CS2040 2022/2023 Special Term (Part 2) Final Assessment – Questions

MCQ: 40 Marks, 10 Questions, each question worth 4 marks

1. Suppose you are given a BST (not necessarily balanced) of **n** integers. You are asked to create a min heap containing all the values in the BST. The best algorithm to do so runs in worst case time of:

a. O(1)
b. O(logn)
c. O(n)
d. O(nlogn)

2. You are given a UFDS (using path compression and union-by-rank) consisting of **n** disjoint items. You are asked to form a single set which is of rank 3 (i.e the representative item of the set has rank 3) and height 1 (i.e the representative item of the set has height 1, where height is number of edges from the representative item to some deepest leaf).

   You are only allowed to use the **unionSet(i, j)** operation on items belonging to 2 different sets. The direct use of **findSet(i)** and **isSameSet(i, j)** is prohibited. Determine the smallest possible value of **n** required to fulfill your task.

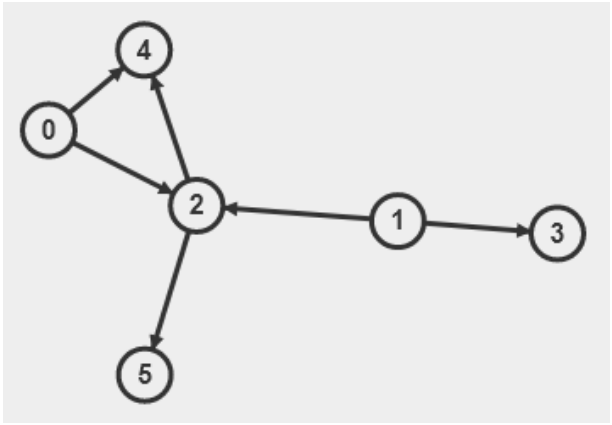
   (Implementation note: for **unionSet(i, j)** the set containing i is moved under the set containing j if ranks are equal)

a. 8
b. 9
c. 10
d. 11

3. You are given an AVL tree where the only information given is that the tree consists of 11 distinct integers. How many of these integers could be the value stored at the root of the tree?

a. 3
b. 4
c. 5
d. 6

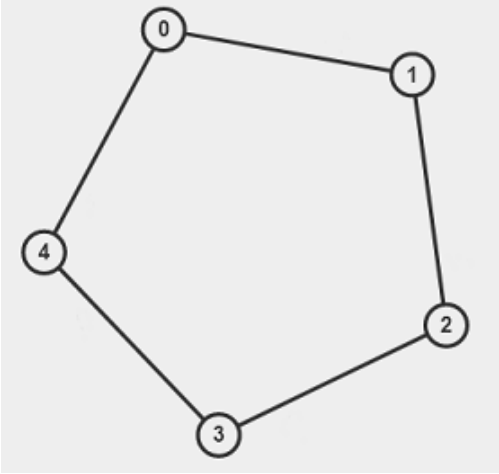
4. You are given the following graph G', which is a subgraph formed from another graph G by removing some edges from G. No vertices were removed.



It is known that (0, 1, 2, 3, 4, 5) is a valid toposort of G. Determine the maximum number of edges that were removed from G to form G'.

a. 8
b. 9
c. 10
d. 11

5. You are given the following graph, consisting of edges of unknown weights:



It is known that if all the edges were of the same weight, there would be a total of 5 valid MSTs for the graph. However, we are more interested in the maximum number of MSTs when the graph consists of 2 distinct edge weights. Suppose that you had to assign each edge a weight of either 1 or 2. At least one edge must have a weight of 1, and at least one edge must have a weight of 2. Under the above constraints, determine the number of edges that should have a weight of 2 in order to achieve the maximum number of valid MSTs for the graph.

a. 1
b. 2
c. 3
d. 4

6. You are given an unsorted array **A** consisting of 15 distinct integers. You are asked to rearrange **A** such that it forms a valid min heap (with the root of the heap at **A**[0]). To do this, you are given a magic function **magicPartition(A, l, r, x)** where:

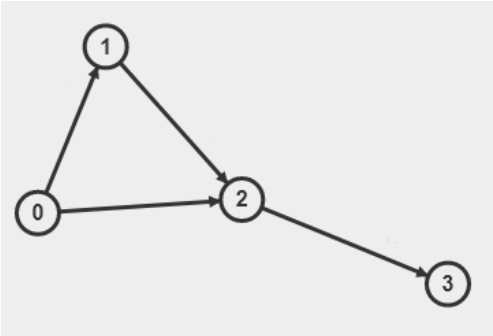
   A is the array
   l and **r** are the left and right indices of a subarray of **A**, so **l** <= **r**
   x is an integer within the range [**l**..**r**]

   The method will take the values currently stored in **A**[**l**] to **A**[**r**] (both ends inclusive) and rearrange them such that from **A**[**l**] to **A**[**r**], all elements to the left of **A**[**x**] are smaller than it, and all elements to the right of **A**[**x**] are larger than it.

   You are only allowed to modify **A** via the use of the **magicPartition(A, l, r, x)** method. Determine the minimum number of calls that needs to be made to **magicPartition(A, l, r, x)** such that **A** is guaranteed to represent a valid min heap.

a. 3
b. 4
c. 7
d. 8

7. You are given the following graph, with edges of unknown weights:



   It is known that:

   1. Each edge has a distinct integer edge weight from [1..4].

   2. D(0,1) < D(0, 2) < D(0, 3), where D(a, b) is the minimum distance from vertex a to vertex b.

Determine the possible edge weight of edge (1, 2). Select all options that apply.

a. 1
b. 2
c. 3
d. 4

8. Given a directed unweighted graph in adjacency matrix form (as given in the lecture notes), the sum of all entries in the adjacency matrix represents:

a. Number of directed edges in the graph
b. Number of vertices in the graph
c. Total distance from vertex 0 to all other vertices combined
d. Nothing in particular

9. You are given the following pseudocode to run on an undirected graph **G**, given in adjacency list form, which is used for Q9 and Q10:

```
int DFS(int vertex, int pre):

    ans = 1
    neighbours = AL.get(vertex)
    for i in neighbours:
        if (i != pre):
            ans += DFS(i, vertex)
    return ans
```

The initial call to this method is **DFS(0, -1)**.

Determine the time complexity of this pseudocode when **G** is a tree with **V** > 5 vertices. You may assume you have infinite memory to run the method.

a. O(V)
b. $O(V^2)$
c. $O(V^V)$
d. The call to DFS(0, -1) will run infinitely

10. Determine the time complexity of the pseudocode (shown in Q9, the previous question) when **G** is a complete graph with **V** > 5 vertices and **DFS(0, -1)** is called.

a. O(V)
b. $O(V^2)$
c. $O(V^V)$
d. The call to DFS(0, -1) will run infinitely

Analysis: 16 marks, 4 questions (4 marks each)

11. Connor wants to implement an ADT with the following operations and time complexity requirements for each operation:

    1. insert(x) - inserts an integer value x into the ADT in amortized or worst case O(1) time.
    2. query(x) - return true if the integer value x is on the ADT, false otherwise. Must be done in worst case O(logn) time or better. (Note that average case O(1) is not better than worst case O(logn)).

    There are 2 more pieces of knowledge that Connor has about the integers to be inserted:
    1. The integers will be inserted from smallest to largest
    2. The largest integer is known

    Connor believes that none of the DSes - {**array, BST (not necessarily balanced), hashtable**} used on their own and not in combination can implement this ADT.

    (Select the most appropriate option)

a. Connor is correct.

b. Connor is incorrect. All the DSes listed can be used to implement his ADT.

c. Connor is incorrect. The array can be used to implement this ADT, since the integers can be inserted to the back of the array (amortized O(1) time) and query can be done by performing binary search (O(logn) time). The other DSes listed cannot implement his ADT in the required time complexity.

d. Connor is incorrect. The array and hashtable can be used to implement this ADT.

e. Connor is incorrect. The BST can be used to implment this ADT since a reference can be maintained to point to the current largest in the BST and when inserting the new node can be inserted as the right child of the current largest abd the reference updated to point to the new node (O(1) time).

Query is simply a search in the BST (O(logn) time).

12. Given a positively weighted, directed graph **G** and a source vertex **s** of **G**, a shortest path spanning tree (SPST) from **s** must include the smallest weighted edge in **G** if the weights of all the edges in G are unique.

    (select all option(s) that is/are part of a correct rationale for whether you think the statement is true or false).

a. True, since the smallest edge must be reachable from **s** and thus must be used in the SPST.
b. True, since the edge weights are unique thus there is no alternative to the smallest edge and so it must be in the SPST.
c. True, because there must be a least costly path from **s** to some other vertex involving the smallest edge (call it (a,b)).
   If we do not use the smallest edge in that path it must be replace with a path from a to b which must be more costly than (a,b) thus there is no better shortest path.
d. False, since the smallest edge might not even be reachable from **s**
e. False, since the smallest edge (call it (a,b)) might not be involved in all possible shortest paths from s to b, thus all possible SPs using s to b as a subpath will not use the edge (a,b) at all, so (a,b) will not be in the SPST at all.
f. False, uniqueness of edge weights is not required to include the smallest edge in the SPST from **s**. Even if there are multiple smallest edges, all of them should be included in the SPST.

Question 13 to 14 refers to the following problem

13. **[2 marks]** Given a sequence of non-repeating integer values to be inserted one by one from the first integer to the last integer into a max heap, the 1-based compact array used to implement the max heap will always contain the same integers at the same position if you reverse the order of insertion (i.e insert the integers from last integer to the first integer in the sequence)
    Note: The insertion algorithm is the one in the lecture notes for binary heaps.

    True/False

14. **[2 marks]** Give your rationale for your answer to the previous question.

Question 15 to 16 refers to the following problem

15. **[2 marks]** Given an unweighted directed graph stored in an Adjacency Matrix A, if we want to reverse the direction of the edges, we can do this without changing A. Simply reverse the meaning of 0 and 1 in the matrix. 0 in an entry A[x][y] will now mean there is an edge x->y and 1 will mean there is no edge from x to y.

    True/False

16. **[2 marks]** Give your rationale for your answer to the previous question.

Structured Questions: 4 questions

This section is worth 44 marks. Answer all questions.

Write in **pseudo-code**.

Any algorithm/data structure/data structure operation not taught in CS2040 must be described, there must be no black boxes.

Partial marks will be awarded for correct answers not meeting the time complexity required.

17. **[10 marks]** District X is the central business district of city Y and people from other districts in Y will go to X for work in the morning and go back home in the evening. These people are so familiar with how to get from their home to X that they will always use the fastest route to get there. You may assume that is only 1 fastest route from each district to district X. In fact all fastest route to get from any district to X is used by someone.

    One day in the middle of the night, there was an earthquake and many of the roads in city Y have been damaged.
    In the morning, you hear from the news that the roads used by people to get from their district to X are the most badly affected since those roads are already under stress from over usage.
    Since you still have to get to work in district X (you are from district Z), you are worried that some roads along your usual route may not be passable. Thus you want to find an alternate route which is the following:

    The fastest route made up of only roads which is not used by anyone to get from their district to X.

    The road network of city Y is represented as a graph of V vertices and E edges where each district is a vertex and and each road (which is 2-way) linking 2 districts are undirected edges linking the respective vertices.
    The weight of each edge is a positive integer value representing the time to travel the edge/road. It is guaranteed that you can always get from any district to any other district in city Y via the road network.

    <u>Come up with the most efficient algorithm to find the best alternative route as given above. If there is no such route return “no such route”. State the time complexity of your algorithm.</u>

    You may assume the graph is stored in an adjacency list **AL**, and you will be given the 2 vertices X and Z (representing district X and Z respectively).

18. **[12 marks]** Give an algorithm to build a binary max heap **H** consisting of positive integers 1 to **N** (1 and **N** inclusive) in worst case **O(N)** time without performing any **shiftDown** operation. If this is not possible, just write "not possible".

19. **[10 marks]** You are given an undirected weighted connected graph **G** with **V** vertices and **V**+**c** number of edges, where **c** is some constant > 1, and where **V** number of the edges are of weight 1 and **c** number of them have weights > 1.
G is stored in an adjacency list **AL**. Give an algorithm to find the cost of the MST of **G** in worst case **O(V)** time.

20. **[12 marks]** The DDEA (Department to discover extraterrestrial activity) has modified their tracking of signals from space.

(Recall that DDEA will record a sequence of positive integers representing signals from space at regular intervals).

The DDEA will first "clean up" each of the recorded signals so that they will be unique from each other.
Also now the DDEA requires you to implement an ADT that consists of the following operations:

1. **insert(int x)**: insert a signal x into the ADT as the latest signal being recorded.

2. **insertAndTrack(int x)**: insert a signal x into the ADT as the latest signal being recorded and also track the largest signal from this point onwards until the next call to **insertAndTrack**. This is called a tracking.

For example if the following signals are inserted with the bolded one being inserted and tracked

1, 2431, 14, 41, **33***, 8178, 221, 13, 11

then at the point when 11 is inserted, the largest signal being tracked will be 8178

Another example, if the following signals are inserted with the bolded one being inserted and tracked

231, **1411***, 13, 131, 111

at the point when 111 is inserted, the largest signal being tracked will be 1411

Note that multiple trackings can be done. An example is as follows

321, 32131, **132***, 3113, 773111, **13119***, 3178111, 1313

when 1313 is inserted, the tracking that starts at 132 will have 773111 as the largest signal being tracked, while the tracking that starts at 13119 will have 3178111 as the largest signal being tracked. This is because the 1st tracking starting at 132 ended at 773111 and thus will not include 3178111.

3. **NumSmalllerThanLatestTracking()**: return the number of trackings with the largest signal less than the largest signal of the latest tracking. If there is no tracking return -1

For examples:

1, 2431, 14, 41, **33***, 8178, 221, 13, 11

There is only 1 tracking with largest signal tracked being 8178, so there is no other tracking thus 0 is returned.

321, 32131, **132***, 3113, 773111, **13119***, 3178111, 1313

There are 2 trackings one with largest signal of the 1st tracking being 773111 and the other (the latest tracking) with largest signal being 3178111. Since the largest signal of the latest tracking is larger than the largest signal of 1 other tracking so 1 is returned.

321, **321312***, 132, **133***, 3113, 77311, 45, **222***, 98982

There are 3 trackings, the first one with largest signal being 321312, the second one with largest signal being 77311, and the third (latest tracking) with largest signal being 98982. Thus 1 is returned since 98982 is only bigger than 77311 and not 321312.

4. **delete()**: This will delete the latest recorded signal. If there is a tracking associated with the signal (i.e the signal was inserted using **insertAndTrack()**) remove the tracking too.

Implement each of the above operations so that they will run in worst case **O(logn)** time, where **n** is the current number of signals recorded.