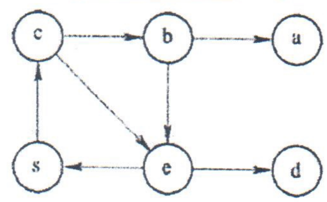
Algorithms
Comprehensive Exam
(Fall 2020)

<u>SHORT QUESTIONS</u> **(Answer all six questions, each carrying 7 points.)**

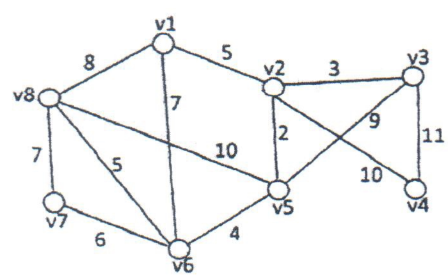
1. Solve the following recurrences to get their runtime <u>upper bounds</u>:

$$T(n) = T(n-2) + n$$
$$T(n) = T\left(\frac{n}{2}\right) + n$$
$$T(n) = T(n-2) + T\left(\frac{n}{2}\right) + n$$

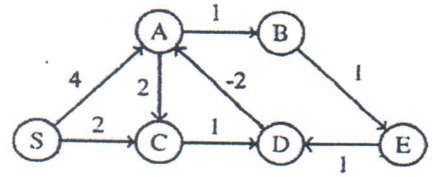
2. Give the visited vertex order of the following directed graph, starting with vertex s, under (1) <u>breadth-first search</u> and (2) <u>depth-first search</u>, respectively.



3. Follow <u>Prim's Algorithm</u> and <u>Kruskal's Algorithm</u>, respectively to derive the <u>minimum spanning tree</u> (MST) of the following graph. For every algorithm, write down the edge picked in each step. For example, Step 1: (v1, v2).



4. Please use <u>Johnson's Algorithm</u> to reweight the directed graph shown below such that all edge weights are <u>non-negative</u>.

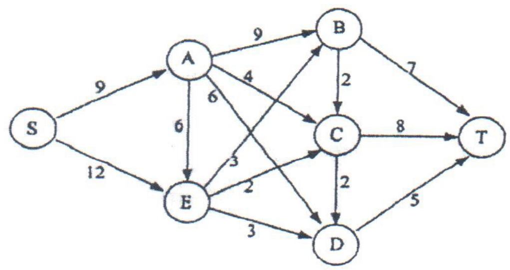


5. The utilization efficiency of a hash table depends heavily on its hashing function(s) employed. Describe with a diagram to illustrate how a multiplication method of hashing works on a machine with the word size of w bits for a hash table with 2^p entries, $p < w$.

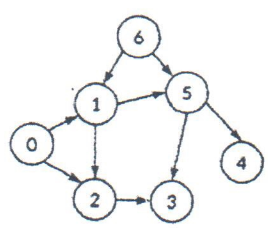
6. Show your construction of an optimal Huffman code for the set of 12 frequencies: **a**:2 **b**:6 **c**:5 **d**:8 **e**:13 **f**:21 **g**:34 **h**:15 **i**:27 **j**: 9 **k**:48 **l**: 26.

LONG QUESTIONS **(Answer all four questions, each carrying 15 points.)**

1. Follow the Ford-Fulkerson Algorithm to compute the max flow of the flow network given below. Illustrate the corresponding residual network at each step of the Ford-Fulkerson Algorithm.
   Also, compute the min-cut of the flow network, with each step shown.



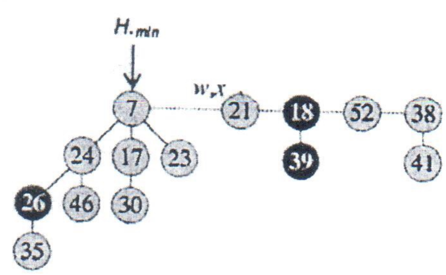
2. The following graph denotes a sequence of courses, with their dependencies on other courses shown. A directed edge from vertex u in the graph to vertex v indicates that course u must be taken before course v may start. Please apply DFS (hint: topological sort) to find an ordering of these courses that conforms to the shown dependencies. Illustrate each step.



3. An <u>optimal binary search tree</u> (OBST) for a given set of keys with known access probabilities ensures the minimum expected search cost for key accesses. Given the set of four keys with their access probabilities of $k_1 = 0.16$, $k_2 = 0.13$, $k_3 = 0.2$, $k_4 = 0.08$, respectively, and five non-existing probabilities of $d_0 = 0.12$, $d_1 = 0.1$, $d_2 = 0.06$, $d_3 = 0.11$, $d_4 = 0.04$, <u>construct OBST</u> following dynamic programming with memoization for the given four keys and <u>demonstrate</u> the constructed OBST, which contains all four keys (k_1, k_2, k_3, k_4) and five non-existing dummies (d_0, d_1, d_2, d_3, d_4). (Show your work using the three tables, for expected costs: $e[i, j]$, access weights: $w[i, j]$, and $root[i, j]$, with i in $e[i, j]$ and $w[i, j]$ ranging from 1 to 5, j in $e[i, j]$ and $w[i, j]$ ranging from 0 to 4, and both i and j in $root[i, j]$ ranging from 1 to 4).

```
OPTIMAL-BST(p, q, n)
let e[1..n+1, 0..n], w[1..n+1, 0..n],
        and root[1..n, 1..n] be new tables
for i = 1 to n + 1
    e[i, i - 1] = q_{i-1}
    w[i, i - 1] = q_{i-1}
for l = 1 to n
    for i = 1 to n - l + 1
        j = i + l - 1
        e[i, j] = ∞
        w[i, j] = w[i, j - 1] + p_j + q_j
        for r = i to j
            t = e[i, r - 1] + e[r + 1, j] + w[i, j]
            if t < e[i, j]
                e[i, j] = t
                root[i, j] = r
return e and root
```

4. A Fibonacci min-heap (H) relies on the procedure of CONSOLIDATE to merge trees in the root list upon the operation of extracting the minimum node. For H below, show the <u>final result</u> after extracting H.min. (Also, give the result of every key consolidation step till its completion.)



Good Luck!