ExamSoft

StudentID: e0958287

Strengths and Improvement Opportunities

CS2040 AY2324 Sem1 Final Assessment

Course: 2310 CS2040 • Instructor: Mr - NUS CIT • 12/01/2023 • Questions: 21
StdDev = 14.94 • Mean = 51.22 • Median = 51 • Percentile Rank = 31.3692

44.00%	51.22%
My Score (44/100)*	Average Score (51.22/100)*

*All points for essays have not been assigned. This may not be the final score.

QUESTION | **POINTS**

● CORRECT 🟥 INCORRECT ⬢ PARTIAL CREDIT

1 **MCQ1**
[4 Marks] You are attempting to run an O(N) createHeap() to create a max 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? — **4/4**

X A
X B
X C
> D

2 **MCQ2**
[4 Marks] Given an array of N integers representing a min heap. What is the worst-case time complexity to convert it to a max heap? — **4/4**

X A
X B
> C
X D

3 **MCQ3**
[4 Marks] You are given the graph below. How many distinct minimum spanning trees are there in the graph? — **0/4**

X A
> B
X C
X D

4 **MCQ4**
[4 Marks] You are given a complete undirected graph G on 4 vertices, having 6 edges with weights being 1, 2, 3, 4, 5 and 6. What is the maximum weight that a minimum spanning tree of G can have? — **0/4**

X A
> B
X C
X D

5 **MCQ5**
Questions 5 and 6 refer to the following graph: Consider an undirected, complete graph with 6 vertices. It is known that every edge has weight either 1 or 2. Given 2 vertices S and D: [4 Marks] How many simple paths are there from S and D? — **0/4**

X A
X B
> C
X D

6 **MCQ6**
[4 Marks] What is the maximum number of shortest paths between S and D possible? — **0/4**

X A
X B
> C
X D

7	**MCQ7** **Questions 7 and 8 refer to the following graph: Consider a connected, weighted, undirected graph G with no negative edge weights. [4 Marks] Given a special vertex X in G, we want to answer Q queries of the form f(A,B) where f(A,B) denotes the shortest path from any vertex A to any vertex B that contains vertex X (there be cycles in such a SP). What is the most efficient and correct algorithm to answer all Q queries among those listed? Vertex X remains the same for all the queries. Note: Some details of each algorithm in the options may be missing, but based on what is given, decide which is the most efficient and correct.** X A > B X C X D	**0/4**
8	**MCQ8** **[4 Marks] Instead of being given a special node X, you are now given a special edge Y. We want to answer Q queries of form g(A,B) where g(A,B) denotes the shortest path from any vertex A to any vertex B that passes through edge Y (there may be cycles in such a SP). What is the best possible worst case time complexity to answer all Q queries including the time required for any pre-processing among those listed? Y will remain the same for all queries.** X A > B X C X D	**4/4**
9	**MCQ9** **[4 Marks] You want to create an AVL tree by adding the elements 1, 2, 3, 4, 5, 6, 7, 8, 9 into the tree one at a time, in any order except that: The first element added is 6 The second element added is 3 NO rotation / rebalancing operation must occur at any time How many different AVL trees of 9 elements can be produced at the end? Choose the interval that contains the correct answer** X A X B > C X D	**0/4**
10	**MCQ10** **[4 Marks] There is a UFDS of N items, with both union-by-rank heuristic and path compression, each item being its own disjoint set initially. Ivan then performs some (zero, one or more) number of findSet, isSameSet and/or unionSet operations. You pick an item from the UFDS and examine the characteristics of that item. Which of the following options is true?** X A X B X C X D > E	**4/4**
11	**ANA11** **[4 Marks] Claim: Given a general directed graph G with V vertices, running Bellman ford algorithm on a source vertex s of G for only k (k < V-1) iterations of the outermost for loop (k passes of relaxation) will only find the shortest paths from s to vertices where the shortest paths are up to k edges long. For vertices with shortest paths from s having more than k edges, no valid path can be found yet. Select the option that has the correct answer to the above claim and has the best explanation.** X A X B > C X D X E	**0/4**

12 **ANA12**
[4 marks] It is known that DFS generally cannot be used to find shortest distance from vertex x to y in an unweighted graph. Now you are given a directed unweighted acyclic graph G with V vertices and E edges, along with valid vertex numbers x and y. There is a modified recursive DFS algorithm, in which the algorithm does NOT store, mark, or check for visited vertices. When exploring each neighbouring vertex, the distance estimate is 1 greater than the distance estimate from x to the current vertex. Recursively visit a neighbouring vertex if the distance estimate is improved over the previous estimate. Claim: This modified recursive DFS algorithm called on vertex x can successfully compute the shortest distance to vertex y   Psuedo-code of the modified DFS matching the description above: modifiedDFS(visited, distances, adjList, currVertex) {   for neighbour in adjList.get (currVertex):   if 1 + distances[currVertex] < distances[neighbour] then   distances[neighbour] = 1 + distances[currVertex]   modifiedDFS(visited, distances, adjList, neighbour) } SSSPOnDAG(adjList, x, y) {   V = adjList.size()   Create visited[V], init all to false   Create distances[V], init all to +INFINITY except 0 for x   modifiedDFS(visited, distances, adjList, x)   return distances[y] } Select the option that has the correct answer to the claim, and has the best explanation.

X A
X B
X C
X D
> E
X F
X G
X H

0/4

13 **ANA13**
Questions 13 and 14 refer to the problem below: Alice and Bob are working at a company in the human resources department and they keep a record for all their employees. Every employee's performance is evaluated once a year and every employee receives a score. The 5 top-scoring employees are recognized and receive a small gift every year at the company's year-end party. Alice and Bob use a max priority queue with all the employee scores so that they easily can determine the 5 top scorers. Recently the company has realized that some of the lowest scores may be due to external factors, such as family situations, etc. Now Alice and Bob's boss has asked them to report to him the 5 lowest scorers in the company. Alice is suggesting to Bob to make a copy of the max priority queue (a max heap), dequeue each score from the max heap and for each dequeued score enqueue it into a min priority queue (a min heap). Then the 5 lowest scorers can easily be extracted. However, Bob thinks this is too slow. He claims that all that is needed is to copy the entire 1-base compact array storing the max heap in reverse into another 1-based compact array. The new 1-based compact array will now become a min heap. This will only take O(N) time.   [2 Marks] Bob's claim is correct.

X T
> F

2/2

14 **ANA14**
[2 Marks] Give your rationale for your answer to the previous question.

A: Bob cannot simply reverse the array representation of the max heap and store in another array for a min heap. Bob will not attain a min heap this way. Consider this case [10, 9 , 8 , 3 ,2 ,1 ,4] By Bob Logic, simply by reversing we get [4 , 1 ,2 ,3 ,8 ,9 ,10] This is not a min heap because the root 4 is bigger than 1, hence there is no min heap in the first place However, if Bob add "Do some swaps to adjust the heap property accordingly after reversing", it will makes sense to be O(n) [since swap is dependent on the height of the tree O(h) and that reverse is O(N) , dominant : O(N)] However, right now his claim is false.

Grader: Faculty
Points: 2/2
Comments: Correct answer

2/2

15 **ANA15**
Questions 15 to 16 refer to the following problem: Given an AVL tree T that contains N numbers and an array A of length N, Pepega wants to fill A with the elements from T such that the numbers in A will be in ascending order. Pepega remembers that removing an element from an AVL tree takes O(log N) time, and repeated removal of the minimum from T to fill A will thus take O(N log N) time.   [2 Marks] Claim: Filling A with all elements in T such that A is now in ascending order takes worst case O(N log N) time.

X T
> F

2/2

16 **ANA16**
[2 Marks] Give your rationale for your answer to the previous question.

A: The min removal for AVL tree is NLogNConsidering we are using an array (which is not an ArrayList), to append the values to the back of the array will require some form of single loop which thus entails O(N) Considering that these two executions are done together, this would mean the complexity would be O(N^2LogN) instead

Grader: Faculty
Points: 0/2
Comments:

0/2

17

APP17

[6 Marks] Given the root of a not necessarily balanced binary search tree (not augmented with the size and height information, just the standard BST vertex) containing n integer values and k (1 <= k <= n) as input, find k-th largest element in the BST i.e., find the value of the (n – k+1)-th smallest element. Assumption: You are given the BST, but you are not given n, i.e., you don't know how many elements are in the tree. For example, in the following BST, if k = 3, then the output should be 20, and if k = 7, the output should be 6. Select the worst case time complexity of the most efficient algorithm from the options listed.

X A
X B
> C
X D

6/6

18

APP18

[6 Marks] You are given a weighted directed graph G modeling a chemical plant. The vertices represent the state of a chemical reaction, and directed edges going from 1 vertex to another vertex represent the transition of the state of the chemical reaction to another state. In order to transit from one state to another, the chemical reaction will give off some positive amount of energy. This amount is the weight of the edge representing the transition. You can assume this amount is an integer value > 0. The maximum amount of energy given off by any transition in the graph is at most an integer value M and at least an integer value M'. The total energy given off is the sum of the edge weights of the shortest path from a starting state S to an ending state S' in the graph. The chemical plant cannot control the chemical reaction if the total energy given off is too much, so one way is to supply reagents into the chemical reaction at S and this will cause the energy given off by all transitions to be lowered. If K amounts of reagents are introduced, it will cause the energy given off to be reduced by exactly K. You can assume K is an integer value and (1 <= K <= M'). For example if there is a path from state 1 to state 4 as follows: 1->2->3->4, if K amounts of reagents are used, then the weights of all the edges/transitions 1->2,2->3 and 3->4 respectively will be reduced by K. Given the graph G, with V vertices and E edges, the values M and M', a positive integer value T, a starting state/vertex S and an ending state/vertex S', you want to answer the query of what is the least amount of reagent to use so that the total energy given off is <= T. If the amount of reagent required is > M', the answer to the query should be "impossible". Note that the chemical reactions cannot transition from one state x to another state y and then go through 1 or more transitions and come back to state x , as this means there is the ability to perpetually give off energy which breaks the laws of physics. Select the worst case time complexity of the most efficient algorithm (using what you have learned in CS2040) to answer the above query.

X A
X B
X C
X D
X E
> F

6/6

19

APP19

[11 Marks] In a manufacturing project, there are T tasks and R pre-requisite rules, 0 ≤ R ≤ T0.5. Assume the tasks are numbered 0, 1, 2, … (T-1) for simplicity. You are given as input – not yet read into any data structure – the value of T, as well as R lines each containing 2 integers xy for 0 ≤ x, y < T and x ≠ y showing that task x can only be started after task y completes. Find if there is a case (just answer INVALID or VALID) where the rules are problematic, i.e. when the rules require some task k to be completed before task k may be started. Remember, for this problem, O(R) is much better than O(T + R) If you solve the problem in O(R) average time or better, you will get the full 11 marks If you solve the problem in O(R log R) time, you will get 10 mark If you solve the problem in O (T + R) time, you will get 9 marks [Advice: If you are not confident of getting the top 2 time complexity bands, aim to get the 3rd band CORRECT] If you are answering using a description, be very clear to explain how to model the problem, any data structure(s) and algorithm(s) you require. If you do make changes to any design or algorithm shown in lecture, you have to clearly explain in detail what the changes are. Give only one answer, do not give alternatives, or the lowest band marks will be awarded

A: 1. Create a class "Task" which contains the following attributes - Task Id and the adjacent Task Object (The task in which x can only be started after task y completes). If there isnt such an adjacent task, just = null . All Task Object will be considered to Completed = False. Write in your getter methods. 2. In a for loop, read in the input and store them in a list . The list will contain T tasks (these will be vertices). 3. Loop the task list to find the task which does not have any adjacent task. That will be our source vertex X 4. Mark the source vertex as Completed = True, search the loop for the next vertex that has X as our adjacent task. Print out Valid Case 5.Repeat this loop until all vertices are marked as True. If there is one which is problematic, print out "Invalid"Maybe i should reverse the list and do some form of toposort algo but i dun have any time so my logic is somewhat there

Grader: Faculty
Points: 6/11
Comments: You have designed a solution with an efficiency that is unclear. You did find the main idea: the rules are INVALID if there is a directed cycle, but VALID otherwise (i.e., no cycle). However, your solution does not describe which specific, efficient data structures should be used. This makes a difference in the efficiency.

6/11

20

APP20

[11 Marks] The valley of crystals is a strange place with M crystal formations of different colors growing on the ground. Each crystal formation is given a coordinates (r,c) where r and c are positive integer values. Energy bridges can be created between a pair of crystal formations to allow travel between the two, if the manhattan distance between the 2 is at most d.The manhattan distance between a pair of crystal formations at coordinates (r1,c1) and (r2,c2) is |r1-r2|+|c1-c2|. For any crystal formation in the valley, there is always another crystal formation within a manhattan distance of d. It is cheaper to create energy bridges between pairs of crystal formations of the same color due to resonance that allow minimal energy to be used. On the other hand it is more expensive to create energy bridges between pairs of crystal formations of different colors. You are given an array A of size M containing triples (r,c,k) describing the location (r,c) and color k (k is an integer >= 1) of each crystal formation in the valley, the value of d, and an integer value t. Formulate this as a graph problem and give the most efficient algorithm you can think of to answer the following query: Determine if at most t bridges that connect pairs of crystal formations of different color need to be built so that a person can get from any crystal formation to any other crystal formation in the valley. If this is possible return true else return false.

4/11

A: We can use coloring over here Each crystal is considered to be a vertice and the energy bridges are considered to be the edge. This graph assumes a bi-directional, weighted graph We set up a graph , where the edge of between two vertices of different colours will have much higher energy (Higher weight : say 5), while the edge between two vertices of the same colour have the weight of 1. Each vertice will consist of a number to represent the location (r,c) Asumming that the question wants to create th cheapest energy bridge, and that theree is no negative wegiht between two vertices, we can use original DijStra Algorithm to find the shortest path in the graph. 4. In cases of cycles, we would need to use Bellman ford to instead

Grader: Faculty
Points: 4/11
Comments: filter by distance?

not an SP problem. need to determine if <= t edges of diff colours need to be included in the entire graph, NOT whether any path length is <= t

btw dijkstra can work in cyclic graphs (still given the precondition of no -ve edge weight at all), you don't need to switch to bellman ford
bellman ford without modification can't save you also if there are -ve weighted cycles

2 0 2 0 0

21

APP21

[10 Marks] We are back to playing dominos from THA4B. You are directly given the layout of N domino tiles (numbered from 0 to N-1) stored in an adjacency list AL. To re-cap, if knocking over tile i will also cause tile j to be knocked over, i->j will be an edge in AL. Thus there are N number of vertices in AL and M edges where each edge models the above relationship. This time you will be given a query max_num_knocked_over(z) where given a tile z return the maximum number of tiles that can be knocked over (excluding z itself) if z was knocked down by hand. There are Q such queries. Each query should be answered in worst case O(1) time. <- NOT EASY If your query runs in worst case O(V+E) time. You will get 30% of the marks. If your query runs in > worst case O(V+E) time, you will get no marks. Your solution may involve a pre-processing step that should run in time at most worst case O(V+E). If your pre-processing step takes more than O(V+E), your marks that you get for the query operation will be halved. If needed, you may assume you have a DFS algorithm DFSLabel(B,visited,v,c) that will take in an integer array B of size v, a visited array (visited[i] = 0 means vertex i is not visited, visited[i] = 1 means vertex is visited), a source vertex v and an integer value c. It will perform DFS from v as source vertex and also label v and all unvisited vertices v' reachable from v with c (by setting B[v] = c and B[v'] = c).

0/10

A: Consider that each domino is a vertice, and that we have the edges between the vertices. We can set up the graph such that we would need to count the the number of Strongly Connected Components/Cycles. First, we perform Kosaraju Alogorithm to determine the number of SCC involves. We would want to keep track of the paths in each SCC using a hashset Loop the given number of hashsets, and in each hashset ,use a counter to count the number of vertices in the given SCC Determine the maximum counter

Grader: Faculty
Points: 0/10
Comments: vague solution.
1. Not sure how the hashset is used to keep track of paths in each SCC and why there are multiple hashsets.