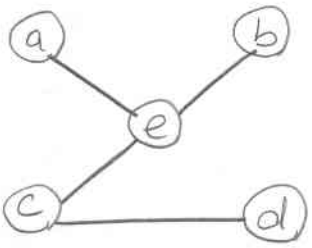
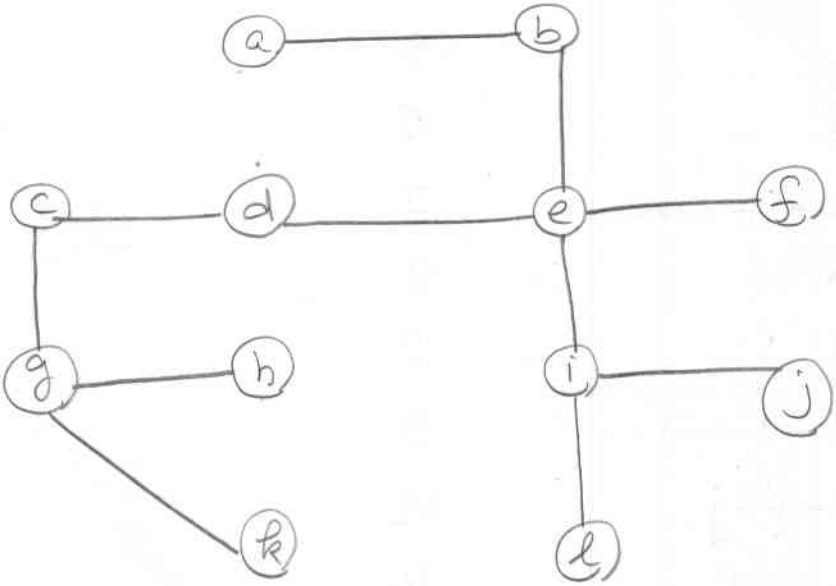
Question 7 in 9.1

(a) The MST consists of the edges (a,e), (e,b), (e,c), and (c,d)



(b) The MST consists of the edges:

(a,b)
(b,e)
(e,d)
(d,c)
(e,f)
(e,i)
(i,j)
(c,g)
(g,h)
(i,l)
(g,k)



Question 9 in 9.1

(a) Set edge weights to 1 (equal weights).

(b) No, applying BFS or DFS is conceptually more suitable and simpler and can be implemented more efficiently.

Question 2 in 9.3

(a) The Shortest-paths and their lengths are :

from a to d : a - d of length 7
" a to b : a - d - b " " 9
" a to c : a - d - c " " 12
" a to e : a - d - c - e " " 18

(b) Shortest paths from a to other vertices are

a - b ——— of length 3
a - d ——— 4
a - c ——— 5
a - d - e ——— 5
a - d - e - f ——— 7
a - d - h ——— 9
a - c - g ——— 9
a - d - e - i ——— 9
a - d - e - f - j ——— 12
a - d - e - i - l ——— 14
a - c - g - k ——— 15

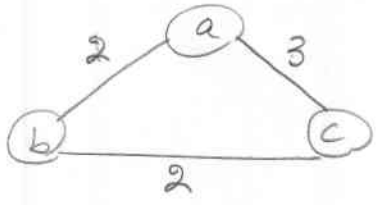
Question 4 in 9.3

(a) True : On each iteration, an edge connecting a non-tree vertex to a a tree vertex (for tree constructed thus far). So, a tree is obtained, which after the last iteration, includes all the vertices in the graph. Hence, it is a spanning tree.

(b) False : Here is a counter example.

MST : (a, b) (b, c)

Shortest paths : a - b and a - c .



Question # 1 in 9.4

(a)

Char	A	B	C	D	-
codeword	0	100	111	101	110

(b) 01000111010001 01

(c)

100	0	101	110	0	101	0
B	A	D	-	A	D	A

Question # 3 in 9.4

(a) Yes, assuming that the number of least frequent chars doesn't exceed two. On the first iteration, the least two frequent chars are combined as left and right subtrees of a new tree. Hence they are at the same level and remain so after each iteration.

(b) Yes. This can be proved by contradiction. Assume that there exists a Huffman code containing two chars. c_i and c_j such that $p(c_i) > p(c_j)$ and $l(w(c_i)) > l(w(c_j))$, where $p(c_i)$ and $l(w(c_i))$ are the probability and codeword's length of c_i, respectively. If we swap the codwords of c_i & c_j and leave the codewords for all other chars the same, we get a new prefix code with expected length $\sum_{k=1}^{n} l(w(c_k))\,p(c_k)$ that is smaller than the initial code. This contradicts the optimality of initial Huffman code. Huffman's encoding algorithm minimizes the sum $\sum_{k=1}^{n} l(w(c_k))\,p(c_k)$