None of the above.

(None of the above. Check that the optimal substructure property does not hold for shortest paths with footprint $\leq B$)

Question 5A What are $A(5)$ and $D(5)$? [2 marks]

◯ 2 and 3	◯ 4 and 2	◯ 4 and 4
◯ 3 and 2	◯ 4 and 3	◯ None of the above.

(4 and 3)

Question 5B What are $A(6)$ and $D(6)$? [2 marks]

◯ 2 and 3	◯ 4 and 2	◯ 4 and 4
◯ 3 and 2	◯ 4 and 3	◯ None of the above.

(None of the above. Correct values are 4 and 5.)

Question 5C Recurrence relation [3 marks]

- ○ $A(i) = A(i-1)$ if $x_i < x_{i-1}$, else $A(i) = 1 + A(i-1)$
  $D(i) = D(i-1)$ if $x_i > x_{i-1}$, else $D(i) = 1 + D(i-1)$.
- ○ $A(i) = A(i-1)$ if $x_i < x_{i-1}$, else $A(i) = 1 + D(i-1)$
  $D(i) = D(i-1)$ if $x_i > x_{i-1}$, else $D(i) = 1 + A(i-1)$.
- ○ $A(i) = 1 + \max\{A(j) : 1 \le j < i, x_j < x_i\}$,
  $D(i) = 1 + \max\{D(j) : 1 \le j < i, x_j > x_i\}$.
- ○ $A(i) = 1 + \max\{D(j) : 1 \le j < i, x_j < x_i\}$,
  $D(i) = 1 + \max\{A(j) : 1 \le j < i, x_j > x_i\}$.
- ○ None of the above.

($A(i) = A(i-1)$ if $x_i < x_{i-1}$, else $A(i) = 1 + D(i-1)$
$D(i) = D(i-1)$ if $x_i > x_{i-1}$, else $D(i) = 1 + A(i-1)$.)

Question 5D Base case [3 marks]

- ○ $A(1) = 1, D(1) = 1$
- ○ $A(1) = 1, D(1) = 1$; $A(2) = 1$ if $x_2 > x_1$, and $A(2) = 2$ otherwise; $D(2) = 1$ if $x_2 < x_1$, and $D(2) = 2$ otherwise.
- ○ For all i, $A(i) = 1$ if $x_i = \min(x_1, \ldots, x_i)$, and $D(i) = 1$ if $x_i = \max(x_1, \ldots, x_i)$.
- ○ For all i, $A(i) = 1$ if $x_i = \max(x_1, \ldots, x_i)$, and $D(i) = 1$ if $x_i = \min(x_1, \ldots, x_i)$.
- ○ None of the above.

($A(1) = 1, D(1) = 1$.)

Question 5E Final answer equal to $\max(A(n), D(n))$? [2 marks]

- ○ True
- ○ False

(True)

Question 5F Running time. [2 marks]

- ○ $O(\log n)$
- ○ $O(n \log n)$
- ○ $O(n^3)$
- ○ $O(n)$
- ○ $O(n^2)$
- ○ None of the above.

($O(n)$)

Question 6A Tighest asymptotic upper bound [2 marks]

○ $O(\log n)$ ○ $O(n \log n)$ ○ $O(n^3)$
○ $O(n)$ ○ $O(n^2)$ ○ $O(2^n)$

($O(n)$)

Question 6B Asymptotic analysis of recurrence [2 marks]

○ $O(\log\log n)$ ○ $O(\sqrt{n})$ ○ $O(n^2)$
○ $O(\log n)$ ○ $O(n)$ ○ $O(2^n)$

($O(n)$)

Question 6C Amortized cost when each cost is the number of descendants [3 marks]

○ $\Theta(1)$ ○ $\Theta(\log n)$ ○ $\Theta(n)$
○ $\Theta(\log\log n)$ ○ $\Theta(\sqrt{n})$ ○ $\Theta(n \log n)$

($\Theta(\log n)$)

Question 6D Amortized cost when each cost is the height [3 marks]

○ $\Theta(1)$ ○ $\Theta(\log n)$ ○ $\Theta(n \log n)$
○ $\Theta(\log\log n)$ ○ $\Theta(\sqrt{n})$
○ $\Theta(\sqrt{\log n})$ ○ $\Theta(n)$

($\Theta(1)$)

Question 6E How to handle deletion of the oldest member? [3 marks]

- ◯ Augment each AVL tree node x with the membership date of x.
- ◯ Augment each AVL tree node x with the latest membership date for any node in the subtree rooted at x.
- ◯ Augment each AVL tree node x with the earliest membership date for any node in the subtree rooted at x.
- ◯ Maintain a separate queue with the members in the order they were inserted (oldest at the front).
- ◯ Maintain a separate stack with the members in the order they were inserted (oldest at the back).
- ◯ None of the above.

(Augment each AVL tree node x with the earliest membership date for any node in the subtree rooted at x. A separate queue can be used to efficiently implement the `deleteOldest` operation, but then it may be expensive to implement the deletion by name while keeping the queue and the tree consistent.)

Question 6F Worst case sequence for weighted union-find [2 marks]

- ◯ $\texttt{find}(x_1), \texttt{find}(x_1), \ldots, \texttt{find}(x_1), \texttt{union}(x_1, x_2), \texttt{union}(x_1, x_3), \ldots, \texttt{union}(x_1, x_k)$
- ◯ $\texttt{find}(x_k), \texttt{find}(x_k), \ldots, \texttt{find}(x_k), \texttt{union}(x_1, x_2), \texttt{union}(x_1, x_3), \ldots, \texttt{union}(x_1, x_k)$
- ◯ $\texttt{union}(x_1, x_2), \texttt{union}(x_1, x_3), \ldots, \texttt{union}(x_1, x_k), \texttt{find}(x_1), \texttt{find}(x_1), \ldots, \texttt{find}(x_1)$
- ◯ $\texttt{union}(x_1, x_2), \texttt{union}(x_1, x_3), \ldots, \texttt{union}(x_1, x_k), \texttt{find}(x_k), \texttt{find}(x_k), \ldots, \texttt{find}(x_k)$
- ◯ None of the above.

(None of the above.)