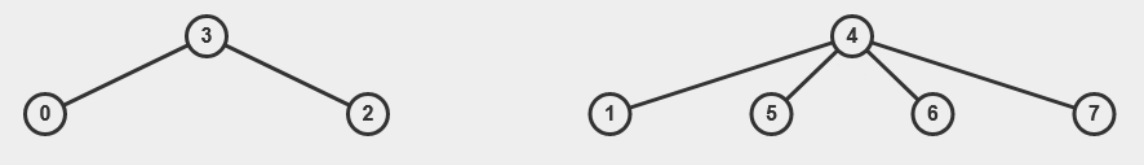
CS2040 2021/2022 Semester 2 Final

MCQ

This section has 10 questions and is worth 30 marks. 3 marks per question.

Do all questions in this section.

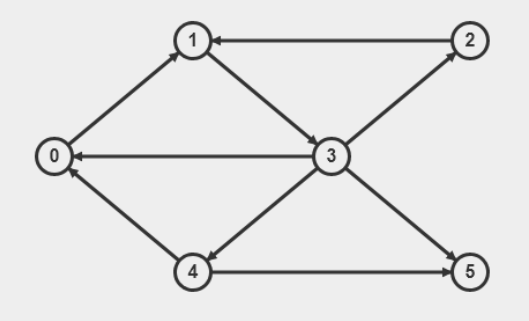
1. You are given the following UFDS, which uses union by rank and path compression:



Initially, the UFDS consists of 8 disjoint sets of 1 item each, before a series of unionSet/findSet calls were made to form it into the UFDS above. What is the rank of the set containing item 6?

a. 1
b. 2
c. 5
d. The UFDS is invalid, or there is more than 1 possible rank for the set containing item 6

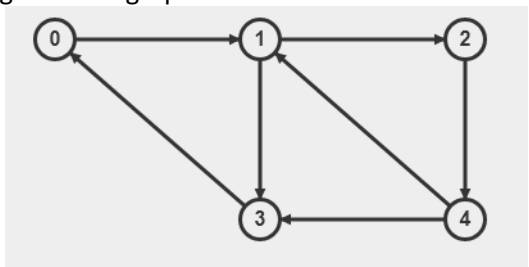
2. You are given the following directed graph:



You are asked to pick exactly one edge from the answers below to be removed from the graph, such that there will not be any cycles in the graph after this edge is removed. Which of the following edges should you remove?

a. 0 -> 1
b. 1 -> 3
c. 3 -> 4
d. 4 -> 0

3. You are given the following directed graph:



The graph is intended to be weighted, but the actual weights of each edge is unknown. However, the following is known:

1. The correct minimum distances from vertex 0 to all other vertices are as follows:

Vertex	0	1	2	3	4
Distance	0	5	3	6	2

2. All edge weights are distinct integers in the range [-2..5] (ie. from -2 to 5, both ends inclusive)
3. No negative cycles are present in the graph.

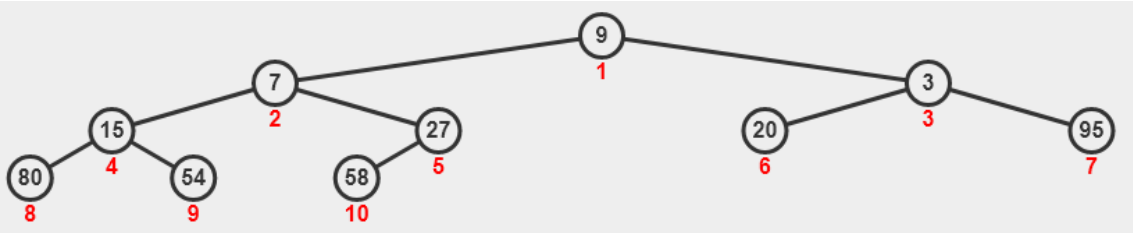
Determine the largest possible weight that the edge 1 -> 3 can have.

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

4. Given a BST (which may not be balanced) with **n** elements, we want to output all elements in the BST in sorted order. This can be done in worst case:

a. O(n) time
b. O(n log n) time
c. $O(n^2)$ time
d. $O(n^2 \log n)$ time

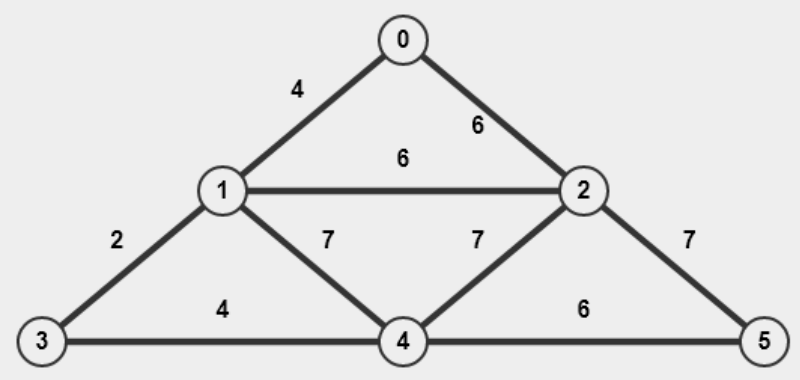
5. You are attempting to run an O(n) createHeap() to create a minimum binary heap. The following is how the heap looks before any shiftDown() operations are called.



How many swaps will occur after all shiftDown() operations have been called?

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

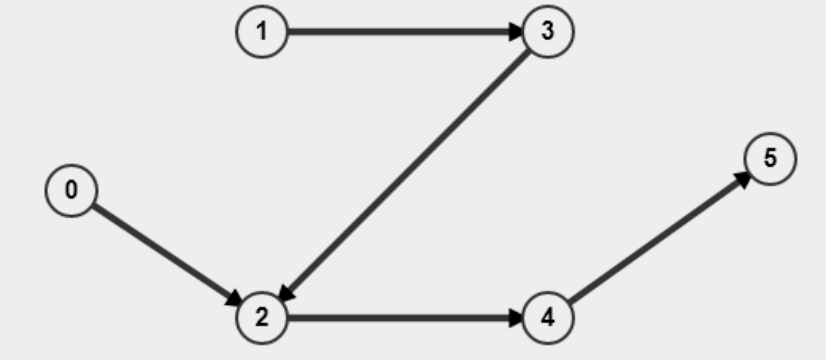
6. How many distinct Minimum Spanning Trees are there in the following graph?



(there are no edges with weights 5 (five) or 8 (eight) in this graph; if you are seeing this, it is most likely an edge with weight 6 (six)).

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

7. You are given the following directed acyclic graph:



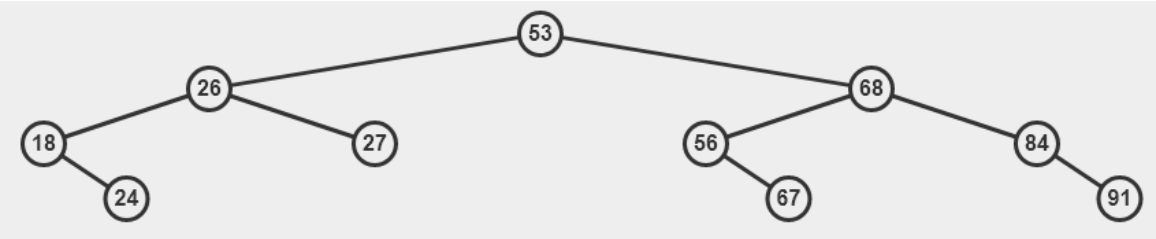
You are asked to pick <u>exactly one</u> edge from the answers below to be inserted into the graph. After this edge is inserted, it should result in the <u>fewest</u> number of valid toposorts. Which of the following edges should you insert? A cyclic graph is considered to have 0 valid toposorts for this question.

a. 3 -> 4
b. 3 -> 5
c. 0 -> 3
d. 1 -> 2

8. Given both a min binary heap and a max binary heap containing the same elements, we can obtain a sorted list of these elements in worst case:

a. O(1) time
b. O(log n) time
c. O(n) time
d. O(n log n) time

9. You are given the following AVL tree:



How many of these elements would, if deleted (lecture implementation of delete) on its own, result in <u>at least</u> one rotation?

a. 0
b. 1
c. 2
d. 3

10. Which of the following <u>may</u> give incorrect SSSP results on a directed graph (having no bi-directed edges) with negative edge weights but no negative cycles?

a. Modified Dijkstra's
b. Bellman Ford's
c. Floyd Warshall's
d. Depth First Search

Analysis

This section has 4 questions and is worth 16 marks. 4 marks per question.

<u>Please select True or False and then type in your reasons for your answer.</u>

Correct answer (true/false) is worth 2 marks.

Correct explanation is worth 2 marks. Partially correct explanation worth 1 marks.

Do all questions in this section.

11. In an undirected connected graph where there exist at least 1 cycle with multiple edges having the same largest edge weight in the cycle, then there always exist multiple MSTs for the graph.

12. For an undirected connected graph with no cycles (standard definition of cycle as given in lecture notes) but where all the edge weights are negative, running modified Dijkstra (as described in the lecture notes) on such a graph (stored in adjacency list) from a source vertex **s** will give the correct SSSP solution from **s** to every other vertex.

13. We can always re-construct a BST (exact same structure) given the in-order sequence of the keys in the BST and which key is the root in the BST.

14. Given an unsorted array of N non-repeated integers and two integers a and b from the array, where **a** < **b** and there is at least 1 integer in the array bigger than **a** and smaller than **b**, it is possible to output the M^{th} smallest integer between **a** and **b** (excluding **a** and **b**), where $1 \leq M \leq K$ (K is the number of integers in the array bigger than **a** and smaller than **b**) in <u>worst case</u> O(N+MlogK) time.

Application Questions

This section has 5 questions and is worth 54 marks.

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.

Note:

1. If you are using the UFDS data structure, you may assume that all operations take worst case O(1) time.

2. Hash table operations are taken as O(1) worst case

3. If you want to solve the question as a graph problem, give the graph modeling first (if the required graph has not been described in the question). That is, state what are the vertices, what are the edges and how are the vertices linked by the edges.

15. John who lives in city X is planning to visit his friend Mary who lives in city Y by driving there from city X.

    Every time John has to enter a city on his way to visit Mary, he will be stopped and will be deducted half of the amount of money that is left in his "toll card" before he is allowed entry. Once the amount of money is < 1 dollar he will be rejected entry into the city. Note that since John is already from city X at the start, money will not be deducted there.

    Given a graph G of N vertices and E edges representing the road network (vertices = cities and edges = bi-directional roads linking the cities) stored in an adjacency list, give the most efficient algorithm in terms of worst case time complexity you can think to determine the minimum amount of money that John needs to have in his "toll card" in order to drive from city X to city Y to visit Mary. (Announced during exam: You may assume there is always a way to get from X to Y)
    [8 marks]

16. Mankind in the year 4XXX has begun to colonize other planets outside the solar system. One such planet is planet Y.

    Due to a recent gigantic planetquake, most of the cities were ruined. In order to rebuild, one urgent matter would be to rebuild the road network linking the cities which have also been destroyed. Since the central database containing the road network was destroyed too, the last resort was to refer to ancient physical blueprints of a proposed network of roads linking the cities.

    In the blueprints, there are N cities (labeled from 0 to N-1) and M proposed roads ($N-1 \leq M \leq \frac{N(N-1)}{2}$) linking the cities such that there is always a way to get between any pair of cities via the road network (announced during exam: the roads are two way roads). Due to the way the cities are built, the proposed roads in the network are all the same length**.** Thus building each road would normally cost the same amount C (for some C > 1,000). However, because different roads go through 3 different types of terrains, the cost of the roads are multiplied by a factor of 1, 2 or 3, so cost of the roads can be C, 2C or 3C.

    Given a list A of the M proposed roads where each road is expressed as a triple (x,y,z) where x and y are integers representing the 2 cities connected by the road and z is the cost of building the road, use the most appropriate DS(es)/ADT(s) and give the most efficient algorithm <u>in terms of worst case time complexity</u> you can think of to determine the set of roads to be actually built such that <u>there is still a way to get between any pair of cities and the cost of building all the roads is</u> **<u>minimum</u>**.
    [12 marks]

17. In the future there is a popular game played among N+1 robots that are placed at integer coordinates (x,y) (x,y ≥0) around a large field. The distance between 2 robots at (x,y) and (x',y') can be calculated as $\sqrt{(x-x')^2+(y-y')^2}$. Each of these robots will hold a ball, called the owned ball at the beginning of the game.

    To play the game, there will be 1 selected robot and the other N robots will each need to pass the owned ball it is holding at the beginning of the game to the selected robot in turns (starting from robot 1 and going to robot N+1, excluding the picked robot itself). A robot can start to pass its owned ball only after the robot before it has finished passing its owned ball to the selected robot or the robot before it has forfeited its turn (more on this in the description below).

    Each robot can throw a ball it is holding to another robot who will catch it, so the catcher can have up to 2 balls, the caught ball and the catcher's owned ball if that has not already been passed to the selected robot. However, each robot can only throw to some other robot that is within a radius D of itself. To successfully get the owned ball to the selected robot, each robot needs to plan a path for the owned ball to be passed among the other robots until it reaches the selected robot in the shortest time possible (the longer the distance traveled by the ball the longer the time it takes).

    If there is no way for a robot to pass the owned ball to the selected robot, the robot will forfeit its turn. All robots that can pass their owned ball to the selected robot are called valid robots. You may assume there is 0 lag time between a valid robot having passed its owned ball to the selected robot and the next valid robot starting to pass its owned ball. In addition, the robots are always placed so that **no more than maximum(100, N)** valid robots can pass their owned ball to any selected robot.

    Given an array A containing the coordinates of the N+1 robots, use the most appropriate DS(es)/ADT(s) and give the most efficient algorithm in terms of worst case time complexity you can think of to determine the selected robot such that it satisfies the following 2 conditions (condition 1 is more important than condition 2)

    1. The most number of valid robots can pass their owned ball to the selected robot
    2. Sum of the distance taken to pass the owned balls of all valid robots to the selected robot is minimized.

    As an example if there are N+1=6 robots placed at coordinates (0,0), (1,1), (5,0), (5,1), (6,2) and (7,0) respectively and the distance every robot can throw is D = 2, then the robot at (5,0) or (5,1) should be the selected robot, with only (0,0) and (1,1) not being able to pass their owned ball to the selected robot, and the sum of the time taken to by the valid robots to pass their owned ball to the selected robot is minimized.
    [12 marks]

18. In the kingdom of Wadanka with a population of N people, each person is given a unique integer id at birth. This id is useful in the following way.

The social ranking of each person in the kingdom is determined by their id. The higher the id the higher the social ranking. The person with social ranking = 1 also has the largest id and the person with social ranking = N also has the smallest id. (Note: social ranking = 1 is the highest social ranking)

Wadanka is made up of many tribes. The leader of the tribe is the person with the highest social ranking and therefore the largest id in the tribe. Whenever 2 tribes in Wadanka encounter each other, they "fight" by comparing the total sum of id of all persons in their tribe. The tribe with the larger total sum will be the victor. If both tribes have the same sum then the leaders will compare their id and the one with the larger id will be the winner (note that every person's id is unique thus there will not be a tie here). The following will then happen

1. Let X be the leader of the winning tribe and Y the leader of the losing tribe.
2. If Y is the person with the current highest social ranking in Wadanka then X will have the id of Y added to his/her id.(e.g if X has id = 10 and Y has id = 15 and is currently the person with the highest social ranking, then new id of X is 25)
3. If Y is not the person with the current highest social ranking, then id of X and Y will be exchanged if id of X is smaller than id of Y. Otherwise nothing is done.
4. The 2 tribes will be merged, and the new leader is the person with the largest id in the new tribe.

Given that there are N starting tribes in Wadanka with 1 person each (the leader him/herself) and the id of these N persons are given in an array A, use the most appropriate DS(es)/ADT(s) and algorithms you can think of to implement the following 3 operations in the required time complexity:

1. **FightAndMerge(a,b)** -> the tribe that person with id = **a** belongs to and the tribe that person with id = **b** belongs to will fight to determine the victor (according to the rules stated above) and the 2 tribes will be merged. If both persons belong to the same tribe then nothing happens. This should run in **worst case ≤ O(logN) time**.

2. **Leader(a)** -> return the id of the leader of the tribe that person with id = **a** belongs to. This should run in **worst case < O(logN) time**.

3. **SocialRanking(a)** -> return the social ranking of person with id = **a**. This should run in **worst case ≤ O(logN) time**.

You may assume that the input arguments given to the 3 operations are always valid, and you can have a pre-processing step that takes **no more than O(NlogN)** time to populate the DS(es)/ADT(s) you use.
[12 marks]

19. You are a treasure hunter and have found yourself trapped in the pyramid of king Tat while trying to discover treasure within!

The pyramid is a maze that is split into N (N > 10) rooms that are labeled from 0 to N-1. There are M (N-1 ≤ M ≤ $\frac{N(N-1)}{2}$) doors which will allow you to pass from one room into another room. If a room is linked to another room, they can only be linked by at most 1 door.

At the start all the M doors are shut. As you approach a shut door in a room, it will automatically open, and you can go into the next room. You can revisit rooms you have visited before, but you can only go through at most K (0 < K ≤ 5N) rooms (including repeated rooms and the room you start in) before all doors in the pyramid will shut trapping you permanently! Luckily, you have a map of the entire pyramid and so know how the N rooms are linked by the M doors.

From the map, you have created an array R that contains M descriptions, each of which is a pair (x,y) meaning that room x is linked to room y by a door.

You start off at room A and you need to get to a room B which has the only door that opens to the outside, thus allowing you to escape, and you are sure there is a way to reach B from A before you become trapped. Once you reach room B, all the other doors in the pyramid will immediately shut permanently and the only door exiting the pyramid will open.

Escaping the pyramid is not the problem, rather you are faced with a dilemma. Starting at room A, you will have T_A years added to your lifespan (which starts at 80 years). If you enter room i next, you will have T_i years deducted from your lifespan. If you enter room j after room i, you will now have T_j years added to your lifespan again. Thus, this addition and deduction will keep alternating as you move from room to room (you will always start off with addition of lifespan then alternate). In fact, you can even reach -ve lifespan! But the effect will kick in only upon you exiting the pyramid.

Again you have created an array T of size N where T[i] will contain the amount of years to be added/deducted for room i (depending on when you visited it).

Given array R and T, model the problem as a graph (clearly state what are the vertices, what are the edges and how the vertices are linked by edges) and give the most efficient algorithm you can think of to start off at room A and reach room B with the most amount of lifespan possible. If you end up with -ve lifespan output "impossible to escape", otherwise output the lifespan.

As an example, if there are N = 6 rooms and the M = 6 doors linking the rooms are as follows (0,1), (1,2), (1,3), (3,4), (2,5) and (5,1) and the lifespan added/deducted for each room is as follows room 0 = 2, room 1 = 1, room 2 = 5, room 3 = 3 and room 4 = 4 and room 5 = 5. Given A = room 0, B = room 4 and K = 9, the best path to take is 0->1->2->1->5->1->3->4 with total lifespan = 80 + 2 + (-1) + 5 + (-1) + 5 + (-1) + 3 + (-4) = 88.
[10 marks]