Analysis of Algorithms Comprehensive Exam (*Five Questions*) Points 100
Fall 2015

Q1. Write a sentence or two explaining your answers for each question below.

a. Dynamic Programming algorithm trades off time for reducing space complexity. True or False?

b. Within a greedy algorithm, the minimum of a list of numbers is picked up in each iteration. However, some numbers also get changed within the iteration. What type of data structure may be used for storing the numbers.

c. In order to prove a problem X to be NP-complete one needs to develop a polynomial transformation from a problem Y. What type of problem should Y be?

d. An algorithm's time-complexity is $O(n^3)$ for input size n. What can you comment about its space complexity?

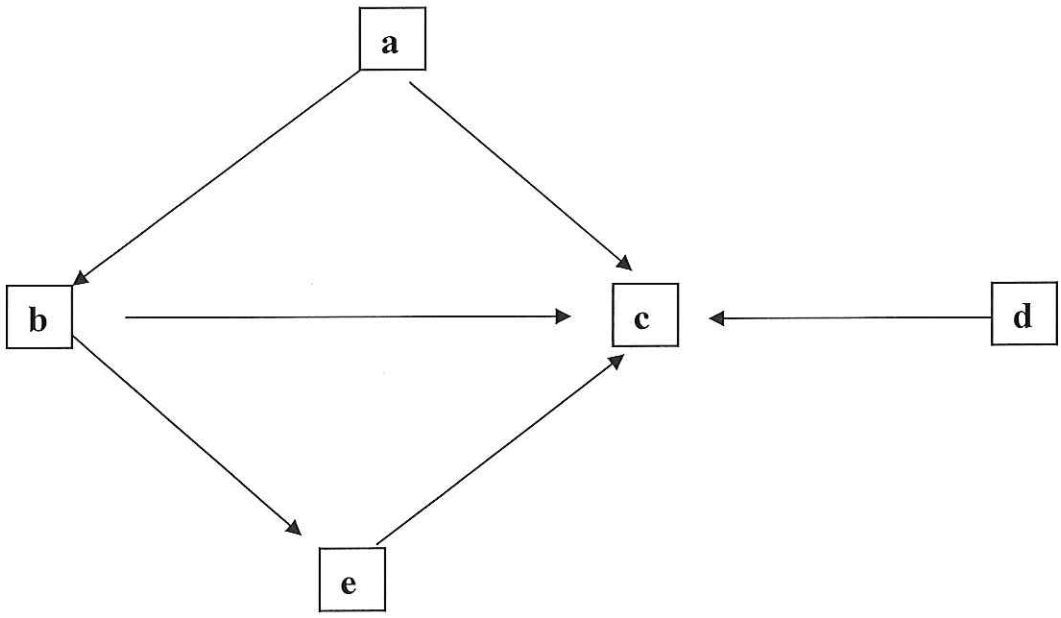
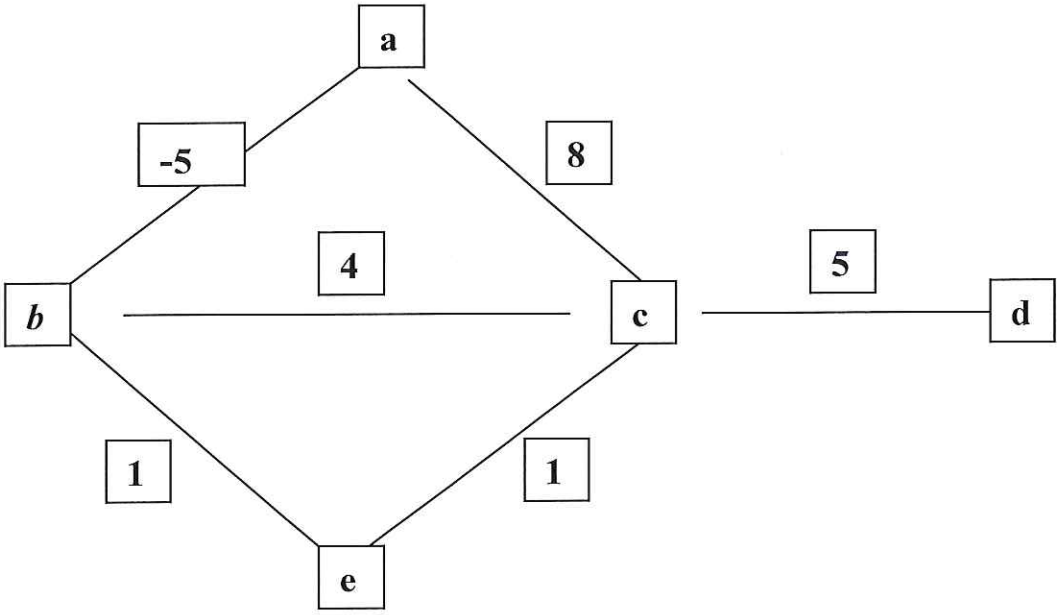
e. Is 5-SAT problem outside the P-class?

Q2. Run the *Huffman encoding* greedy algorithm on the following example. Intermediate trees should be shown. Input list of *node*(*frequency*) is below.
$\{V_1(5), V_2(3), V_3(7), V_4(6), V_5(6), V_6(2)\}$.

Q3. *0-1 Knapsack backtracking* algorithm (without any bounding function used) may have efficient pruning by weight if the input objects are pre-sorted. Describe briefly such an ordering and apply that to the following problem instance showing your backtrack tree. {O1($5, 3Kg), O2($3, 30Kg), O3($2, 12Kg), O4($8, 56Kg)}, Knapsack limit = 32kg.

Explain the asymptotic time-complexity of your algorithm in terms of the number of objects *n*.

Q4. Run the *topological sort* algorithm on the following directed graph. Show steps in each iteration. Measure the time-complexity of the algorithm over this example (as an integer number of steps) and explain how do you get that number.



Q5. Djikstra: Find the *shortest path* length from node b to **all** other nodes for the following undirected and weighted graph, using *Djikstra's* algorithm <u>with a queue</u> to store the candidate nodes to be used for updating others. Numbering each of your iterations, show the content of the queue and the shortest distance value of each node from node b within each iteration.