EL9343 Final Exam Solutions (2020 Spring)

Name: **ID:**

Exam duration 9:30AM-12:00PM, Upload deadline: 12:10PM

May 13, 2020

Write all answers on your own blank answer sheets, scan and upload answer sheets to NYUclasses at the end of exam, keep your answer sheets until the grading is finished. Make sure you use Adobe Scan to have all you pages in one single PDF file. Check your submission online before leave the exam.

Multiple choice questions may have multiple correct answers. You will get partial credits if you only select a subset of correct answers, and will get zero point if you select one or more wrong answers.

1. (**24 points**) **True or False (3 points each)**

   (a) **T or F:** Worst case tree height for AVL tree with n nodes is $\theta(\log(n))$;

   (b) **T or F:** For an undirected graph with n vertices to be acyclic, the maximum number of edges is $n-1$;

   (c) **T or F:** For an undirected graph, DFS yields no forward or cross edges;

   (d) **T or F:** Any P problem is also a NP problem;

   (e) **T or F:** 0-1 knapsack problem can be solved by dynamic programming, and computation time is a polynomial function of the input size;

   (f) **T or F:** If adjacency list is used to represent a graph, the worst case time complexity to determine whether an edge (u, v) exists is $\theta(E)$.

   (g) **T or F:** There is only one valid topological ordering of vertices in a directed acyclic graph.

   (h) **T or F:** Kruskal's algorithm for minimum spanning tree does not work on a graph with negative weight edges.

2. (**4 points**) For a given node v in a binary search tree, which of the following statements about its successor are true?

   (a) v always has a successor in the tree;

   (b) The successor can be the left child of v's right child;

   (c) The successor can be the right child of v's left child;

   (d) The successor can be v's right child;

(e) The successor can be an ancestor of v.

3. (**4 points**) Which of the following statements are true?

(a) The algorithm to decide whether there is a Hamiltonian cycle in a graph cannot always be completed in polynomial time;

(b) If problem A is reducible to problem B in polynomial time, then problem B is easier to solve than problem A;

(c) All NP-complete problems can be verified in polynomial time;

(d) If a polynomial time algorithm is found for a NP-complete problem, all NP-complete problems are solvable in polynomial time;

(e) None of the above.

4. (**4 points**) Which of the following statements about breadth-first search (BFS) are true?

(a) Complexity of BFS on a dense graph is $\theta(E)$;

(b) BFS search starting from a single vertex s always visits all vertices in a graph;

(c) BFS employs a queue to store the grey nodes;

(d) BFS can be used to solve single source shortest path in an unweighted graph;

(e) None of the above.

5. (**4 points**) Which of the following sets of codewords can be Huffman codes?

(a) [0,1];

(b) [00, 10, 11];

(c) [000,001, 01, 10, 11];

(d) [1, 01, 000, 001];

(e) None of the above.

6. (**4 points**) Which of the following statements about shortest path are true?

(a) In Bellman-Ford algorithm, the edges can be relaxed in any given order;

(b) Greedy-based shortest path algorithms work on a graph with negative weight edges;

(c) In a directed acyclic graph, shortest path distance between any pair of vertices is always well-defined;

(d) In Floyd-Warshall, the sub-problems are limited by the number of hops allowed;

(e) None of the above

7. (**4 points**) In the activity selection problem, which of the following is true?

(a) It can be solved by divide-and-conquer;

(b) It can be solved by greedy and select the activity with earliest finish time;

(c) It can be solved by greedy and select the activity with shortest duration;

(d) Any optimal solution cannot include the activity with the latest finish time;

(e) None of the above.

8. (**8 points**) For the AVL Tree in Figure 1, when a key with value 8 is inserted, how many rotations are needed to restore the AVL tree property? what is the restored AVL tree with the inserted key?

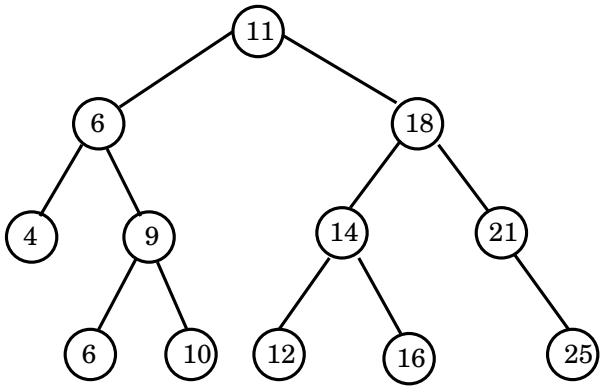
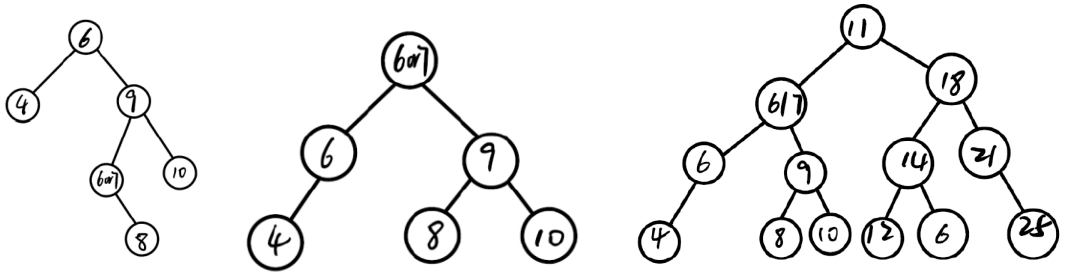


Figure 1: AVL Tree for Question 8

Answer: Node 8 will be inserted in the left subtree of the right child of node 6, see Figure 2(a), where the imbalance happens the first. This is an inside case, requires double rotation, see Figure 2(b). The final restored AVL tree is indicated in Figure 2(c).



(a) Tree After Insertion (b) Tree After Double-Rotation (c) Final Restored Tree

9. (**10 points**) Show how depth-first search works on the graph in Figure 2. Assume that the main loop of DFS considers vertices in alphabetical order, and assume that each adjacency list is ordered alphabetically. Show the discovery and finishing times for each vertex, and show the classification of each edge.

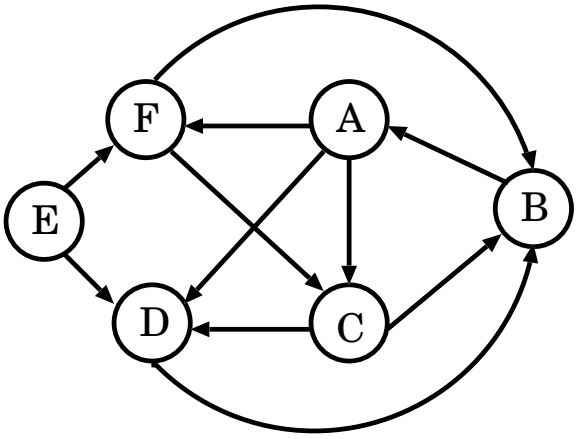


Figure 2: Directed Graph for Question 9

Answer:

	A	B	C	D	E	F
discovery	1	3	2	5	11	8
finish	10	4	7	6	12	9

Table 1: Discovery and Finish Time for Question 9

Tree Edges: AC, CB, CD, AF

Back Edges: BA

Forward Edges: AD

Cross Edges: EF, ED, DB, FC, FB

10. (**8 points**) Compute the transitive closure of the following directed graph using modified Floyd-Warshall algorithm, show the $T^{(k)}$ matrices at all steps, show the final transitive closure.

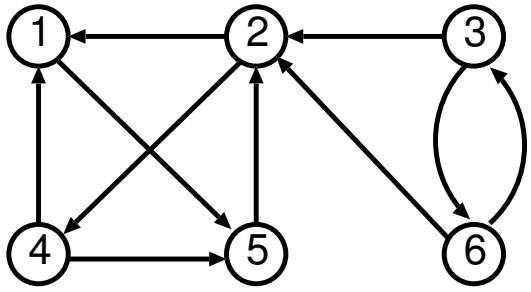


Figure 3: Directed Graph for Question 10

Answer:

$$T^{(0)} = \begin{matrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{matrix}$$

(a) Initial Matix

$$T^{(1)} = \begin{matrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{matrix}$$

(b) First Iteration

$$T^{(2)} = T^{(3)} = T^{(4)} = \begin{matrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{matrix}$$

(c) Second to Fourth Iteractions

$$T^{(5)} = T^{(6)} = \begin{matrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{matrix} ,$$

(d) Final Result

11. (**12 points**) For a given array $A[1 \cdots n]$, the *Maximum Subarray* problem is to find the contiguous subarray $A[l \cdots r]$, such that the summation of $A[l] + A[l+1] + \cdots + A[r]$ is the maximum among all contiguous subarrays. It can be solved by a divide-and-conquer algorithm with $O(n \log n)$ complexity. Develop another algorithm to solve it with $O(n)$ complexity. (*Hint: use dynamic programming, write down the recurrence of optimal solutions and the detailed pseudo code, and complexity analysis*)

Answer: Among all contiguous subarrays ended at A[k], including empty subarray, let $S[k]$ be the maximum summation. Then we have a recurrence:

$$S[k] = \max\{0, S[k-1] + A[k]\},$$

where 0 corresponding to empty subarray.

Obviously, we can calculate all S[k], $1 \leq k \leq n$ in linear time, and maintain the maximum value as the iteration goes. We also need to return the left and right indexes of the maximum subarray in the final output. To do that, we need to maintain another array $L[1 \cdots n]$, where $L[k]$ records the left index of the maximum contiguous subarray ended at $A[k]$, with $L[k] = k + 1$ for empty subarray. Then $L[k]$ can be updated as

$$L[k] = \begin{cases} k+1 & \text{if } S[k] == 0; \\ L[k-1] & \text{if } S[k] == S[k-1] + A[k]. \end{cases}$$

12. (**14 points**) Given a weighted undirected graph $G = (V, E)$, suppose all the edge weights are distinct, and a minimum-spanning tree T has been found, if the weight on one edge $e \in E$ is reduced from w_e to w'_e,

(a) (**6 points**) design an $O(|V|)$ algorithm to construct a new minimum-spanning tree T' for the updated graph G';

(b) (**3 points**) justify the complexity of your algorithm;

(c) (**5 points**) prove the correctness of your algorihtm.

Answer:

a). Suppose $e = \langle u, v \rangle$;
Case 1: if e was part of the minimum-spanning tree, then T is still the MST for G', the total weight is reduced by $w_e - w'_e$;
Case 2: if e was not part of T, run BFS on T starting from u to build a BFS tree rooted at u, find the largest weight edge m on the path from v to u, if $w_m > w'_e$, then generate a new MST by replacing edge m with edge e, i.e., $T' = T - \{m\} + \{e\}$.

b) Complexity analysis:
Case 1: T has $|V| - 1$ links, the algorithm completes within $O(|V|)$;
Case 2: BFS complexity is $O(|V| + |E|)$, since T has $|V| - 1$ edges, BFS on T takes $O(|V|)$, the path from v to u is at most $|V| - 1$ hops, so finding maximum weight edge is $O(|V|)$, the total complexity is $O(|V|)$.

c) Correctness:
Let $W(\cdot)$ be the weight function before the weight change, $W'(\cdot)$ be the weight function after the weight change;
Case 1: Suppose there is another MST T_1 has lower total weight than T in G', then $W'(T_1) < W'(T)$.

- if $e \notin T_1$, the total weight of T_1 was not affected by the weight decrease on e, then $W(T_1) = W'(T_1) < W'(T) < W(T)$, T_1 was MST for G, **contradiction!**
- if $e \in T_1$, then the total weight of edges other than e in T' should be lower than the total weight of edges other than e in T, i.e., $W'(T_1 - \{e\}) < W'(T - \{e\})$, since $e \notin T_1 - \{e\}$, and $e \notin T - \{e\}$, then we have $W(T_1 - \{e\}) = W'(T_1 - \{e\}) < W'(T - \{e\}) = W(T - \{e\})$, then $W(T_1) < W(T)$, T_1 was MST for G, **contradiction!**

Case 2:

If $w'_e \geq w_m$, our algorithm will still choose T as the MST for G'. Suppose there is another MST T_1 has lower total weight than T in G', then $W'(T_1) < W'(T)$.

- if $e \notin T_1$, the total weight of T_1 was not affected by the weight decrease on e, then $W(T_1) = W'(T_1) < W'(T) = W(T)$, T_1 was MST for G, **contradiction!**
- if $e \in T_1$, T_1 is a spanning tree, after removing $e = \langle u, v \rangle$, the vertices are partitioned into two subsets: $v \in V_1$ and $u \in V_2$. The path from v to u in T starts in V_1 ends in V_2, let l be the link connects V_1 and V_2, then $w_l \leq w_m \leq w'_e$, (since m has the maximum

weight on the path from v to u in T). $T_1' = T_1 - \{e\} + \{l\}$ is now a new spanning tree for G, and

$$W(T_1') = W'(T_1') = W'(T_1) - w_e' + w_l \leq W'(T_1) < W'(T) = W(T),$$

which means T_1' should be MST for G, **contradiction!**

If $w_e' < w_m$, our algorithm will find $T' = T - \{m\} + \{e\}$ as the new MST for G'. Suppose there is another MST T_2 has lower total weight than T' in G', then $W'(T_2) < W'(T')$.

- if $e \notin T_2$, the total weight of T_2 was not affected by the weight decrease on e, then $W(T_2) = W'(T_2) < W'(T') < W'(T) = W(T)$, T_2 was MST for G, **contradiction!**
- if $e \in T_2$, then the total weight of edges other than e in T_2 should be lower than the total weight of edges other than e in T', i.e., $W'(T_2 - \{e\}) < W'(T' - \{e\}) = W'(T - \{m\})$, since $e \notin T_2 - \{e\}$, and $e \notin T - \{m\}$, then we have

  $$W(T_2 - \{e\}) = W'(T_2 - \{e\}) < W'(T - \{m\}) = W(T - \{m\}).$$

  T_2 is a spanning tree, after removing $e = \langle u, v \rangle$, the vertices are partitioned into two subsets: $v \in V_1$ and $u \in V_2$. The path from v to u in T starts in V_1 ends in V_2, let l be the link connects V_1 and V_2, then $w_l \leq w_m$, (since m has the maximum weight on the path from v to u in T). $T_3 = T_2 - \{e\} + \{l\}$ is now a new spanning tree for G, and

  $$W(T_3) = W(T_2 - \{e\}) + w_l < W(T - \{m\}) + w_l \leq W(T - \{m\}) + w_m = W(T),$$

  which means T_3 should be MST for G, **contradiction!**