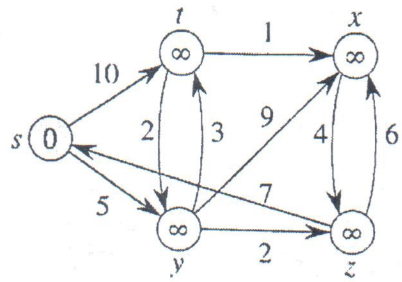
1. The Bellman-Ford algorithm (*BF*) solves the single-source shortest-path problem in a weighted directed graph $G = (V, E)$. Given the graph G below, follow *BF* to find shortest paths from vertex s to all other vertexes, with all predecessor edges shaded and estimated distance values from s to all vertexes provided at the end of each iteration. (8%)

   How many iterations are involved in *BF* for a general graph $G = (V, E)$? (2%)



10

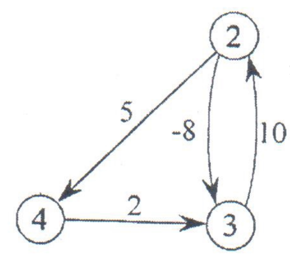
2. The Floyd-Warshall algorithm (*FW*) obtains all pairs of shortest paths in a weighted directed graph, with its pseudo code listed below. Fill in the missing statement in the code. (2%)

   Consider the following graph, with its vertexes labelled 2, 3, and 4. Derive all distance matrices $D^{(k)}$ so that the $d_{i,j}^{(n)}$ element of final matrix $D^{(n)}$ denotes $\delta(i, j)$ for every vertex pair $<i, j>$. (8%)

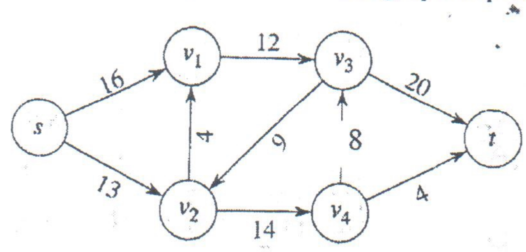
FLOYD-WARSHALL(W, n)

$D^{(0)} = W$
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
      $d_{ij}^{(k)} =$
return $D^{(n)}$

8



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) for computing 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%)



10

-1-

4. The traveling-salesman problem (TSP) belongs to NP-completeness, whose proof in general consists of two components. What are the two proof components to show a problem to be NP-complete? (3%)

TSP 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), if the graph edge weights observe triangle inequality. Sketch a brief proof to demonstrate that such a solution satisfies 2-approximation. (7%)

3+7

5. Solve the recurrence of $T(n) = 2 \cdot T(\sqrt{n}) + \lg n$. (6%)

6

6. Basic quicksort relies on the PARTITION routine, which fragments array elements into two parts according to the pivot element, say, the rightmost element. Let i and j denote respectively the rightmost element of the left part and the element to be examined in the course of partitioning. Illustrate the results of partitioning for $1 \le j \le 4$ and the final partitioned result under array elements of <13, 8, 15, 4, 10>. (6%)

6

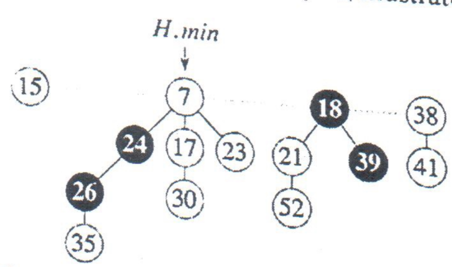
7. Selecting the i^{th} element in a set of n numbers can be done in $O(n)$ in the worst case. Such a selection algorithm can be the basis of an efficient procedure for obtaining the sorted j largest numbers in the set, where the procedure comprises only three algorithms (including the selection algorithm). Describe the procedure and list its time complexity. (8%)

8. 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%) ✓

Briefly state how a hash function can be employed for de-duplication in data archival. (3%)

9. The Fibonacci heap exhibits O(1) for a key decrease. Given the Fibonacci heap below, show the resulting heap after key of 35 is decreased down to 6. (7%; illustrate every step involved)

(b)



i pj r
13 8 15 4 10

pi j r
$\boxed{13}$ 8 15 4 10

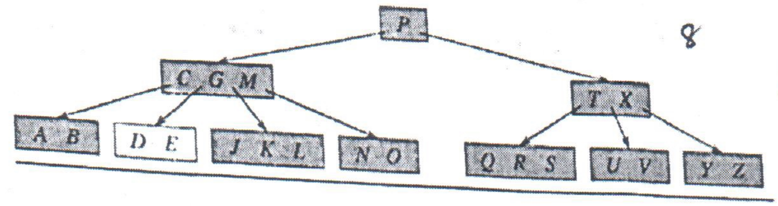
pi j r
8 $\boxed{13}$ 15 4 10

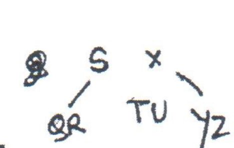
pi j r
8 $\boxed{13\ 15}$ 4 10

p i r
$\boxed{8\ 4}$ $\boxed{15\ 13}$ 10

p i
$\boxed{8\ 4}$ 10 $\boxed{13\ 15}$

10. Given a B-tree with the minimum degree of $t = 3$ below, show the results of (i) deleting M and (ii) then followed by deleting V. (8%)





11. An optimal binary search tree (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, k_i, $1 \leq i \leq 4$, with their access probabilities of $k_1 = 0.25$, $k_2 = 0.15$, $k_3 = 0.1$, $k_4 = 0.2$, respectively, and five non-existing probabilities of $d_0 = 0.1$, $d_1 = 0.05$, $d_2 = 0.04$, $d_3 = 0.06$, $d_4 = 0.05$, construct OBST following dynamic programming for the given four keys. (15%; show your work using the three tables, for expected costs: $c[i, j]$, access weights: $w[i, j]$, and $root[i, j]$).

Good Luck!