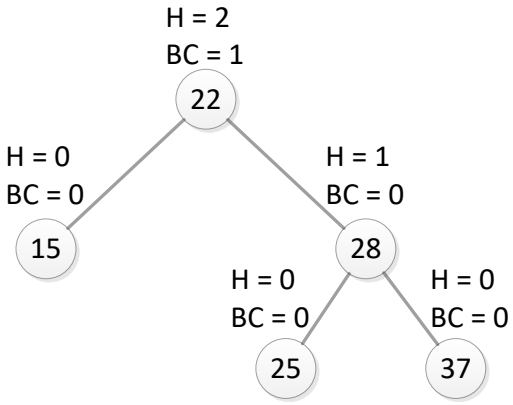
Design and Analysis of Computer Algorithms

Homework #6: Students do not need to submit the solution. It is for practicing purpose for the final exam!

Problem #1.

Consider the following BST, insert numbers 10, 40, 50, and 30 into the BST in the mentioned order. For each insertion, balance the tree using AVL technique, and show height (H) and balance factor of each node in each step.



Problem #2 Find an optimal parenthesization of a matrix-chain product whose sequence of dimensions is <5; 10; 3; 12; 5; 50; 6>

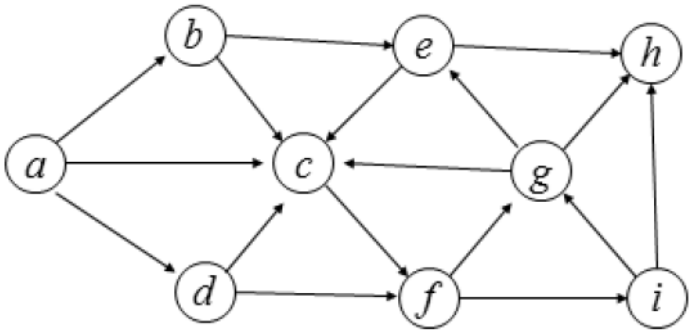
Problem #3: Given a set of frequencies based on 8 numbers.

a:1, b: 1, c: 2, d: 3, e: 5, f: 8, g: 13, h: 21

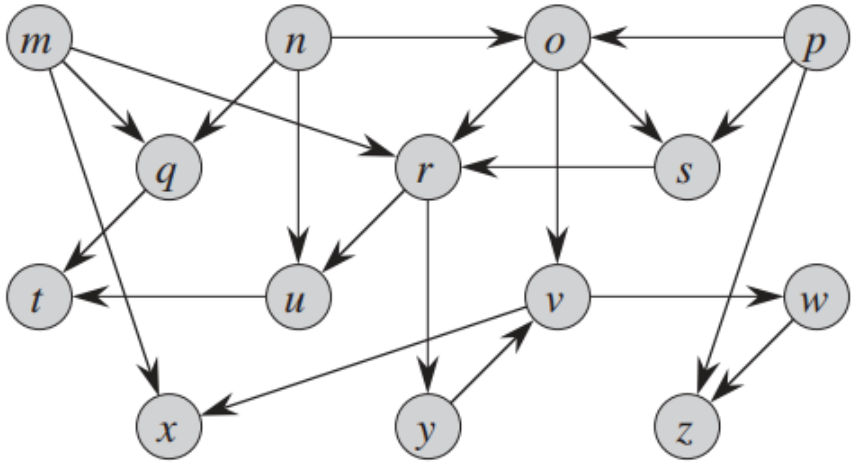
What is an optimal Huffman code for the set?

Problem #4: Show the ordering of vertices produced by TOPOLOGICAL-SORT when it is run on the following graph, assume that DFS considers the vertices in alphabetical order.

a)



b)



Hint:

a) If the graph has circles, the topological ordering does not exist.

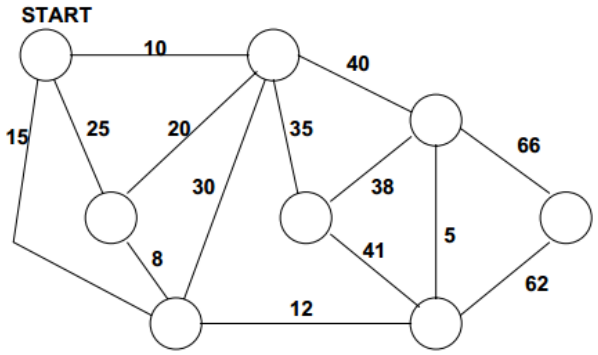
b)

Our start and finish times from performing the DFS are

label	d	f
m	1	20
q	2	5
t	3	4
r	6	19
u	7	8
y	9	18
v	10	17
w	11	14
z	12	13
x	15	16
n	21	26
o	22	25
s	23	24
p	27	28

And so, by reading off the entries in decreasing order of finish time, we have the sequence $p, n, o, s, m, r, y, v, x, w, z, u, q, t$.

Problem #5. Illustrate the execution of Prim's algorithm to find the minimum spanning tree for the graph below. At each step, you should show the vertex and the edge added to the tree and the **key** values of each node. Use START vertex as the first vertex in your traversal.



Problem #6. Given an adjacency matrix representation of a graph.

How long does it take to compute the in-degree and out-degree of all vertices?

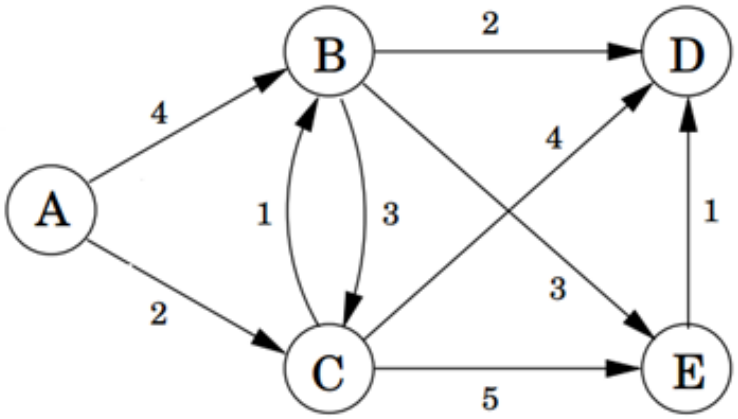
Hint:

In-degree: Each vertex requires to scan a column, so O(n) for each vertex and $O(n^2)$ overall. Out-degree: Same, but need to scan a row, $O(n^2)$.

Problem #7. Run Dijkstra's algorithm on the following directed graph, starting at vertex A.

a) Show the associated distance from vertex A to every vertex in each step

b) Show the final shortest-path tree?



Problem #8. Suppose you are given a diagram of a telephone network and two switching centers a and b. The diagram is a graph G whose

- vertices represent switching centers,
- edges represent communications lines between two centers
- The edges are marked by their bandwidth.
- The bandwidth of a path is the bandwidth of its lowest bandwidth edge.

Describe an algorithm that calculate the maximum bandwidth of a path between a and b. (Just report the maximum bandwidth; you do not have to give the actual path)

Hint:

This can be accomplished by modifying Dijkstra's algorithm. Instead of representing the shortest path from a to u, the label D[u] represents the maximum bandwidth of any path from a to u. The maximum bandwidth for path from a through u to a vertex z adjacent to u is min{D[u], w((u; z))} so that the relaxation step updates D[z] to max{D[z],min{D[u], w((u, z))}}.

Problem #9 A directed graph G = (V; E) is singly connected if G contains at most one simple path from u to v for all vertices$u, v \in V$. Give an efficient algorithm to determine whether or not a directed graph is singly connected.

Hint:

This can be done in time $O(|V||E|)$. To do this, first perform a topological sort of the vertices. Then, we will contain for each vertex a list of it's ancestors with in degree 0. We compute these lists for each vertex in the order starting from the earlier ones topologically. Then, if we ever have a vertex that has the same degree 0 vertex appearing in the lists of two of its immediate parents, we know that the graph is not singly connected. however, if at each step we have that at each step all of the parents have disjoint sets of degree 0 vertices as ancestors, the graph is singly connected. Since, for each vertex, the amount of time required is bounded by the number of vertices times the in degree of the particular vertex, the total runtime is bounded by $O(|V||E|)$.

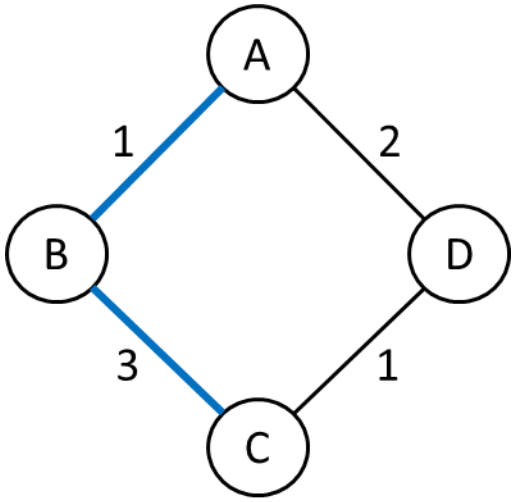
Problem #10. A student proposes a greedy strategy to find a shortest path from vertex $start$ to vertex $goal$ in a given connected graph:

1: Initialize *path* to *start*.
2: Initialize *VisitedVertices* to $\{start\}$.
3: If *start*=*goal*, return *path* and exit. Otherwise, continue.
4: Find the edge (*start*,*v*) of minimum weight such that *v* is adjacent to *start* and *v* is not in *VisitedVertices*.
5: Add *v* to *path*.
6: Add *v* to *VisitedVertices*.
7: Set *start* equal to *v* and go to step 3.

Justify if this greedy strategy always finds a shortest path from $start$ to $goal$, if not, give a counterexample.

Solution:

The greedy algorithm presented in this problem is not guaranteed to find the shortest path between vertices in graph. This can be seen by counterexample with $start = A$ and $goal = C$:



The path found by the greedy strategy is A → B → C (distance = 4) but the shortest path is A → D → C (distance = 3).

Problem #11.

In the bin packing problem, we need to pack n items into bins so that number of bins used is minimized. Assume that each bin has capacity C and size of each item is smaller than the bin capacity.

We are using an approximation algorithm to solve the problem. For each item, the algorithm tries to pack it into the last bin used. If the item size exceeds empty space of the current bin, it will be put into a new bin. Determine approximation ratio of this algorithm.

Solution:

Approximation ratio is 2

- Let S denote sum of all item sizes.
- Let m^* denote the number of bins used by an optimal solution, then:

$$m^* \geq \frac{S}{C} = R \qquad (1)$$

- Let $B_1, B_2, \dots, B_m$ denote m items used by the approximation algorithm and $s_1, s_2, \dots, s_m$ denote sum of item sizes in each bin
- Sum of item sizes of two consecutive bins B_i and B_{i+1} is greater than bin capacity C (otherwise, we can store all items of B_{i+1} in B_i) which means

$$s_i + s_{i+1} > C$$

So:

$$(s_1 + s_2) + (s_3 + s_4) + \cdots + s_m = S \geq C\frac{m}{2} \qquad (2)$$

From (1) and (2):

$$m \leq 2\frac{S}{C} \leq 2R \leq 2m^*$$