P. 290

2. The application of the dynamic programming algorithm to the input 5, 1, 2, 10, 6, 2 in section 8.1 yielded the following table:

index	0	1	2	3	4	5	6
C		5	1	2	10	6	2
F	0	5	5	7	15	15	17

Using the data in the first six columns, we conclude that the largest amount of money that can be obtained for the input 5, 1, 2, 10, 6 is $F(5) = 15$, which is obtained by taking coins $c_4 = 10$ and $c_1 = 5$.

P. 296

1. a.

	i	capacity j: 0	1	2	3	4	5	6
	0	0	0	0	0	0	0	0
$w_1 = 3, v_1 = 25$	1	0	0	0	25	25	25	25
$w_2 = 2, v_2 = 20$	2	0	0	20	25	25	45	45
$w_3 = 1, v_3 = 15$	3	0	15	20	35	40	45	60
$w_4 = 4, v_4 = 40$	4	0	15	20	35	40	55	60
$w_5 = 5, v_5 = 50$	5	0	15	20	35	40	55	65

The maximal value of a feasible subset is $F[5, 6] = 65$. The optimal subset is {item 3, item 5}.

b.-c. The instance has a unique optimal subset in view of the following general property: An instance of the knapsack problem has a unique optimal solution if and only if the algorithm for obtaining an optimal subset, which retraces backward the computation of $F[n, W]$, encounters no equality between $F[i-1, j]$ and $v_i + F[i-1, j-w_i]$ during its operation.

P. 312

7. Applying Floyd's algorithm to the given weight matrix generates the following sequence of matrices:

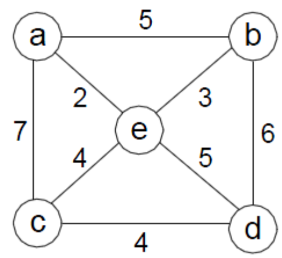
$$D^{(0)} = \begin{bmatrix} 0 & 2 & \infty & 1 & 8 \\ 6 & 0 & 3 & 2 & \infty \\ \infty & \infty & 0 & 4 & \infty \\ \infty & \infty & 2 & 0 & 3 \\ 3 & \infty & \infty & \infty & 0 \end{bmatrix} \qquad D^{(1)} = \begin{bmatrix} 0 & 2 & \infty & 1 & 8 \\ 6 & 0 & 3 & 2 & \mathbf{14} \\ \infty & \infty & 0 & 4 & \infty \\ \infty & \infty & 2 & 0 & 3 \\ 3 & \mathbf{5} & \infty & \mathbf{4} & 0 \end{bmatrix}$$

$$D^{(2)} = \begin{bmatrix} 0 & 2 & \mathbf{5} & 1 & 8 \\ 6 & 0 & 3 & 2 & 14 \\ \infty & \infty & 0 & 4 & \infty \\ \infty & \infty & 2 & 0 & 3 \\ 3 & 5 & \mathbf{8} & 4 & 0 \end{bmatrix} \qquad D^{(3)} = \begin{bmatrix} 0 & 2 & 5 & 1 & 8 \\ 6 & 0 & 3 & 2 & 14 \\ \infty & \infty & 0 & 4 & \infty \\ \infty & \infty & 2 & 0 & 3 \\ 3 & 5 & 8 & 4 & 0 \end{bmatrix}$$

$$D^{(4)} = \begin{bmatrix} 0 & 2 & \mathbf{3} & 1 & \mathbf{4} \\ 6 & 0 & 3 & 2 & \mathbf{5} \\ \infty & \infty & 0 & 4 & \mathbf{7} \\ \infty & \infty & 2 & 0 & 3 \\ 3 & 5 & \mathbf{6} & 4 & 0 \end{bmatrix} \qquad D^{(5)} = \begin{bmatrix} 0 & 2 & 3 & 1 & 4 \\ 6 & 0 & 3 & 2 & 5 \\ \mathbf{10} & \mathbf{12} & 0 & 4 & 7 \\ \mathbf{6} & \mathbf{8} & 2 & 0 & 3 \\ 3 & 5 & 6 & 4 & 0 \end{bmatrix} = D$$

P. 324

9. a. Applying Prim's algorithm to the graph

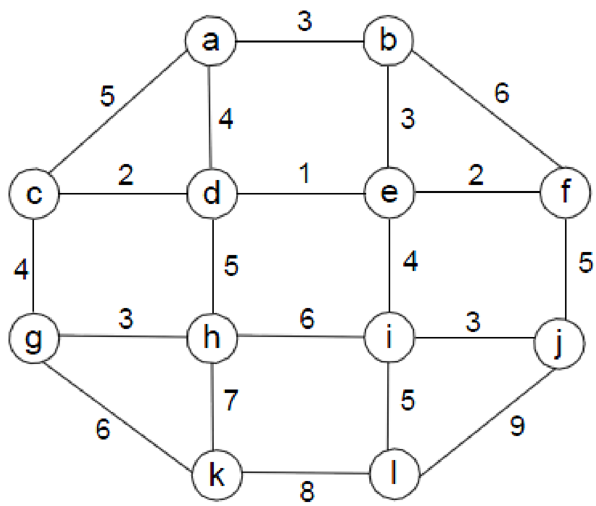


we obtain

Tree vertices	Priority queue of remaining vertices
a(-,-)	b(a,5) c(a,7) d(a,∞) **e(a,2)**
e(a,2)	**b(e,3)** c(e,4) d(e,5)
b(e,3)	**c(e,4)** d(e,5)
c(e,4)	**d(c,4)**
d(c,4)	

The minimum spanning tree found by the algorithm comprises the edges *ae*, *eb*, *ec*, and *cd*.

b. Applying Prim's algorithm to the graph given, we obtain

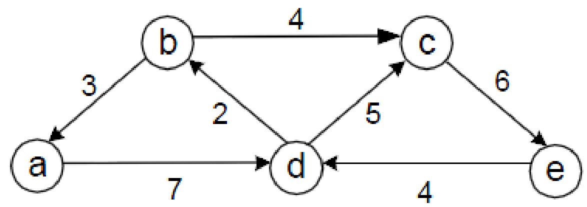


Tree vertices	Priority queue of fringe vertices
a(-,-)	**b(a,3)** c(a,5) d(a,4)
b(a,3)	c(a,5) d(a,4) **e(b,3)** f(b,6)
e(b,3)	*c(a,5)* **d(e,1)** f(e,2) i(e,4)
d(e,1)	**c(d,2)** f(e,2) i(e,4) h(d,5)
c(d,2)	**f(e,2)** i(e,4) h(d,5) g(c,4)
f(e,2)	**i(e,4)** h(d,5) g(c,4) j(f,5)
i(e,4)	h(d,5) g(c,4) **j(i,3)** l(i,5)
j(i,3)	h(d,5) **g(c,4)** l(i,5)
g(c,4)	**h(g,3)** l(i,5) k(g,6)
h(g,3)	**l(i,5)** k(g,6)
l(i,5)	**k(g,6)**
k(g,6)	

The minimum spanning tree found by the algorithm comprises the edges ab, be, ed, dc, ef, ei, ij, cg, gh, il, gk.

P. 337

2. a.

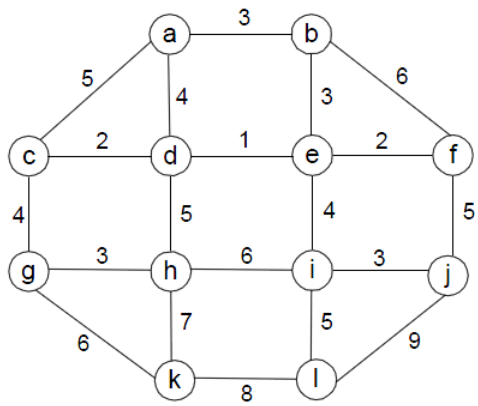


Tree vertices	Remaining vertices
a(-,0)	b(-,∞) c(−,∞) **d(a,7)** e(-,∞)
d(a,7)	**b(d,7+2)** c(d,7+5) e(-,∞)
b(d,9)	**c(d,12)** e(-,∞)
c(d,12)	**e(c,12+6)**
e(c,18)	

The shortest paths (identified by following nonnumeric labels backwards from a destination vertex to the source) and their lengths are:

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

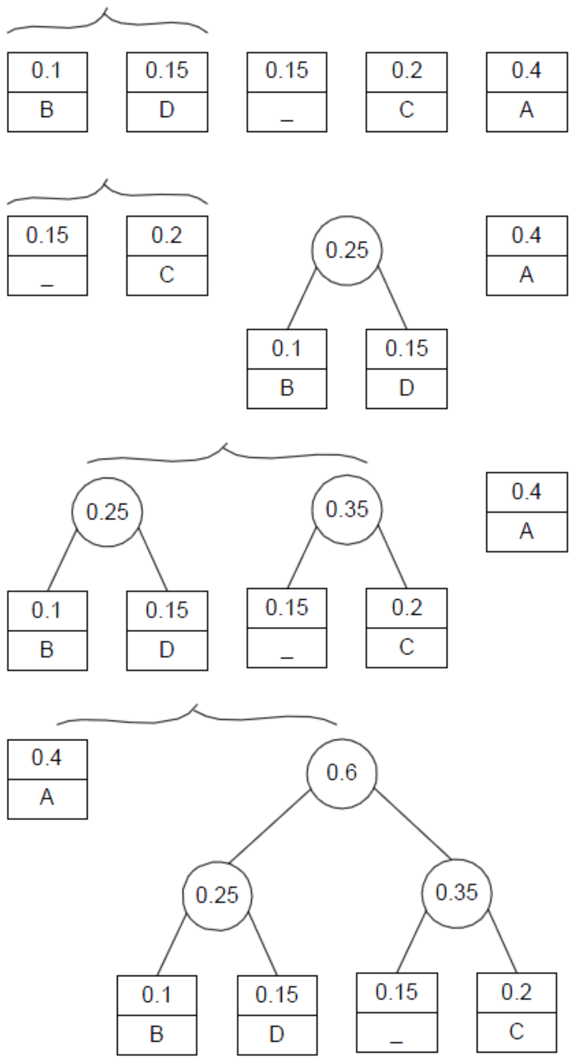
b.

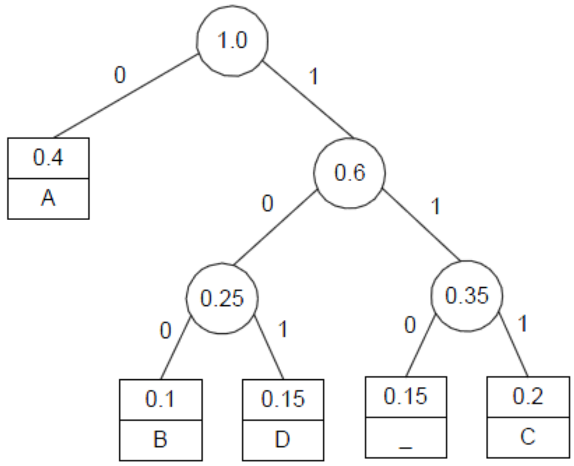


Tree vertices	Fringe vertices	Shortest paths from a
a(-,0)	**b(a,3)** c(a,5) d(a,4)	to b: $a-b$ of length 3
b(a,3)	c(a,5) **d(a,4)** e(b,3+3) f(b,3+6)	to d: $a-d$ of length 4
d(a,4)	**c(a,5)** e(d,4+1) f(a,9) h(d,4+5)	to c: $a-c$ of length 5
c(a,5)	**e(d,5)** f(a,9) h(d,9) g(c,5+4))	to e: $a-d-e$ of length 5
e(d,5)	**f(e,5+2)** h(d,9) g(c,9) i(e,5+4)	to f: $a-d-e-f$ of length 7
f(e,7)	**h(d,9)** g(c,9) i(e,9) j(f,7+5)	to h: $a-d-h$ of length 9
h(d,9)	**g(c,9)** i(e,9) j(f,12) k(h,9+7))	to g: $a-c-g$ of length 9
g(c,9)	**i(e,9)** j(f,12) k(g,9+6)	to i: $a-d-e-i$ of length 9
i(e,9)	**j(f,12)** k(g,15) l(i,9+5)	to j: $a-d-e-f-j$ of length 12
j(f,12)	k(g,15) **l(i,14)**	to l: $a-d-e-i-l$ of length 14
l(i,14)	**k(g,15)**	to k: $a-c-g-k$ of length 15
k(g,15)		

P. 342

1. a.


0.1
B
0.15
D
0.15
_
0.2
C
0.4
A
0.15
_
0.2
C
0.25
0.1
B
0.15
D
0.4
A
0.25
0.35
0.4
A
0.1
B
0.15
D
0.15
_
0.2
C
0.4
A
0.6
0.25
0.35
0.1
B
0.15
D
0.15
_
0.2
C



character	A	B	C	D	_
probability	0.4	0.1	0.2	0.15	0.15
codeword	0	100	111	101	110

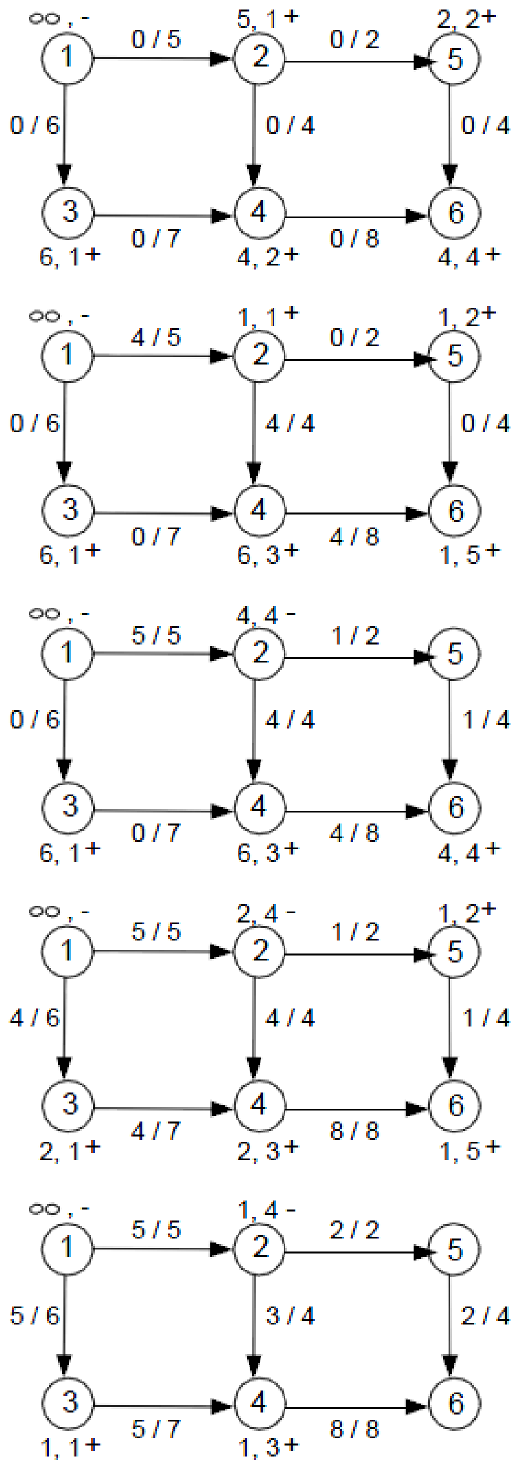
b. The text `ABACABAD` will be encoded as 0100011101000101.

c. With the code of part a, 100010111001010 will be decoded as

$$\underset{B}{100}|\underset{A}{0}|\underset{D}{101}|\underset{\_}{110}|\underset{A}{0}|\underset{D}{101}|\underset{A}{0}$$

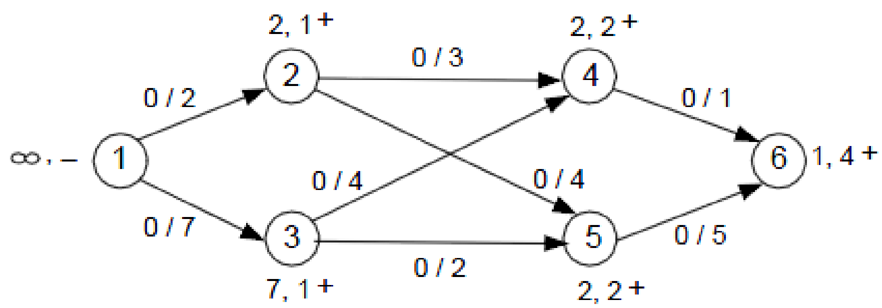
P. 371 (Done slightly differently than what I showed in class. Follow the in-class method. Check the final solution. On the exam, make sure you show the residual network with back edges for each step.)

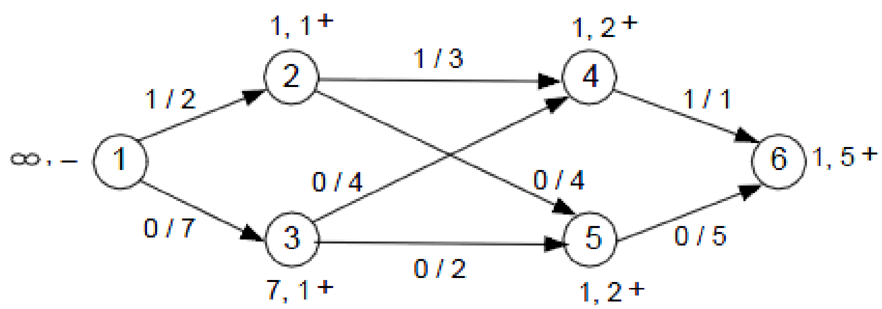
2\. a. Here is an application of the shortest-augmenting path algorithm to the network of Problem 2a:

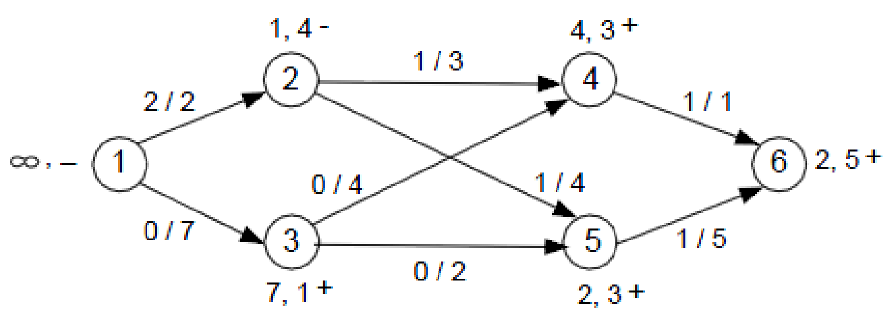


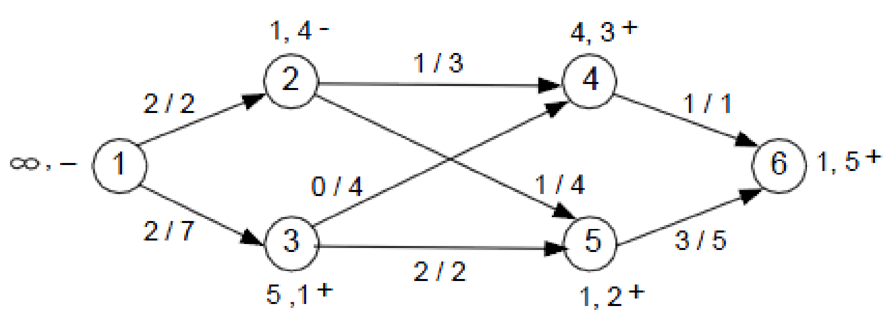
The maximum flow is shown on the last diagram above. The minimum cut found is $\{(2,5\}, (4,6)\}$.

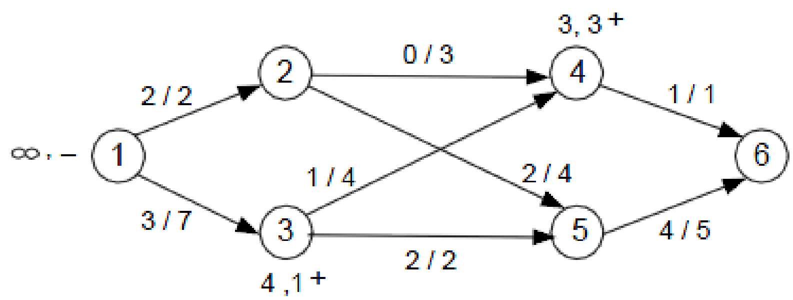
b. Here is an application of the shortest-augmenting path algorithm to the network of Problem 10.2b:











The maximum flow of value 5 is shown on the last diagram above. The minimum cut found is $\{(1,2),(3,5),(4,6)\}$.