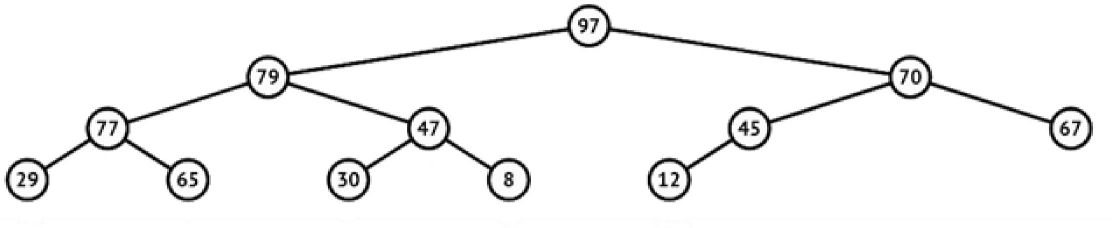
Multiple-Choice Questions

Q1 [4m]

We apply Insert(**82**) to the binary max heap shown below. Which of the following arrays correctly represents the resulting binary max heap? Note that the first element of the array in each option has index 1.



A) 97, **82**, 79, 77, 70, 67, 65, 47, 45, 30, 29, 12, 8

B) 97, 79, 70, 77, 47, 45, 67, 29, 65, 30, 8, 12, **82**

C) 97, **82**, 79, 70, 77, 47, 45, 67, 29, 65, 30, 8, 12

D) 97, 79, **82**, 77, 47, 70, 67, 29, 65, 30, 8, 12, 45

E) 97, 79, **82**, 77, 47, 67, 70, 29, 65, 30, 8, 12, 45

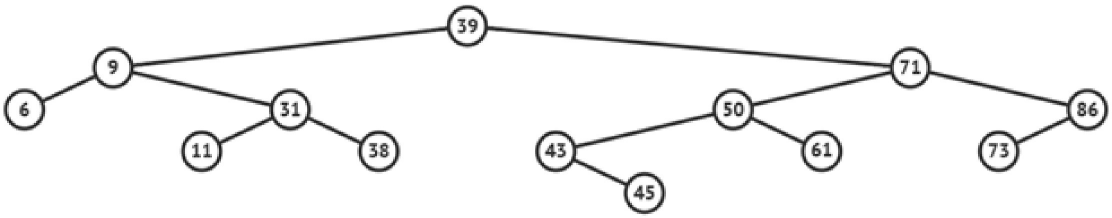
F) None of the other options is the correct answer.

Q2 [4m]

An AVL tree containing N integer keys is shown below. Each node is augmented with subtree size (in the diagram the keys are shown, but subtree sizes are not shown). The AVL tree supports the O(log N) recursive rank operation as described in tutorials.

`rank(key)` returns the 1-based in-order position of a given key. Key 11 has rank 3.

When `rank(61)` is performed, what happens at the recursive `rank(current_node, key)` call where `current_node` is the node with key 50 (if it does reach that node)?



A) Node with key 50 will not be reached.

B) The recursive call at node with key 50 will return 9.

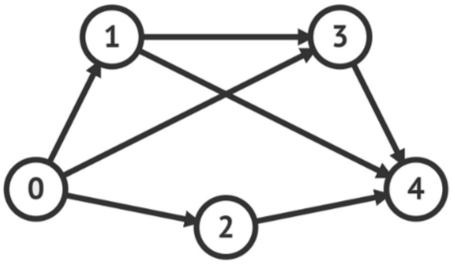
C) The recursive call at node with key 50 will return 10.

D) The recursive call at node with key 50 will recurse on the right child and add 9.

E) The recursive call at node with key 50 will recurse on the right child and add 2+1.

Q3 [4m]

How many topological orderings does the directed graph with 5 vertices shown below have?



A) 0

B) 1

C) 2

D) 3

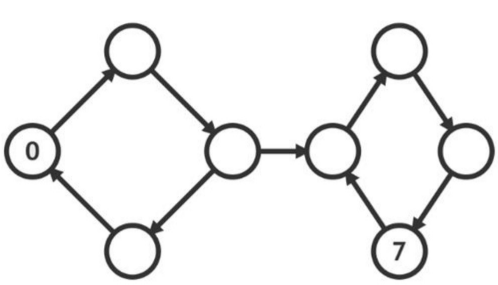
E) 4

F) 5

Q4 [4m]

DFS on a graph can be performed pre-order or post-order. Pre-order DFS "visits" an unvisited vertex before recursing on its neighbours, while post-order DFS holds on to an unvisited vertex and only "visits" it after first recursing on its neighbours. **"Visit" here refers to adding of the vertex to a sequence of vertices**, and NOT to when each vertex is marked as visited.

A directed graph with 8 vertices, numbered 0 to 7 inclusive, is shown below and stored as an adjacency list A. The location of only vertices 0 and 7 are known, while the **order of edges in A are unknown**.



i) The first "visited" vertex in `pre_order_DFS(A, 0)` must be 0.

ii) The last "visited" vertex in `pre_order_DFS(A, 0)` must be 7.

iii) The first "visited" vertex in `post_order_DFS(A, 0)` must be 7.

iv) The last "visited" vertex in `post_order_DFS(A, 0)` must be 0.

Which of the above claims are true?

A) i and ii only.

B) i and iii only.

C) i and iv only.

D) ii and iii only.

E) iii and iv only.

F) All of i, ii, iii and iv.

Q5 [4m]

There is a graph G with V vertices and at least V edges. Which of the following statements about running Dijkstra's algorithm on G is **false**?

A) If there are cycles within G, Original Dijkstra's algorithm may still work.

B) If G is a directed graph, Original Dijkstra's algorithm may still work.

C) Original Dijkstra's algorithm can work on sources where some vertices are unreachable.

D) If ALL edge weights in G are negative, Dijkstra's algorithm and/or G can easily be modified to allow finding of SSSP on G in the same time complexity as the Original Dijkstra's algorithm.

E) If G works properly with the Original Dijkstra's algorithm, the algorithm will not relax each edge more than once.

Q6 [4m]

Here are some operations on some data structure containing N elements:

i) Finding the higher **key** (smallest key that is strictly larger than a given key) in an AVL tree.

ii) Given a **node** in an AVL tree, finding the node with the lower key (largest key that is strictly smaller than the key of the given node).

iii) Removing element with largest key in a fixed-capacity **maximum** binary heap.

iv) Removing element with largest key in a fixed-capacity **minimum** binary heap.

How many of the above operations have **worst-case** that runs in O(log N) time?

A) 0

B) 1

C) 2

D) 3

E) 4

Q7 [4m]

Which of the following about running Kosaraju's algorithm on input graph G is **true**?

A) While running the algorithm, we get a sequence that shows the topological ordering of G.

B) Every vertex in G must have at least one outgoing edge.

C) BFS must be used within the algorithm.

D) After identifying a sequence of vertices to visit in transpose(G), each vertex in the sequence will either have already been visited, or traversing the vertex in transpose(G) covers exactly one SCC in G.

E) If G is a DAG with V vertices, the algorithm runs in O(V log V) time or slower.

Q8 [4m]

Given a UFDS which uses both union-by-rank and path compression (as in lectures), with 2 of the (possibly many) elements being X and Y, which of the following is **false**?

A) If X is a representative, the height of the tree rooted at X could range from 0 to `rank[X]` inclusive.

B) `unionSet` will never link a taller tree (having greater actual tree height) under a shorter tree (having lesser actual tree height).

C) If we want to maintain the size of each set, we can create an array that stores the size of each element, but only maintain those that are a set representative.

D) After `unionSet(X, Y)`, `findSet(X)` will be the same as `findSet(Y)`.

E) After `unionSet(X, Y)`, the level (number of edges upwards to root) of both X and Y will never be more than 2.

Q9 [4m]

You are given 2 AVL trees T and U, with nodes that have parent pointers, each having N elements. In O(1) extra space (extra space includes space taken up by any recursive call), what is the time complexity of the most efficient algorithm that outputs (prints) a list of the 2N elements in non-descending order?

A) O(N)

B) O(N log N)

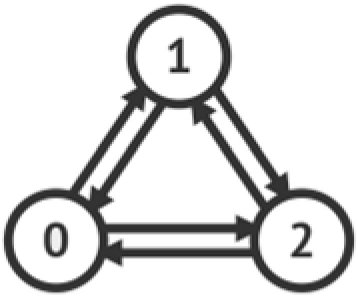
C) $O(N^2)$

D) $O(N^2 \log N)$

E) No such algorithm exists given O(1) extra space

Q10 [4m]

A weighted directed graph G with 6 edges is shown below with weights hidden, along with the **output** of running Floyd-Warshall APSP algorithm on G. There are 6 distinct edge weights in {2, 4, 6, 8, 10, 12}.



fr \to	0	1	2
0	0	8	2
1	4	0	6
2	10	6	0

Each **option is an adjacency matrix.** Which of the following adjacency matrices could G possibly be?

A)

fr \to	0	1	2
0	0	6	2
1	4	0	8
2	10	12	0

B)

fr \to	0	1	2
0	0	8	2
1	4	0	6
2	10	12	0

C)

fr \to	0	1	2
0	0	12	2
1	4	0	6
2	8	10	0

D)

fr \to	0	1	2
0	0	12	2
1	4	0	6
2	10	8	0

E)

fr \to	0	1	2
0	0	12	2
1	4	0	8
2	10	6	0

Analysis Questions

Q11 [6m]

Oizne Mak is trying to implement insertion into a Binary Search Tree (BST) so that the tree can be balanced. He decides to use the implementation below by randomly inserting nodes into the left or right with equal probability (i.e. flipping a coin). Consider his implementation below:

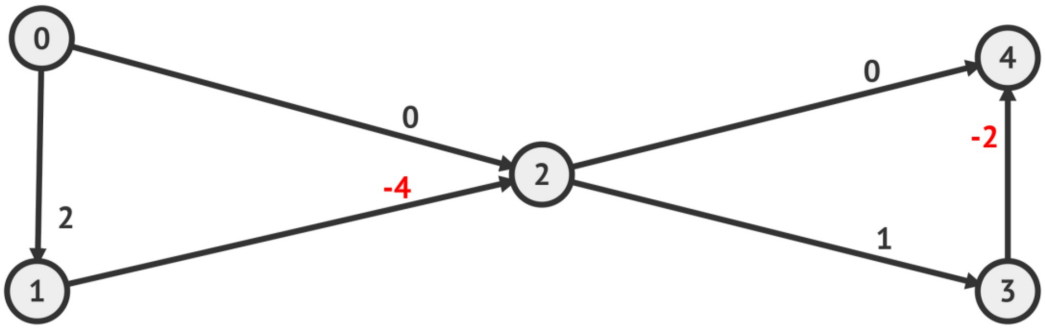
```
Node insert(Node node, int key) {
    double rand = Math.random(); // generates a random number betwen 0.0 and
1.0
    if (node == null) {
        return new Node(key);
    }
    if (rand < 0.5) {
        node.left = insert(node.left, key);
    } else { // i.e. rand >= 0.5
        node.right = insert(node.right, key);
    }
    return node;
}
```

Claim: "After insertion using the implementation of `insert` above, we correctly obtain a BST."

a) Is the claim true or false?

b) Justify your answer.

Q12 [6m]



Consider the following list of 6 edges representing a directed weighted graph of the form (node1, node2, weight) as shown in the diagram above.

- (0, 1, 2)
- (0, 2, 0)
- (1, 2, -4)
- (2, 3, 1)
- (2, 4, 0)
- (3, 4, -2)

We consider one **reduction** in the distance estimate from a source vertex to another vertex to be **one relaxation step**.

Claim: "Running the Bellman-Ford algorithm from vertex 0 (using the edge list sequence shown above) will result in more relaxation steps than the Modified Dijkstra (always add to PQ on distance estimate improvement) from vertex 0."

a) Is the claim true or false?

b) Justify your answer.

Q13 [6m]

"Close one eye"

Tom is using topological order to obtain a sequence of tasks to perform, given N tasks numbered with distinct integers 0 to N-1 inclusive, and M constraints each in the form (u, v), showing that task v can only be started after task u completes.

Tom uses Kahn's algorithm to get an ordering, but he realizes that it is impossible for some tasks to get done because two or more of those tasks are waiting for each other to complete. For example, there could be constraints (a, b), (b, c), (c, d), (b, e), (e, a) and (c, a).

Tom notices that ALL such "stuck" tasks either have 1 or 2 constraints left immediately before it to be cleared. He therefore decides to **loosen the constraints** to allow a task x to be performed if there are just 0, 1 or 2 tasks that must be completed before task x can begin (instead of just 0). Tasks further up the chain of constraints don't contribute to this count. In the earlier example, task a can be performed because only tasks c and e have not yet been completed, but tasks b and d do NOT count towards whether task a can be performed or not.

More formally, the output ordering should NOT have 3 distinct tasks (x,y,z) performed after task w, such that there exists constraints (x, w), (y, w) and (z, w).

He thinks Kahn's algorithm can be modified as such:

- Maintain a visited array besides the in-degree array, where initially all vertices are marked as false i.e. not yet enqueued
- After computing the in-degree of each vertex, mark as true in the visited array (previously there was no such array) those vertices with indegree <= 2 and enqueue them in the usual FIFO Queue (instead of only those with indegree == 0)
- When processing each edge (u, v), first decrease the in-degree of v (as usual), then check if the new in-degree of v $\leq$ 2 AND v is not already "visited" (instead of == 0). If so, then mark v as true in the visited array (previously there was no such array) and enqueue v in the usual FIFO Queue.

Claim: "Tom's modification allows tasks to be performed when they have either no constraints, or constraints requiring only 1 or 2 tasks 'to be completed first'. No task having constraints involving 3 or more 'to be completed first' tasks can be done yet. This algorithm still runs in O(N + M) time."

Choose the best response to the claim:

A) False. A cycle can cause vertices to be stuck in the Queue and hence not be processed.

B) False. A cycle can cause vertices to never be enqueued to the Queue and hence not be processed.

C) False. A cycle can cause infinite loop with this algorithm.

D) False. The algorithm works but runs in better than O(N + M) time, since for each vertex u, there are more neighbours of u being enqueued to the Queue.

E) False. The algorithm works but runs in worse than O(N + M) time as each vertex may be enqueued to the Queue more than once, since indegree can drop to 2 then to 1 and then again to 0.

F) True. Kahn's algorithm will skip vertices with cycles so the claim is still correct even though not all vertices are enqueued to the Queue.

G) True. Loosening the constraints allows vertices with higher in-degree to be selectively enqueued to the Queue, while the visited array ensures that every vertex is enqueued at most once and hence visited at most once.

H) True. Loosening the constraints changes the graph into a DAG, and Kahn's algorithm works on a DAG.

Q14 [6m]

Consider a simple unweighted, undirected graph G with *n* vertices and *m* edges. You can assume that G is connected. We want to obtain a list of edges that we can remove from G such that the resulting graph G' is a tree. Note that this list of edges need not be unique, there could be multiple choices of edges to remove.

Claim: "It is not possible to find a list of edges to remove from G to obtain G' in $O(n + m)$ time."

Choose the best response to the claim:

A) True. We need to run Prim's or Kruskal's algorithm to find a Minimum Spanning Tree (MST) first and remove edges not in the MST, which takes $O(m \log n)$ time.

B) True. We need to run Dijkstra's algorithm to find a Shortest Path Tree (SPT) first and remove edges not in the SPT, which takes $O(m \log n)$ time.

C) True. We need to run cycle detection to find all cycles and back edges in the graph first and remove all back edges, which takes $O(nm)$ time.

D) False. We can run a DFS or BFS to find a Spanning Tree of the graph first and remove edges not in the Spanning Tree, which takes $O(n + m)$ time.

E) False. Using an adjacency matrix representation, we can find the edges to remove in $O(1)$ time.

F) False. We can run DFS or BFS to obtain a 2-colouring of the graph, then remove all edges that connect vertices of the same colour, which takes $O(n + m)$ time.

G) None of the other options are correct.

Open-Ended Questions

Q15 [12m]

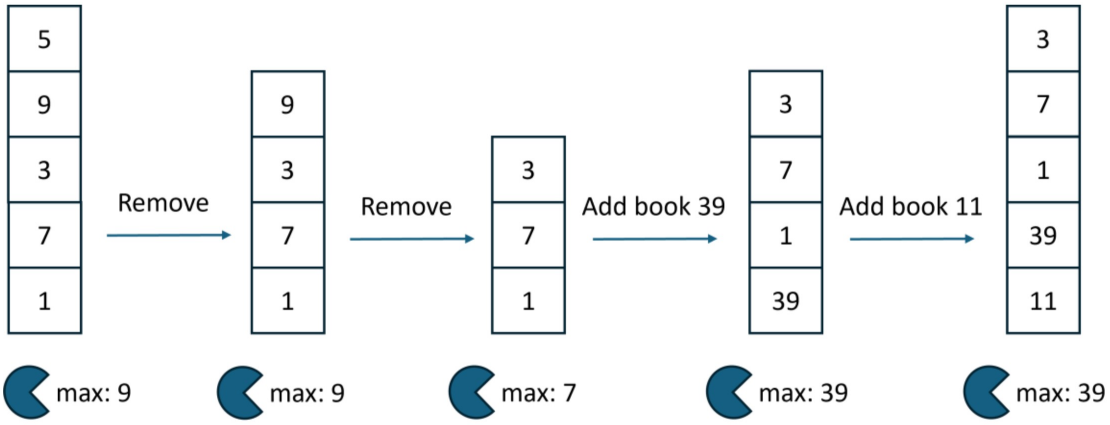
The Great Library of Cossun is due to undergo renovations. Oizne Mak, the librarian, is tasked to move some of the books in the library to the storage room. Each book has a magic number, where the magic number x for each book is a unique integer such that $1 \leq x \leq n$ for integer n. There are a total of n books that need to be moved. Oizne Mak wants to arrange the books into piles so that it will be easier to transport the books. He will add the books one by one to a pile, sometimes also removing books one by one if he realises that he made a mistake, and those books belong somewhere else. However, the playful magician Ecneics strikes again! He plays another prank on Oizne Mak and casts a spell, which now requires Oizne Mak to shout the maximum magic number of ALL the books currently in the pile.

Oizne Mak realises that he can use the magic he learnt last year so that it is easier to arrange the books. He can now lift the entire pile of books with his magic to remove books from the bottom of the pile. Oizne Mak needs to support the following operations on a pile of books:

i. `addBook(i)`: Add a book i to the bottom of the pile.

ii. `removeBook()`: Remove the book at the top of the pile.

iii. `shout()`: Shout the largest magic number of all the books currently in the pile.



Above figure shows a pile of books with magic numbers 1, 7, 3, 9, 5 from bottom to top. Removing book 5 from the top does not change the maximum magic number. Subsequent removal of book 9 reduces the maximum magic number to 7. Then, adding book 39 to the bottom increases the maximum magic number to 39, but subsequently adding book 11 to the bottom does not cause any changes to the maximum magic number.

Design an efficient algorithm and/or data structure so that Oizne Mak is able arrange the books such that you get the **best worst-case OR amortised time complexity**. Remember that the magic number x for each book is a **unique integer**.

To help you in streamlining your ideas, please follow the below steps in your answer:

a) Explain your high-level ideas in 8 lines or less.
b) Initialize your data structures where necessary and/or state any augmentations required (Implementation is not required).
c) Implement the operations in pseudocode or plain English
d) Specify and **explain** the time complexity of the above methods.

Indicate answers to each subsection - each subsection will have marks allocated to them.

Q16 [12m]

Clustering or Cluster Analysis in statistics is a method for grouping similar data points into clusters based on their characteristics, usually measured by some distance function. Data points in the same cluster are more similar to each other than to those in other clusters.

Single-linkage clustering is one method of hierarchical clustering, a type of clustering method. We make use of the fact that clusters that are close to each other can be merged together to form a larger single cluster, based on their distance. We define the distance between two clusters as the minimum distance between any pair of points points across the two clusters.

Consider a dataset with n data points where we are looking for k clusters for integers k, n where $1 \leq k \leq n$. The single-linkage clustering algorithm works as follows:

1. Treat each data point as its own singleton cluster, i.e. there are initially n clusters.
2. Find the two clusters with the smallest distance between them.
3. Merge the two clusters found in Step 2 into a single cluster. (number of clusters reduces by 1)
4. Repeat Steps 2 and 3 until there are only k clusters remaining.

Let x_i represent each datapoint for $1 \leq i \leq n$. Assume that the distances between each pair of data points x_i and x_j are stored in a matrix / 2D-array `A` where `A[i][j]` stores the distance between x_i and x_j. Note that `A[i][j]` = `A[j][i]` and thus `A` is symmetric and `A[i][i] = 0`. You should assume that the distance values can be extremely large numbers.

Consider the following example of the matrix `A` given below. Values in the matrix not relevant to the example have been omitted and marked as `x`. You may assume all such numbers are numbers larger than any of the other distances in the table.

A	1	2	3	4	5
1	0	7	21	x	x
2	7	0	25	x	x
3	21	25	0	41	x
4	x	x	41	0	35
5	x	x	x	35	0

Going through the single linkage clustering algorithm step by step:

- Step 1: Initially, we have 5 clusters: [1], [2], [3], [4], [5].
- Step 2.1: The smallest distance is between clusters [1] and [2] with distance 7 (Points 1 and 2).

- Step 3.1: We merge clusters [1] and [2] to form a new cluster [1, 2]. Now we have 4 clusters: [1, 2], [3], [4], [5].
- Step 2.2: The smallest distance is between clusters [1, 2] and [3] with distance 21 (Points 1 and 3).
- Step 3.2: We merge clusters [1, 2] and [3] to form a new cluster [1, 2, 3]. Now we have 3 clusters: [1, 2, 3], [4], [5].
- Step 2.3: The smallest distance is between clusters [4] and [5] with distance 35 (Points 4 and 5).
- Step 3.3: We merge clusters [4] and [5] to form a new cluster [4, 5]. Now we have 2 clusters: [1, 2, 3], [4, 5].
- Step 2.4: The smallest distance is between clusters [1, 2, 3] and [4, 5] with distance 41 (Points 3 and 4).
- Step 3.4: We merge clusters [1, 2, 3] and [4, 5] to form a new cluster [1, 2, 3, 4, 5]. Now we have 1 cluster: [1, 2, 3, 4, 5].

Notice that at Step 3.2, we did not consider the Points 2 and 3 even though their distance of 25 is lower than the distance between Points 4 and 5 of 35. Since Points 2 and 3 are already in the same cluster [1, 2, 3], we do not need to consider distances between points in the same cluster.

Oizne Mak was reading about single-linkage clustering in the The Great Library of Cossun, when Scitsitats walks in. Scitsitats tells him that single-linkage clustering can be done using ideas from graph algorithms.

Explain how single-linkage clustering to find k clusters from n data points can be done by answering the following questions:

a) What are the graph vertices and edges?

b) What graph representation are you using to represent and store the graph?

c) Explain the algorithm you use including any other data structures you may need.

d) State clearly how you obtain the k clusters or what is used to represent the k clusters

e) State the time and space complexity of your algorithm in terms of n and/or k, whichever is relevant.

Q17 [12m]

A ride-hailing company wants to monitor how long customers wait for a driver in real time across Singapore. Every time a new ride is completed, the system logs that trip's waiting time (in seconds) into a live data stream. To ensure fair surge pricing, the company's algorithm must continuously report the median waiting time of all rides so far:

If we have odd number of price reports, the median would be the middle price. If we have even number of price reports, the median price would be the average of the middle pair price.

Because new trip data arrives thousands of times per second, the system cannot afford to re-sort all recorded waiting times each time.

Instead, engineers design a data structure that can return the median waiting time in O(1) time complexity. The following data structure supports the following methods:

i. `addNum(x)`: insert a new waiting time

ii. `findMedian()`: instantly return the median waiting time

The engineers need your help to support both `findMedian()` in O(1) time complexity and `addNum(x)` in better than linear time complexity.

Note:

- Implementations of `findMedian()` with a linear time complexity or worse will not have any marks awarded.
- Implementations of `addNum(x)` with a linear time complexity or worse will not have any marks awarded.

To help you in streamlining your ideas, please follow the below steps in your answer:

a) Explain your high-level ideas in 5 lines or less
b) Initialize your data structures where necessary and/or state any augmentations required (Implementation is not required)
c) Implement the two methods in pseudocode or plain English
d) Specify the time complexity of the above methods