Name: Rollno:

ESO207: Data Structures and Algorithms (Final Exam)

19th November 2023

Total Number of Pages: 11 Time: 3 hr Total Points 100

Instructions

1. All questions are compulsory.
2. Answer all the questions in the space provided in the question paper booklet.
3. Use the space provided in the paper for rough work.
4. The symbols or notations mean as usual unless stated.
5. You may cite and use algorithms and their complexity as done in the class.
6. Cheating or resorting to any unfair means will be severely penalized.
7. Superfluous and irrelevant writing will result in negative marking.
8. Using pens (blue/black ink) and not pencils. Do not use red pens. for answering.
9. Please bring your ID cards.
10. For questions involving algorithm design, it is sufficient to give a clear and concise description of your algorithm. You dont need to give the formal pseudocode.
11. If you design an algorithm, that is less efficient than what has been asked, you will get partial credit, provided you clearly mention what is the time complexity of your algorithm.
12. It is advisable to solve a problem first before writing down the solution.
13. The questions are *not* arranged according to the increasing order of difficulty.

Question	Points	Score
1	5	
2	10	
3	25	
4	5	
5	5	
6	10	
7	10	
8	10	
9	10	
10	10	
Total:	100	

Question 1. (5 points) Fill in the blank type questions.

(a) If a graph on n vertices has k connected components, then the number of edges in the graph is at least $\underline{\quad n-k \quad}$.

(b) Let $f(n) = n^a$, $g(n) = n^b$ and $h(n) = n^c$. If $f(g(n)) = O(h(n))$, then what is the relation between a, b and c?

Answer: $\underline{\quad ab \leq c \quad}$

(c) Let u and v be two vertices in the queue during an execution of BFS from s. Then difference between `Distance`(s, u) and `Distance`(s, v) is at most $\underline{\quad 1 \quad}$.

(d) If $T(n) = T(n/7) + T(5n/6) + 2n$, then $T(n) = O(\underline{\quad \mathbf{n} \quad})$. (Give the best possible bound)

(e) If the array $A = [5, 4, 3, 2, 1]$ is converted into a heap using the linear time function `Build_Heap`, then the last element in the corresponding array is $\underline{\quad 4 \quad}$.

Question 2. (10 points) Multiple choice type questions (**one or more** choices may be correct).

(a) Let G be an undirected weighted graph where every edge has a distinct weight. Consider the following two statements:

A : G has a unique minimum spanning tree.

B : The maximum weight edge in G is not part of some minimum spanning tree of G.

Which of the following statement is correct?

A. A is true and B is false

B. A is false and B is true

C. Both A and B are true

D. Both A and B are false

(b) Consider the standard implementation of heapsort discussed in class.

A. Heapsort is stable but not in-place sorting algorithm

B. Heapsort is in-place but not stable sorting algorithm

C. Heapsort is stable and in-place sorting algorithm

D. Heapsort is neither stable nor in-place sorting algorithm

(c) Consider the radix sort algorithm to sort n integers having d digits each, and each digit can take k different values. Then which of the following statement is true?

A. Radix sort takes time $O(n^d)$ and extra space $O(n + k)$

B. Radix sort takes time $O(n^d)$ and extra space $O(nk)$

C. Radix sort takes time $O(dn)$ and extra space $O(n + k)$

D. Radix sort takes time $O(dn)$ and extra space $O(nk)$

Solution: Marks have also been given if someone has marked all 4 choices A,B,C,D as correct.

(d) Which of the following is an equivalent definition of a tree?

A. The graph is connected and acyclic.

B. The graph has n vertices and $n - 1$ edges.

C. There is a unique path between every pair of vertices in the graph.

D. The graph is connected and has at least $n - 1$ edges.

(e) Which of the following statements is/are true?

A. The DFS tree of a graph is always unique.

B. If DFN(x) < DFN(y), then x is an ancestor of y in the DFS tree.

C. If x is an ancestor of y in the DFS tree, then DFN$(x) <$ DFN(y).
D. The time complexity of DFS is $O(|V| + |E|)$.

Question 3. State whether the following statements are true or false. Give reasons or a counterexample to suitably justify your answer.

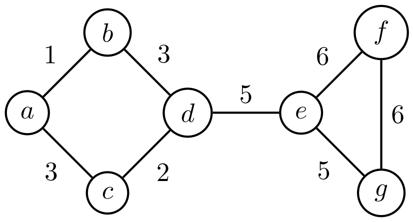
(a) (4 points) Every directed acyclic graph (DAG) has at least one source.

Solution: True.
Start at a vertex and travel backwards. If the graph has no source then this can be done indefinitely. However since the graph is finite, you would eventually reach a vertex that has already been visited. This gives a cycle and contradicts the fact that the graph is acyclic.
Grading Rubric:

- Answer - 1 point
- Argument - 3 points

(b) (4 points) In a weighted undirected graph, if each edge weight appears at most twice, then the graph has at most two MSTs.

Solution: False.
Consider the graph.



It has 4 MSTs.
Grading Rubric:

- Answer - 1 point
- Correct graph - 2 points
- Showing the MSTs - 1 points

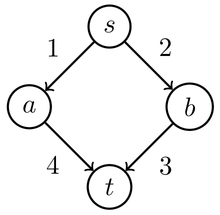
(c) (4 points) If we had an algorithm to square an n-bit number in $O(n)$ time, then we can multiply two n bit numbers in $O(n)$ time.

Solution: True.
We use the relation $xy = [(x+y)^2 - x^2 - y^2]/2$. Addition and subtraction takes linear time. And division by 2 is just removing the last bit.
Grading Rubric:

- Answer - 1 point
- Argument - 3 points

(d) (4 points) Let G be a directed graph where each edge is assigned a unique positive weight. If there is a path from s to t in G, then there is a unique shortest path from s to t.

Solution: False.



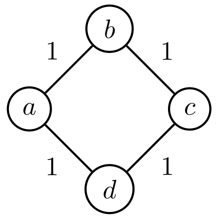
There are two shortest paths from s to t of weight 5 each.

Grading Rubric:

- Answer - 1 point
- Correct graph - 2 points
- Mentioning the shortest paths - 1 points

(e) (4 points) Let G be a weighted undirected graph and let C be a cycle in G. Then every minimum weight edge in C is part of every MST of G.

Solution: False.



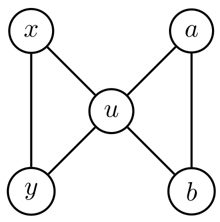
Grading Rubric:

- Answer - 1 point
- Correct graph - 2 points
- Justification - 1 points

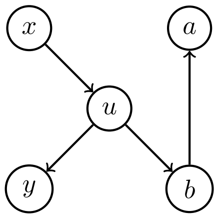
(f) (5 points) Let T be a DFS tree obtained by doing DFS in a connected undirected graph G. If u is an articulation point in G such that x is an ancestor of u in T and y is a descendent of u in T, then all paths from x to y in G must pass through u.

Solution: False.

Consider the following graph G.



Here is a DFS tree of G, by starting DFS at x, say T.



Clearly u is an articulation point in G, and x is an ancestor of u in T and y is a descendent of u in T. However there is a path from x to y in G that does not pass through u (the edge (x, y)).

Grading Rubric:

- Correct graph G - 1 point
- Correct DFS tree - 1 point
- Labelling x, y, u correctly in G and T - 1 point
- Showing alternate path from x to y in G - 1 point

Question 4. A triangle in a graph is a cycle of length 3. Consider the following algorithm to check if a connected, undirected graph G has a triangle.

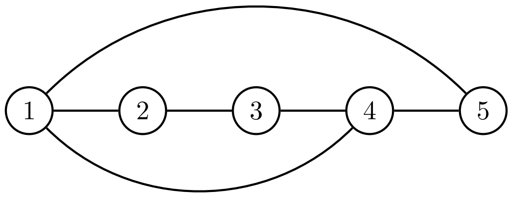
- Run DFS on G starting at any vertex.
- In the DFS tree of G, if there is a non-tree edge between a vertex and its grandparent, then G has a triangle.
- If no such edge is present, then G does not have a triangle.

Answer the following questions about the above algorithm.

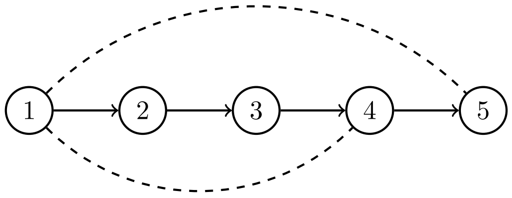
(a) (1 point) Does the above algorithm correctly detect if G has a triangle? **No**

(b) (4 points) If yes, then give a proof of correctness of the algorithm, else give a counterexample.

Solution: Consider the following graph G.



Here is the DFS tree obtained by doing DFS from 1, along with the non tree edges;



Now there is no edge between a vertex and its grandparent in the DFS tree. But the graph has a triangle $1-4-5-1$.

Grading Rubric:

- Correct graph G - 2 points
- Correct DFS (tree and non-tree edges) - 1 point
- Showing the triangle - 1 point

Question 5. Consider the algorithm discussed in class, `Select`(S, i) to find the i-th smallest element in a set S, containing n elements. Suppose we group the elements into groups of 3 elements each, instead of 5 elements.

(a) (1 point) Will the algorithm still give the correct answer? **Yes**

(b) (2 points) If we collect the median of all the groups (say M), and compute the median of M (say x), then what can we claim about x?

Solution: x is a 1/3-approximate median of S.

(c) (2 points) Give an analysis of the running time of this algorithm.

Solution: The running time of the algorithm would be

$$\begin{aligned} T(n) &= T(n/3) + T(2n/3) + cn \\ T(1) &= d \end{aligned}$$

Then $T(n) = O(n \log n)$.

Grading Rubric:

- Recurrence relation - 1 point
- Final answer - 1 point

Question 6. (10 points) You are given an $m \times n$ matrix A, filled with distinct integers, such that each row is sorted in ascending order from left to right, and each column is sorted in ascending order from top to bottom.

Design an $O(m+n)$ time algorithm that given the matrix A and a number x, searches if x is present in A. Analyze the time complexity of your algorithm.

Solution: Here is the algorithm to find x in the matrix A.

```
Algorithm 1: FindElement
Data: Given an $m \times n$ matrix $A$, number $x$
$i \leftarrow m-1$;
$j \leftarrow 0$;
while $i \geq 0$ and $j \leq (n-1)$ and $A[i,j] \neq x$ do
    if $A[i,j] < x$ then
        $j \leftarrow j+1$;
    else
        $i \leftarrow i-1$;
    end
end
if $A[i,j] = x$ then
    return found;
else
    return not found;
end
```

Time Complexity Analysis: In each iteration of the while loop either i is decremented by 1 or j is incremented by 1. i runs from $m-1$ to 0 and j runs from 0 to $n-1$. Number of steps inside the while loop is constant. Therefore time complexity is $O(m+n)$.

Grading Rubric:

- Algorithm - 8 points
- Time complexity - 2 points
- If $O(mn)$ time algorithm is given then 2 points will be awarded (including points for time complexity).
- If $O(m \log n)$ time algorithm is given then 3 points will be awarded (including points for time complexity).
- If $O(m \log n)$ time algorithm is given then 3 points will be awarded (including points for time complexity).
- If matrix structure is incorrectly assumed then 2 points will be given. For example in the following matrix the output is incorrect, if x is considered as 4.

$$\begin{pmatrix} 1 & 2 & 5 \\ 4 & 12 & 16 \\ 8 & 17 & 18 \end{pmatrix}$$

Question 7. (10 points) An array is said to be k-sorted if each element in the array is at a distance at most k from its final sorted position. For example the array $[8,3,10,12,5]$ is 3-sorted, since the element 5 is at a distance 3 from its position in the final sorted array.

Given a k-sorted array containing distinct integers, design an $O(n \log k)$ time algorithm to sort the array. You may use $O(k)$ extra space. Analyze the time complexity of your algorithm.

Solution: We will maintain a heap of size $k+1$ and extract the minimum element in each iteration.

Build a heap consisting of the first $k+1$ elements of the array. Run a loop for $(n-k-1)$ times. In each iteration of the loop, extract the minimum element from the heap, and insert the next element of the array into the heap. After that, extract the minimum element from the heap for $(k+1)$ iterations.

Since the heap has size $k+1$, inserting and extracting the minimum element from the heap takes time $O(k)$. Both loops run for a total of n iterations. Therefore the time complexity is $O(n \log k)$.

Alternate Solution: Sort the first $2k$ elements. This ensures the smallest k elements are in the first k positions. Next consider elements from position $k+1$ to $3k$ and sort them. This ensures the next k elements are in the next k positions. Do this n/k times to sort the entire array.

Grading Rubric:

- Algorithm - 7 points
- Time complexity analysis - 3 points
- $O(nk)$ or $O(nk \log k)$ solution will get 3 marks
- Incorrect algorithm will get 0 marks

Question 8. Given an initial sequence $S = \langle x_0, x_1, \ldots, x_{n-1} \rangle$ of n numbers, your task is to design a linear sized data structure to perform the following operations in $O(\log n)$ time. Assume n is a power of 2.

`Report_sum`(i, j): Reports the sum $\sum_{k=i}^{j} x_k$, that is, the sum of the elements from x_i to x_j.

`Update`(i, a): Update the value of x_i to a.

(a) (4 points) Give a clear and concise description of the data structure that you will be using.

Solution: Maintain a complete binary tree of size $2n-1$ where the leaf nodes store the elements of the sequence from left to right. And each internal node stores the sum of the leaves of the rooted at that node. The binary tree is stored in an array A of size $2n-1$ indexed from 0 to $2n-2$, where the elements of S are stored in the last n positions from $A[n-1]$ to $A[2n-2]$.

Grading Rubric:

- Complete binary tree of size $2n-1$ - 1 point
- Storing the sequence S in the leaves - 1 point
- Each internal node stores the sum of the leaves of the subtree rooted at that node - 2 point

(b) (6 points) Give algorithms for the operations `Report_sum`(i, j) and `Update`(i, a).

Solution:

Algorithm 2: Report_Sum

```
Function Report_Sum(i, j):
    $i \leftarrow (n-1) + i$;
    $j \leftarrow (n-1) + j$;
    $sum \leftarrow A[i]$;
    if $i < j$ then
        $sum \leftarrow sum + A[j]$;
        while $\lfloor (i-1)/2 \rfloor \neq \lfloor (j-1)/2 \rfloor$ do
            if $i\%2 = 1$ then $sum \leftarrow sum + A[i+1]$;
            if $j\%2 = 0$ then $sum \leftarrow sum + A[j-1]$;
            $i \leftarrow \lfloor (i-1)/2 \rfloor$;
            $j \leftarrow \lfloor (j-1)/2 \rfloor$;
        end
    end
return
```

Algorithm 3: Update

```
Function Update(i, a):
    $i \leftarrow (n-1) + i$;
    $diff \leftarrow a - A[i]$;
    if $i \geq 0$ then
        $A[i] \leftarrow A[i] + diff$;
        $i \leftarrow \lfloor (i-1)/2 \rfloor$;
    end
return
```

Grading Rubric:

- `Report_Sum` - 4 marks
- `Update` - 2 marks

Question 9. (10 points) Let $G = (V, E)$ be a directed graph with a weight function $w : E \rightarrow \mathbb{R}^+$ that assigns positive weights to its edges. You need to compute the shortest distance from s to t in G. However there is a catch. You can choose to reduce the weight of any one edge in G by half. Let $t(n)$ be the time complexity of Dijkstra's algorithm.

Design an $O(t(n))$ time algorithm to compute the shortest path from s to t in G, with respect to a weight function w', where w' is obtained from w by reducing the weight of any one edge by half.

Mention how you plan to choose the edge, whose weight you will reduce by half.

(Note: The input to your algorithm will be $\langle G, s, t, w \rangle$.)

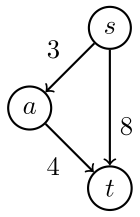
Solution:

```
Algorithm 4: ShortestPath
Data: Given $\langle G, s, t, w \rangle$
Compute a graph $G^r$ by reversing all the edges in $G$ (weights remain the same);
Run Dijkstra's in $G$ starting from $s$ and store the distances in the array DistFromS;
Run Dijkstra's in $G^r$ starting from $t$ and store the distances in the array DistToT;
$min \leftarrow \infty$;
foreach edge $e = (u, v) \in G$ do
    $cost \leftarrow$ DistFromS$(u)$ + DistToT$(v) + w(e)/2$;
    if $cost < min$ then $min \leftarrow cost$;
    return $min$;
end
```

Choice of edge: Choose the edge (u, v) that minimizes the quantity DistFromS(u) + DistToT(v) + $w(u, v)/2$ over all edges (u, v).

Grading Rubric:

- Algorithm - 8 points
- Correct edge choice - 2 points
- If $O(m \cdot t(n))$ time algorithm is given then 3 points will be awarded.
- Incorrect algorithm - 0 marks. Following counterexample will work in most cases.



Question 10. (10 points) You are given a set of n intervals $S = \{[a_i, b_i] \mid 1 \leq i \leq n\}$ on the real line. Design an $O(n \log n)$ time algorithm to find the smallest subset C of S, such that union of all intervals in C is equal to the union of all intervals in S. Prove the correctness of your algorithm.

Solution:

```
Algorithm 5: IntervalCover
Data: Given set $S$ of $n$ intervals
$C \leftarrow \emptyset$;
Sort all intervals in $S$ in increasing order of $a_i$'s. For intervals having equal $a_i$, order them
  in decreasing order of $b_i$'s;
while $S \neq \emptyset$ do
    Let $x = [a_i, b_i]$ be the first interval in $S$;
    $C \leftarrow C \cup \{x\}$;
    foreach $y = [a_j, b_j] \in J$ such that $a_j < a_i$ do
        Set $a_j \leftarrow a_i$;
    end
    $S \leftarrow S - \{[a_j, b_j] \mid b_j \leq b_i\}$;
end
return $C$;
```

Proof of Correctness:

Lemma 1: There is an optimal solution of S that contains the interval with the least start time and maximum finish time (i.e. our choice x in the algorithm).

Proof of Lemma 1: Consider an optimal solution that does not include x. There must be some interval y, in the solution with start time same as x, since x must be covered as well. Exchange y with x to get another collection of same size. This is also a solution since finish time of x is greater than finish time of y, by choice of x. Hence x would cover all intervals that y had covered.

Let J be the collection of intervals at the beginning of some iteration of the while loop, and let J' be the collection of intervals at the end of that iteration. Let $Opt(J)$ be the size of an optimal solution of J. We will show that $Opt(J) = Opt(J') + 1$. This proves the correctness of the algorithm since one interval is added to the solution set in each iteration of the while loop.

Proof of $Opt(J) \leq Opt(J') + 1$: Consider an optimal solution O', of J'. Note that J' does not contain x. If we add x to O' then all intervals in J are covered as well. This is because the interval removed from J to form J', have finish time less than the finish time of x. Therefore x covers all these intervals. Therefore this forms a solution of J.

Proof of $Opt(J') \leq Opt(J) - 1$: Consider an optimal solution O, of J that contains x. Such a solution is guaranteed to exist by Lemma 1. Now if we remove x from O, we claim that this set would cover all intervals in J'. Let y be an interval in J'. Then start time of y is greater than or equal to the finish time of x. Therefore there was some interval in O other than x that covered y. The same interval would cover y now as well. Therefore this set forms a solution of J'.

Grading Rubric:

- Algorithm - 3 points
- Proving there is an optimal solution with the correct local choice - 3 points
- Proving $Opt(J) = Opt(J') + 1$ - 4 points