ESO 207A: End Semester Exam

Full time: 180 mts Full marks: 100

Full time for this examination is 180 mts.
Mention your name and roll number in each page of the paper.
Answer the questions in the spaces provided on the question sheets.
Give brief & precise explanations for your answers wherever appropriate.
Kill extra space after your answer in each page.

(April 24, 2018)

Name: ______________________

Roll No: ______________________

1. (15 points) Mark TRUE/FALSE. No explanation is required. Each correct answer will fetch 1 mark and a wrong answer -1 mark. Write T or F clearly in the space provided. Any ambiguous answer will be treated as wrong answer.
   (a) TRUE/FALSE: A RAM program is stored in the memory and it can be modified. (**FALSE**)
   (b) TRUE/FALSE: Average time complexity is defined as the average of the running times over all the inputs of a given size. (**TRUE**)
   (c) TRUE/FALSE: No data shifting is required in insertion sort to sort a reverse sorted sequence. (**FALSE**)
   (d) TRUE/FALSE: Running times $f(n)$ and $g(n)$ are comparable, if $0 < \lim_{n\to\infty} \frac{f(n)}{g(n)} < 1/2$. (**TRUE**)
   (e) TRUE/FALSE: Preorder traversal of a tree is generated by listing out a node when it is visited third time by a Euler traversal around the tree starting with the root. (**FALSE**)
   (f) TRUE/FALSE: The pre-order successor of a node x in a BST is the smallest element in the left subtree (LST) of x, assuming that the LST is nonempty. (**FALSE**)
   (g) TRUE/FALSE: A LL type single rotation involves fixing pointers of three nodes: the root of the tree, its right child and the left child of its right child. (**TRUE**)
   (h) TRUE/FALSE: In a red black tree, every path from the root to a leaf node has the same number of red nodes. (**FALSE**)
   (i) TRUE/FALSE: Division function for hashing maps any key of form $k = (am + x)$ to $h(x)$ even if m is prime. (**TRUE**)
   (j) TRUE/FALSE: Primary clustering occurs in hashing with open addressing when multiple keys map to a same hash table slot and linear probing is used for resolving collisions. (**TRUE**)
   (k) TRUE/FALSE: For a cross edge $x \to y$ in DFS of a directed graph, the DFS discovery time of x must be lower than the DFS discovery time of y. (**FALSE**)
   (l) TRUE/FALSE: Deleting an articulation point from a graph G splits it into two or more disconnected pieces. (**TRUE**)
   (m) TRUE/FALSE: A graph where every edge weight is unique (there are no two edges with the same weight) has a unique MST. (**TRUE**)
   (n) TRUE/FALSE: In a BFS, the set of vertices that are at level $i + 1$ equal to the vertices reachable from the vertices at level i by one hop. (**FALSE**)
   (o) TRUE/FALSE: An MST of a graph can be used to find shortest path between two vertices. (**FALSE**)

2. (10 points) Give short answers against each question in the space provided. You are not required to give any reason for your answers.

   (a) (1 point) What is the worst case time complexity for sorting n elements using heap sorting?
   Solution: $\mathrm{O}(n \log n)$.

   (b) (1 point) Which of the sorting algorithms: insertion sort, bubble sort or selection sort is most efficient when input array is almost sorted?

   Solution: Insertion sort.

   (c) (1 point) What is the running time of insertion sort when all keys are equal.

   Solution: $\mathrm{O}(n)$

   (d) (1 point) What is the number of comparisons required for unsuccessful search when a hash table has size 20 and in which 60 keys are stored assuming that chaining is used for collision resolution?

   Solution: $1+\alpha/2 = 1+3 = 4$.

   (e) (1 point) What is color of a newly inserted node in a red black tree?

   Solution: Red

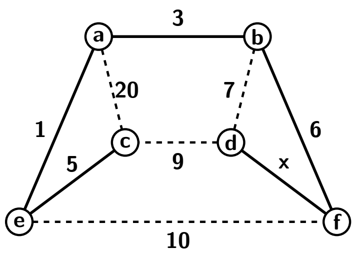
   (f) (1 point) Let h and bh respectively denote height, and black height of a red black tree. What is the upper bound of h in terms of bh?

   Solution: $h \leq 2bh$.

   (g) (1 point) What is the asymptotic running time of Knuth-Morris-Pratt (KMP) algorithm.

   Solution: $\mathrm{O}(n+m)$.

   (h) (3 points) The solid edges are part of the MST in the graph shown in the figure below. One of the edge weight is marked x. What is the range of values for x, assuming all the edge weights are positive integral values?



   Solution: [1, 7].

3. (6 points) Suppose odd-even merge sort is called on the sequence of input numbers as in the following figure. Give the output of each phase (even and odd) indicating low and high comparison orders (using arrows pointing from low to high).

Solution:

Write solution at the appropriate places in the space provided in the figure

Odd	Even	Odd	Even	Odd	Even	Odd	Even	Sorted
4	4	4	1	1	1	1	1	1
8	8	1	4	2	2	2	2	2
2	1	8	2	4	3	3	3	3
1	2	2	8	3	4	4	4	4
9	5	5	3	8	5	5	5	5
5	9	3	5	5	8	7	7	7
7	3	9	7	7	7	8	8	8
3	7	7	9	9	9	9	9	9

[1-Mark each for each column except last 2.]

[If last two are not same -1 Mark]

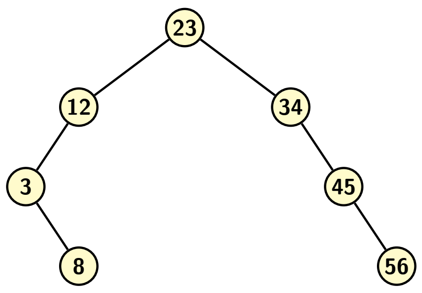
4. (3 points) Draw the final Binary Search Tree (BST) after all the following insertions are complete.

$$23, 12, 34, 3, 45, 8, 56$$

Assume that elements are inserted in the order they appear from left to right, and also assume that the smaller elements are inserted to the left.

Solution:

The tree is:



[Binary Grading]

5. (3 points) Given $n > 2$ points on a 2D-plane (not all collinear), what is the minimum number of points on the convex hull of n points, and why?

   Solution: The minimum number points that define a convex polygon is a triangle. Since not all n points are collinear, we must have at least a set of three points that can define a triangle. If $n - 3$ input points lie completely insides a triangle then we have a convex hull with minimum number of points.

   [Binary Grading]

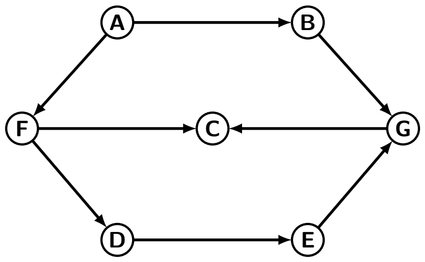
6. (3 points) Suppose a graph $G = (V, E)$ has positive weights defined on the vertices instead of edges. Is it possible to compute shortest path using Dijkstra's shortest path algorithm for G, if so how, if not why not?

   Solution: Yes, just assign edge weight as the sum of weights of end vertices and used Dijkstra's algorithm.

   [Binary Grading]

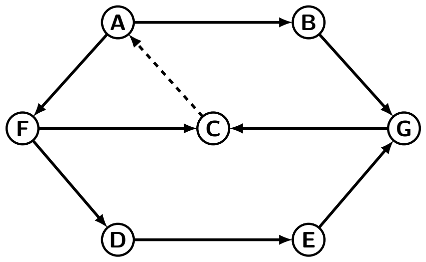
7. (8 points) A strongly connected graph G is a directed graph such that for any pair of vertices $u, v \in G$ there is directed path from u to v and vice versa.

   (a) (2 points) What is the minimum number of edges which when added to the following directed graph makes it strongly connected. Specify those edges.



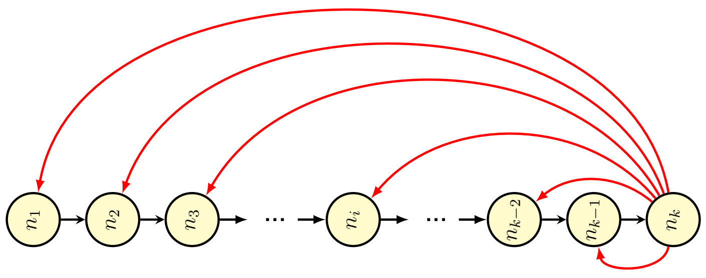
   (b) (6 points) Give an example of a directed graph G with minimum number of edges and k strongly connected components such that if a single edge is added to it then the number of strongly connected components can be reduced by any number from 1 to $k-1$. (HINT: Think about simple graphs.)

   Solution (a): Only one edge (C, A) should be added.



   [Binary Grading]

   Solution (b): A pathological case is exhibited by a linear tree strucutre as shown below.



Adding a single edge can reduce the number of strongly connected components by any where between 1 to $n-1$. So, if m and m' respectly denote the number of SCCs in G and $G' = (V, E \cup \{e\})$, then we have $m' \leq m$ and $m' \geq 1$.

[Correct example 4 Marks, Edges defined and correct 2 Marks]

8. (3 points) Give an example of a weakly connected, directed acyclic graph with minimum number of edges (but > 0) such that it has non-unique topological orderings of the vertices.

   Note: A directed graph is weakly connected if the underlying undirected graph obtained by ignoring direction of edges is connected.

   Solution: Consider graph $G = (V, E)$ where $V = \{v_1, v_2, v_3\}$ and $E = \{(v_1, v_2), (v_1, v_3)\}$. Then two possible orderings would be v_1, v_2, v_3 and v_1, v_3, v_2.

   [Binary Grading]

9. (9 points) This question is about the Knuth-Morris-Pratt's algorithm for pattern matching.

   (a) (3 points) What would be the general form of a pattern that maximizes KMP prefix function π, i.e., maximize $\sum_{i=1}^{m} \pi(i)$.

   (b) (2 points) What is the maximum value of the above sum.

   (c) (3 points) What would be the general form of a pattern that minimizes KMP prefix function π, i.e., minimizes $\sum_{i=1}^{m} \pi(i)$.

   (d) (1 point) What is the minimum value of the above sum.

   Solution (a): Pattern having the same character repeated throughout. This is because any proper suffix will be a proper prefix

   [Explanation Missing, deduck 1 Mark]

   Solution (b): When same character is repeated throughout, prefix function will increase by 1 going from i to $i+1$, $0 \leq i \leq m$. Since, $\pi(1) = 0$, the maximum value of the sum

$$\sum_{i=1}^{m} \pi(i) = \sum_{i=1}^{m} (i-1) = \frac{m(m-1)}{2}$$

   [Explanation Missing, deduck 1 Mark]

   Solution (c): Pattern where first character is never repeated. It means no suffix can be a prefix except for the null string ϵ. [Explanation Missing, deduck 1 Mark]

   Solution (d): Since $\pi(i) = 0$, for $1 \leq i \leq m$, the value of the sum $\sum_{i=1}^{m} \pi(i) = 0$.

   [Binary Grading]

10. (6 points) Following algorithm takes an array as input and returns the array with all duplicates removed.

```
$S$ = newEmptySet();
A = newDynamicArray(); // A is new array elements without duplicates
for (every element $x \in$ InputArray) {
    if (S.Find($x$) == Nil) { // returns Nil if $x \notin S$
        S.Insert($x$); //
        A.append($x$);
    }
}
```

For example, if an input array has [1, 2, 2, 4, 5, 5, 2, 3] then algorithm outputs [1,2,4,5,3]. What is the big Oh complexity of the above algorithm if the set is implemented using:

(a) (3 points) An AVL tree?

(b) (3 points) A hash table?

Provide adequate but brief explanation (not exceeding 2-3 lines) for your answer in each case.

Solution (a): O($n \log n$). For loop runs n times at most. Each insert takes O($\log n$) time in AVL tree. Append take time O(n) time in total as at most n element can be appended to output A. Overall time is O($n \log n$) time.

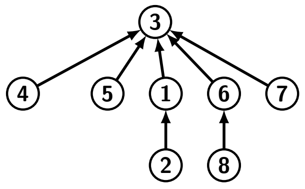
[2 Mark if no explanation]

Solution (b): Expected O(n). Because hash table operation takes expected time of O(1).

[If expected key word is missing then 1 mark less]

11. (7 points) This question is about UNION and FIND operations on disjoint sets. Each set represented by a tree with the root representing the name of the set. Assume that UNION has to apply FIND on its arguments to get the names of sets before performing UNION by size. Also assume that if a tie occurs (on the sizes) then the name of the set containing the first argument becomes the name of the output set.

(a) (3 points) Suppose following disjoint set was generated without using path compression.



Which of the following sequence of UNIONs would be appropriate for generating the above set.

i. UNION(3, 7), UNION(3, 6), UNION(6,8), UNION(3, 1), UNION(1,2), UNION(3, 5), UNION(3,4).

ii. UNION(4,3), UNION(5,3), UNION(2,1), UNION(2, 3), UNION(8,6), UNION(8,3), UNION(7,3).

iii. UNION(1,3), UNION(3,4), UNION(4,5), UNION(2, 5), UNION(6,8), UNION(6,2), UNION(7,6).

iv. None of the above.

(b) (4 points) Disjoint set operations UNION and FIND can be used to design an algorithm to detect cycles in a graph. The outline of this algorithm is given below with parts are missing. Missing part are marked by labels 1, 2, and 3. Provide the missing parts.

Step 1: Create a disjoint set for each –1–

Step 2: For every edge (u, v) in G

i. –2–

ii. –3–

iii. UNION(S_u, S_v) // S_u and S_v are set containing u and v, respectively.

Solution (a): Only (iv).

Solution (b): Blank(1): vertex of the input graph G. [1 Mark]

Blank(2): Find root of the sets containing u and v. [1 Mark]

Blank(3): If u and v belong to same set then a cycle is found. [2 Mark]

12. (7 points) This question is about text encoding and decoding.

    (a) (5 points) Build Huffman code tree for coding text in which frequencies of occurrences of the letters are:

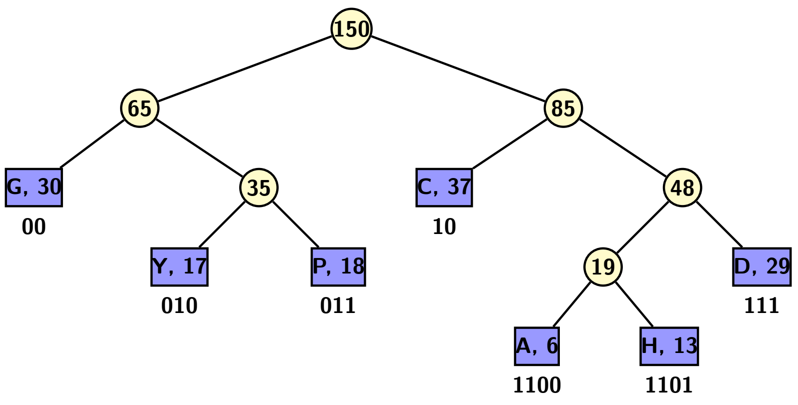
A	C	D	G	H	P	Y
6	37	29	30	13	18	17

    (b) (2 points) Using the above code decode the following coded message: 110111001100011011010.

Solution (a): Huffman code tree for the frequencies is given below.

A	C	D	G	H	P	Y
1100	10	111	00	1101	011	010

The corresponding code tree is:



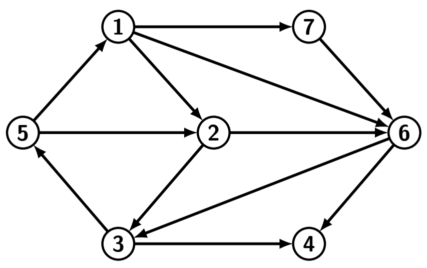
[Consistent in selection left/right child for any credit]

Solution (b): The coded string is: 1101 1100 1100 011 011 010 when decoded gives "HAPPY"

[Binary Grading]

13. (5 points) Classify the edges of the following directed graph assuming that the vertices have been labeled with their respective DFS discovery times.

Solution

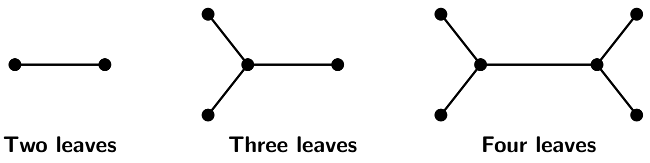


Write your solution in space provided in table:

Tree Edges	Back Edges	Cross Edges	Forward Edges
(1, 2)	(5, 1)	(6, 3)	(1, 6)
(1, 7)	(5, 2)	(6, 4)	
(2, 3)		(7, 6)	
(2, 6)			
(3, 5)			
(3, 4)			

[2 Mark - Tree edges, 1 Mark each Back, Forward and Cross edges]

14. (15 points) A free unrooted tree is a connected graph with no cycles. Vertices with one neighbor each are leaves, and remaining vertices are internal. If each internal vertex has a degree exactly three, then such a tree is called a free binary tree. First three of free binary trees are shown in the figure below.



Now answer the following questions.

(a) (10 points) Prove that if a free binary tree T has $k > 2$ leaves, then there must be at least one internal node in T which is connected to two leaves.

(b) (5 points) Use the above result to prove by induction that if a free binary tree T has $k > 1$ leaves then it has $2k - 2$ nodes.

Solution (a): To prove it, consider the maximum length path $P : v_0, v_1, \cdots, v_m$ in T where v_0 and v_m are leaves. Since $k > 2$, P must have an internal node, and also the degree of this internal node is exactly 3. Consider node v_{m-1} on P. It has v_m as one of its neighbors and the other neighbor is v_{m-2}. Since v_{m-1} an internal node, it must also have a third neighbor, say x. If x is not a leaf then we have a longer path than P namely, $v_0, v_1, \cdots, v_{m-1}, x, nbr(x), \cdots$. Hence, x must be a leaf.

[Proof by contradiction through non terminating tree 8 marks, Other correct proofs 10 marks]

Solution (b): For a free binary tree with $k = 2$ leaves, there is no internal node and two leaves are connected by one edges.

Basis: The number of nodes in the tree is $2k - 2 = 2.2 - 2 = 2$. So the statement holds.

Assume that any free binary tree with $k - 1$ leaves has $2(k - 1) - 2$ nodes. Now consider a free binary tree T with k leaves.

Using the result proved in part (a), for $k > 2$, we have at least on internal node v_I connected to 2 leaves. We delete the two leaves from v_I, then v_I becomes a leaf. Call the new tree as T'. T' has $k - 1$ leaves. So, by induction hypothesis the number of nodes in T' is $2(k - 1) - 2$. The difference in the number of nodes between T and T' is 2. So, T has $2(k - 1) - 2 + 2 = 2k - 2$ nodes.

[No mark for non inductive proofs]