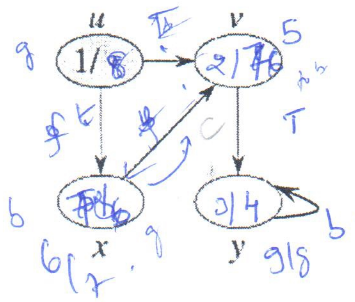
CSCE 500 Final Exam

12/10/2015

1. Depth First Search (DFS) colors every vertex of a given graph in white, gray, and black during the process to build a search tree, with a discovery time and a finish time kept at each vertex. All edges in a directed graph $G = (V, E)$ under DFS are classified into four types: tree, back, forward, and cross edges. For G given below, conduct DFS that starts from vertex u at time clock = 1 to
   (a) mark the discovery time and the finish time of every vertex, and
   (b) classify the types of all edges. (Note: one intermediate result should be shown upon each edge classification for clarity. **10%** in total)



2. The Floyd-Warshall algorithm (FW) obtains all pairs of shortest paths efficiently in a weighted directed graph, with its pseudo code listed below.
   (a) Fill in the missing statement in the code. **(2%)**
   (b) Consider the following graph, with its vertexes labelled 2, 3, and 4. Derive all distance matrices $D^{(k)}$ following FW so that the $d_{ij}^{(n)}$ element of final matrix $D^{(n)}$ denotes $\delta(i, j)$ for every vertex pair $<i, j>$. **(10%)**

FLOYD-WARSHALL(W, n)

$D^{(0)} = W$

for $k = 1$ **to** n

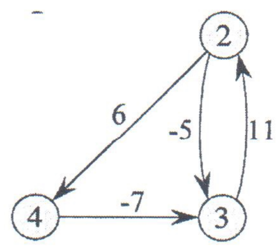
  let $D^{(k)} = (d_{ij}^{(k)})$ be a new $n \times n$ matrix

  for $i = 1$ **to** n

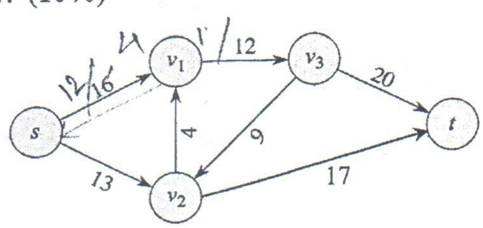
   for $j = 1$ **to** n

    $\boxed{d_{ij}^{(k)} = min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}}$

return $D^{(n)}$



3. The Edmonds-Karp algorithm (EK) follows the basic Ford-Fulkerson method with breadth-first search to choose the shortest augmenting path (in terms of the number of edges involved) following the Ford-Fulkerson method to compute the maximum flow iteratively from vertex s to vertex t in a weighted directed graph. Illustrate the maximum flow computation process (including the augmenting path chosen in each iteration and its resulting residual network) via EK for the graph depicted below. **(10%)**



Scanned with CamScanner

4. The NP-complete class contains those NP problems which are *hardest among all.*
   (a) How do we prove the very first NP-complete problem? **(3%)**
   (b) After NP-complete problems are proven, how do we show a new problem at hand to be NP-complete? **(2%)**

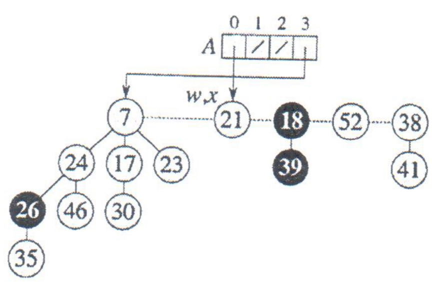
   The traveling-salesman problem is an NP-complete problem, and it has a 2-approximation solution in polynomial time based on establishing a minimum spanning tree (MST) rooted at the start/end vertex (in polynomial time following MST-PRIM), provided that the graph edge weights observe triangle inequality. Sketch a brief proof to demonstrate that such a solution satisfies 2-approximation. **(10%)**

5. Show your construction of an optimal Huffman code for the set of 7 frequencies: **a**:2 **b**:6 **c**:8 **d**:11 **e**:14 **f**:17 **g**:34. **(6%)**

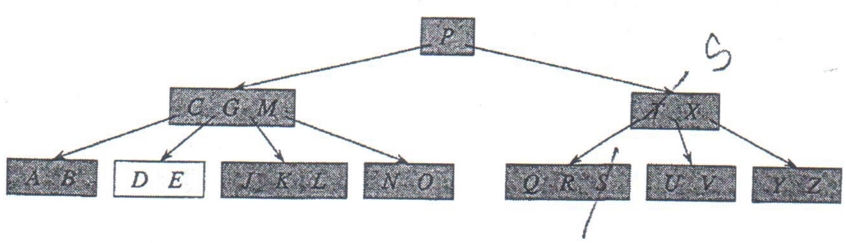
6. Solve the recurrence of $T(n) = T(n/2) + T(n/4) + n$. **(8%)**

7. The utilization efficiency of a hash table depends heavily on its hash 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, $p < w$. **(7%)**

8. A Fibonacci min-heap relies on the procedure of CONSOLIDATE to merge trees in the root list upon the operation of extracting the minimum node. Given the following partially consolidated diagram, show every subsequent consolidation step till its completion. **(10%)**



9. Given a B-tree with the minimum degree of $t = 3$ below, show the results after
   (a) deleting T and
   (b) then followed by deleting U. **(8% in total)**



Scanned with CamScanner

10. Consider the matrix-chain multiplication problem for four matrices A_1, A_2, A_3, A_4, with their sizes being 30×40, 40×20, 20×10, and 10×80, respectively. Follow the tabular, bottom-up method in the procedure of MATRIX-CHAIN-ORDER below to construct tables that keep respectively entry $m[i,j]$ for all $1 \le i, j \le 4$ and entry $s[i, j]$ for $1 \le i \le 3$ and $2 \le j \le 4$ to get the optimal parenthesized multiplication result.

(a) Construct the table, with its entry values shown. **(12%)**

(b) Show the parenthesized multiplication of the matrix-chain. **(2%)**

```
MATRIX-CHAIN-ORDER(p)
n = p.length − 1
let m[1..n, 1..n] and s[1..n − 1, 2..n] be new tables
for i = 1 to n
    m[i, i] = 0
for l = 2 to n                  // l is the chain length
    for i = 1 to n − l + 1
        j = i + l − 1
        m[i, j] = ∞
        for k = i to j − 1
            q = m[i, k] + m[k + 1, j] + p_{i−1} p_k p_j
            if q < m[i, j]
                m[i, j] = q
                s[i, j] = k
return m and s
```

Good Luck!

$p_0 = 30$

$p_1 = 40$

$p_2 = 20$

$p_3 = 10$

$p_4 = 80$.