EL9343 Final Exam (2020 Fall)

Name: **ID:**

Exam Time: 9:00 AM -11:30 AM (US EST), Upload Deadline: 11:45 AM (US EST)

December 20, 2020

Write all answers on your own blank answer sheets. Scan and upload answer sheets to NYUclasses at the end of the exam, and 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, name your file with your NetID, Check your submission online before leaving the exam.

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

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

   (a) **T or F:** AVL tree rotation is required after each insertion or deletion of a node.

   (b) **T or F:** For a dense graph, memory usage for adjacency-list representation and adjacency matrix representation are on the same order.

   (c) **T or F:** DFS can be used to check if a directed graph has cycles.

   (d) **T or F:** For any Huffman code, if we exchange the codewords for the two least frequent symbols, the new code is still optimal.

   (e) **T or F:** For any optimization problem that can be solved with divide-and-conquer, there must be a dynamic programing based algorithm that solves the same problem.

   (f) **T or F:** In a directed acyclic graph, only vertices with in degree of zero can be the first vertex in the topological sorted order.

   (g) **T or F:** For a directed graph with a negative cycle, there can be no pair of vertices that has a well-defined shortest path cost.

   (h) **T or F:** Dynamic programming can be applied to solve the shortest path problem in a graph with negative weight edges.

2. (**4 points**) Which of the following statements are true?

   (a) Any P problem is also a NP problem;

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

   (c) NP-Complete problems are not in NP-Hard problem set;

(d) There is no known algorithm that can solve the 3-SAT problem in polynomial time;

(e) None of the above.

3. (**4 points**) Which of the following statements about Strongly Connected Components (SCCs) in a directed graph are true?

(a) In one SCC, there is always a path from any vertex to any other vertex;

(b) DFS, without modification, cannot find SCCs for any directed graph;

(c) During the second pass DFS in Kosaraju's algorithm, it works on a graph that the edges are reversed;

(d) Tarjan's algorithm is able to find SCC with just one pass of modified DFS;

(e) None of the above.

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

(a) DFS will visit all vertices, even if the graph is not fully connected;

(b) DFS can generate shortest pathes in a graph;

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

(d) In a directed graph, if there is an edge (u, v), then $f(u) > f(v)$;

(e) None of the above.

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

(a) [01, 11];

(b) [00, 01, 1];

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

(d) [1, 01, 000, 010];

(e) None of the above.

6. (**4 points**) Assuming all edge weights are distinct, which of the following statements about minimum spanning tree (MST) are true?

(a) Prim's algorithm can start with any vertex;

(b) The shortest path between a pair of vertices is always on the MST;

(c) For any given graph, the same MST will be returned by any MST algorithm;

(d) There can be more than $|V| - 1$ edges in the MST;

(e) None of the above.

7. (**4 points**) In the activity selection problem, if a new activity is added to the input activity set, which has the earliest finishing time among all activities. Compare with the solution of the original problem, which of the following are true?

(a) The maximum number of activities can be scheduled together will not reduce;

(b) This new activity is in at least one of the solutions to the new problem;

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

(d) The maximum number of activities can be scheduled together is increased by one;

(e) None of the above.

8. (**8 points**) When we build an AVL tree from scratch, keys are inserted in the order of:

$$10 \rightarrow 20 \rightarrow 30 \rightarrow 18 \rightarrow 15 \rightarrow 19,$$

Please plot the AVL tree after each key is inserted, and mark the type of rotation taken, if any, at each step.

See attached solutions at the end.

9. (**10 points**) For the DAG in Figure 1, the link weights are set as in the following table.

Link	AD	AF	BA	BE	DC	DF	EC	FE
Weight	2	3	-2	-10	2	-4	-8	-3

(a) (**5 points**) calculate the topological sort order of all vertices, (DFS considers the vertices in alphabetical order, and assume the adjacency list is ordered alphabetically), re-plot the DAG on your answer sheet so that all edges go from left to right;

(b) (**5 points**) calculate the shortest path distances from vertex A to all the other vertices, mark the shortest paths on your re-plotted DAG in question (a).

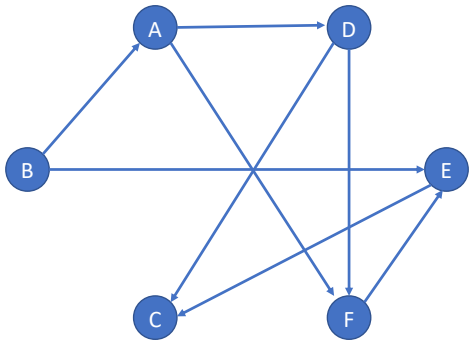


Figure 1: Directed Acyclic Graph for Question 9

See attached solutions at the end.

10. (**10 points**) Given a rod with integer length of n, you are asked to cut the rod into pieces so that the length of each piece is a positive integer, please find the optimal cutting strategy so that the product of the piece lengths are maximized. In other words, let l_i be the length for piece i, we want to cut the rod into k pieces so that $\sum_{i=1}^{k} l_i = n$ and $\prod_{i=1}^{k} l_i$ is maximized. The number of pieces k is NOT fixed, as long as $2 \leq k \leq n$.

(a) write down the pseudo code for getting the maximum product value, as well as printing the length of each piece in your optimal solution;

(b) analyze the complexity of your algorithm.

(*Hint: use dynamic programming similar to the rod-cutting problem studied in Lecture 9*).

Answer: Let $MaxProd[n]$ be the maximum product of the optimal cutting of a length-n rod (with $k \geq 2$ pieces), the recurrence can be write down as

$$MaxProd[n] = \max_{1 \leq i \leq n-1} (i * \max(MaxProd[n-i], n-i)),$$

where i is the length of the first piece of the cut.

The pseudo code is illustrated in Figure 2.

```
Max_prod(n)
P[1…n]: max_prod of length n with at least two pieces
S[1…n]: first piece length of the optimal solution for length n

P[1]=P[2]=S[1]=S[2]=1;
For i=3 to n
        q=-infinity;
        for j=1 to i-1
            if q < j*max(P[i-j], i-j)
                then q= j*max(P[i-j], i-j); S[i]=j;
        P[i]=q;

Return P and S


Print_Max_prod_solution(n)
(P,S)=max_prod(n);
while n >=2
        print S[n];
        if P[n]==S[n]*(n-S[n])
                then print n-S[n]; break;
                else n=n-S[n];
```

Figure 2: Pseudo Code for Question 10

The time complexity is $\Theta(n^2)$.

++++++++++++ **Continue on the next page** ++++++++++++

11. (**10 points**) For the activity selection problem studied in Lecture 10, given the start and finish time of all the activities,

(a) (**4 points**) construct another greedy algorithm to select the maximal number of compatible activities so that the activity with the latest start time will always be selected;

(b) (**6 points**) prove that your greedy algorithm is optimal by showing its greedy choice property and its optimal substructure property.

Answer: (a) The new greedy algorithm is the dual of the earliest finish time algorithm learnt in Lecture 10, it is called *the latest start time* algorithm:

(a) Sort the activities by start time in decreasing order

(b) Select the first activity

(c) Then schedule the next activity in sorted list which finishes before the previously selected activity starts

(d) Repeat until no more activity left

It first selects the activity with the latest start time, remove all the other activities have conflict/overlap with it, repeat the same greedy choice on the remaining activities, until there is no more activity left. *If you flip the time axis horizontally, solve it using the earliest finish time algorithm on the flipped problem, the set of activities selected is exactly the same as applying the latest start time algorithm on the original problem.*

b) *Greedy choice property:* If S is an optimal solution, and the activity a with the latest start time is not in S, replace the last activity b in S with a, a should not have any conflict with the other activities in S, since its start time is later than the start time of b, then $S' = S - \{b\} + \{a\}$ is also an optimal solution.

Optimal substructure property: If $S = S_1 + \{a\}$ is an optimal solution, then S_1 must be an optimal solution for the sub-problem of selecting the maximum number of activities out of all the activities that finish before a starts. Otherwise, if S_2 with $|S_2| > |S_1|$ is a better solution for the subproblem, obviously $S' = S_2 + \{a\}$ is a better solution for the original problem. Contradiction!

12. (**14 points**) In a weighted directed graph G, the width of a path is the minimum weight of all links along the path. For example, for the path $A \to B \to C \to D$, if the link weights are $AB = 5$, $BC = 3$, $CD = 4$, the path width is 3. A *widest path* from a node s to another node v is a path with the largest width among all the possible paths from s to v.

(a) (**4 points**) Prove that at lease one widest path for each pair is loop-free, i.e., does not contain a cycle;

(b) (**10 points**) Develop a dynamic programming based algorithm to find the widest path from a source node s to each of the other nodes in G. Write down the detailed pseudo code of your algorithm, analyze its complexity. (*Hint: use a similar procedure as Bellman-Ford's algorithm, you don't have to formally prove its correctness*)

Answer: (a) If a widest path has loop, since the path width is the minimum weight of all edges on the path, if we remove the cycle from the path, the width of the resulted simple

path will be larger than or equal to the widest path with loop, therefore, there is at least one loop-free widest path.

(b) Observe that the width of a path from s to v through u is the minimum of the path width from s to u and $w(u, v)$. Let $W(u)$ be the width of the widest path from s to u, then we have $W(v) \geq \min\{W(u), w(u, v)\}$, and

$$W(v) = \max_{u:\langle u,v\rangle \in G} \min\{W(u), w(u, v)\}.$$

Due to (a), we can guarantee to find a widest path for any pair with at most $|V| - 1$ hops. Then the problem can be solved using a Bellman-Ford like algorithm.

Step 0: Initialize $W(s) = \infty$, and $W(v) = -\infty$, $\forall v \neq s$;

Step 1: Take $|V| - 1$ iterations

Step 2: at each iteration, for each vertex u, update the width estimate of each of its neighbor v as follows:

if $\min(W(u), w(u, v)) > W(v)$, update $W(v) = \min(W(u), w(u, v))$, and set $\pi(v) = u$

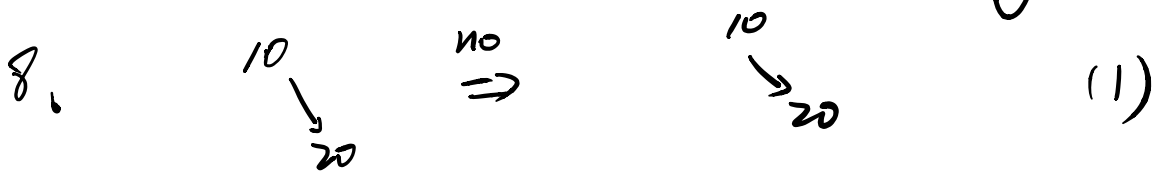
Step 3: Go back to step 1.

Step 4: return $W(v)$ and $\pi(v)$ $\forall v \in V$.

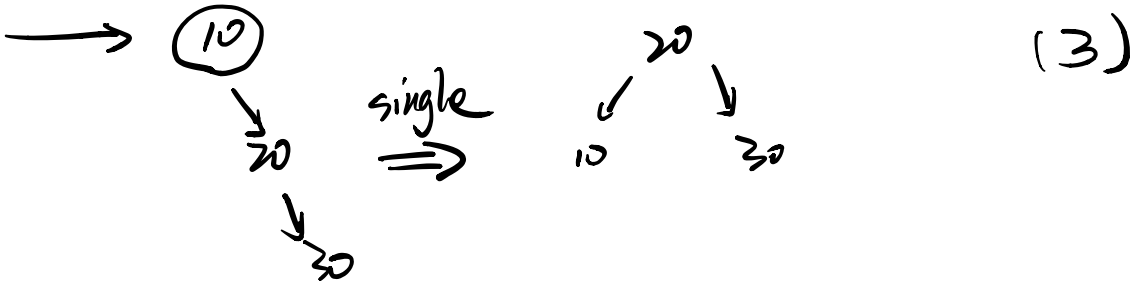
The complexity is $\Theta(|V||E|)$.

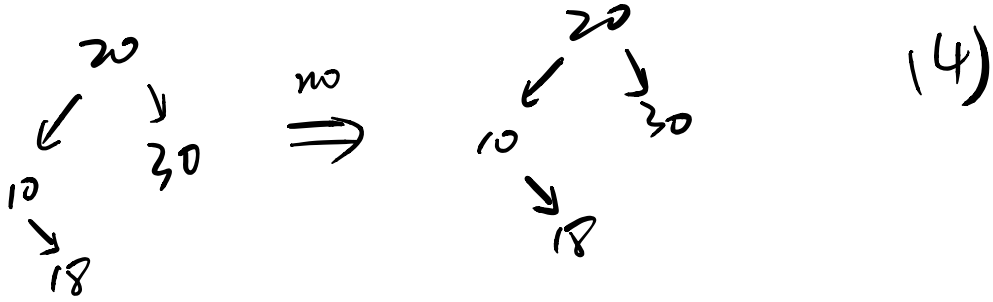
++++++++++++ **End of Final Exam** ++++++++++++

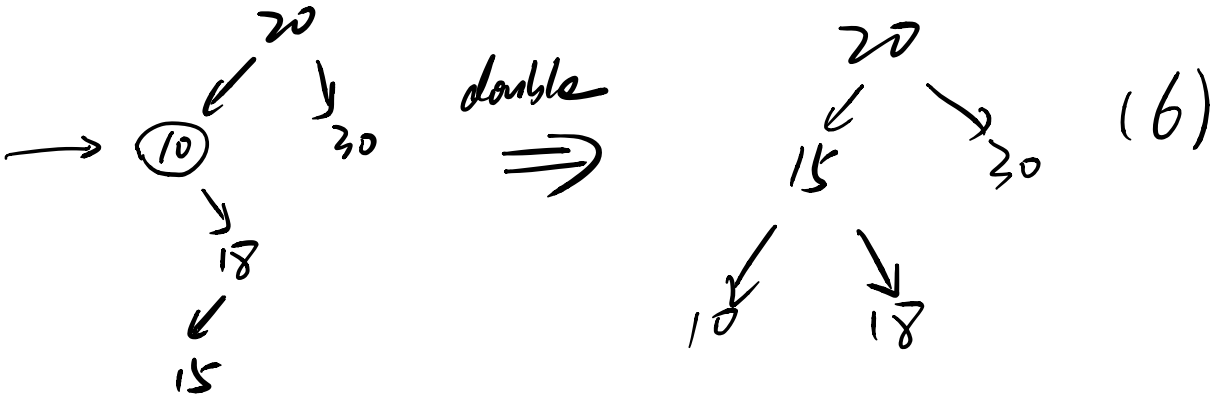
grade

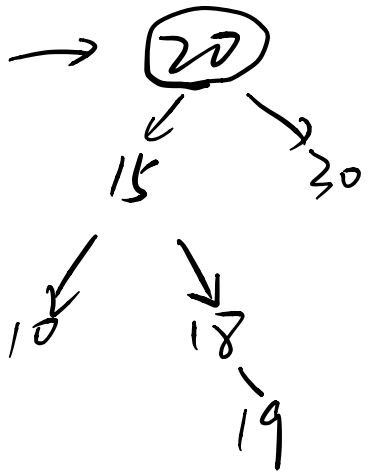


The nearest imbalance

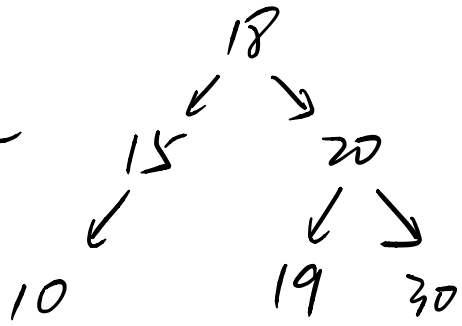








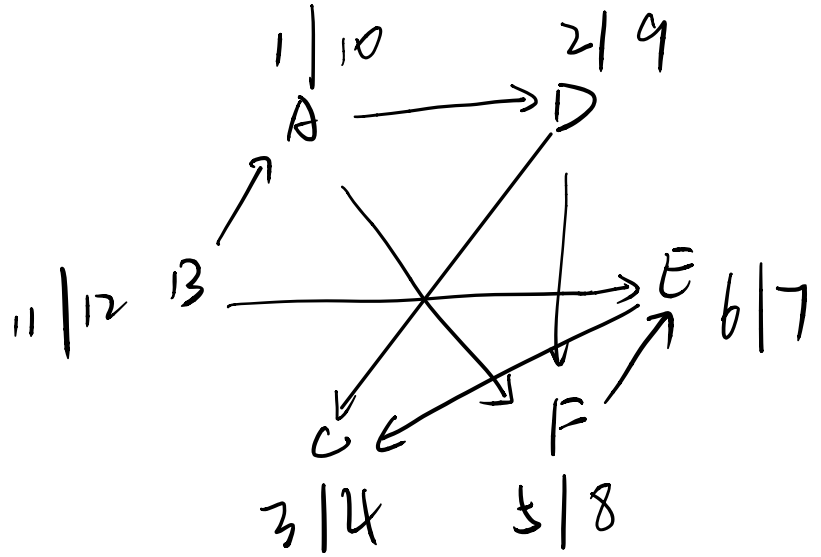
double ⇒



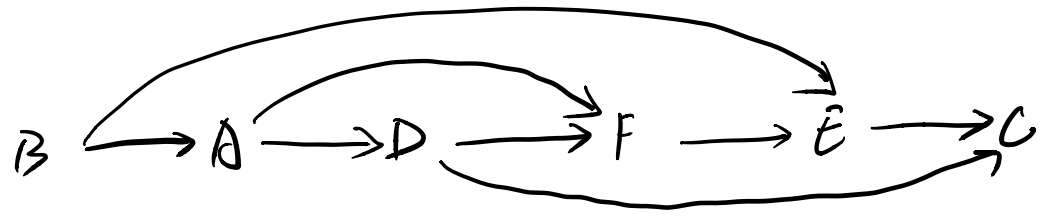
(8)

9. (a) DFS:

grade



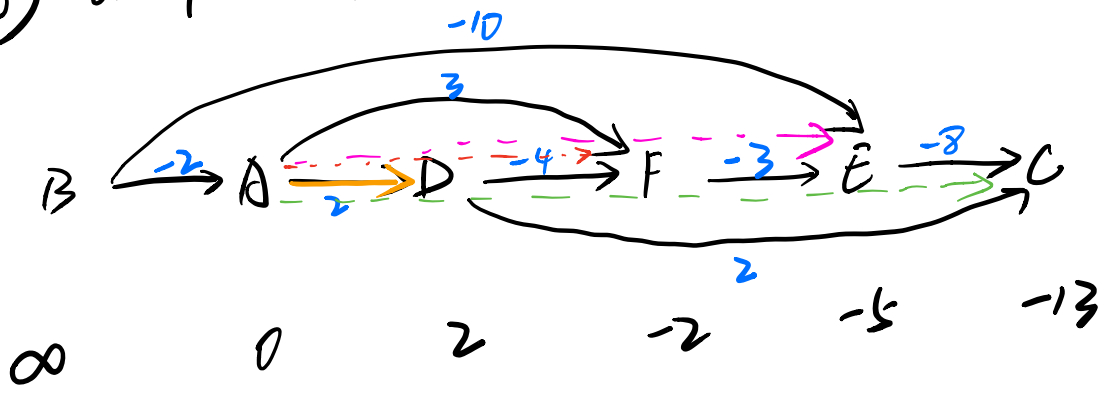
(3)



11|12 1|10 2|9 5|8 6|7 3|4

(4)

(b) one pass Bellman-Ford



A → D : 2

A → F : A → D → F 2 - 4 = -2

A → E : A → D → F → E 2 - 4 - 3 = -5

A → C : A → D → F → E → C 2 - 4 - 3 - 8 = -13

A → B : ∞