

CSCI 741, Algorithm Analysis, Final Exam

Name: (Capital letters, First, Middle Init, Last). \_\_\_\_\_

Do as many problems (or their parts) as possible. You may not be able to finish all the problems in the given time. Show all steps. Accuracy of computations is important.

1.(15pts)

1.(a). (7 points) Consider the following recurrence.

$$T(1) = a, a > 0$$

$$T(n) = T(\lfloor n/2 \rfloor) + bn, n \geq 2, b > 0.$$

Assume  $n = 2^k$  and solve the above recurrence to obtain its exact solution. Then show that  $T(n) = \Theta(n)$  if  $n = 2^k$ .

$$T(n) = T(n/2) + bn$$

$$= T(n/2^2) + bn(1 + 1/2)$$

$$= T(n/2^3) + bn(1 + 1/2 + (1/2)^2)$$

.

.

$$T(n) = T(n/2^k) + bn(1 + 1/2 + (1/2)^2 \dots + (1/2)^{k-1}) \quad (2^k = n \Leftrightarrow k = \log n)$$

$$T(n) = T(n/2^k) + 2bn(1 - (1/2)^k)$$

$$T(n) = T(1) + 2b(n - 1)$$

$$T(n) = a + 2b(n - 1)$$

$$T(n) = a - 2b + 2bn, \quad T(n) = \Theta(n).$$

1.(b). (2 pts) Show  $f(n) = n$  is smooth.

$$f(2n) = 2n = 2f(n) = \Theta(f(n)), \text{ and } f(n) \text{ is nondecreasing} \Rightarrow f(n) \text{ is smooth.}$$

1.(c). (4 pts) Show  $T(n)$  is non decreasing for all  $n$ , i.e.,  $T(n) \leq T(n+1)$  for all  $n$ .

Proof by strong induction.

Base case;  $T(2) = T(1) + b = a + b > a = T(1)$ .

Inductive step: Assume  $T(i) \leq T(j)$  for all  $1 \leq i \leq j \leq n$ , and show that  $T(n) \leq T(n+1)$ . Now,

$$T(n+1) = T(\lfloor (n+1)/2 \rfloor) + b(n+1),$$

$$T(n) = T(\lfloor n/2 \rfloor) + bn. \text{ Hence,}$$

$$T(n+1) - T(n) = T(\lfloor (n+1)/2 \rfloor) - T(\lfloor n/2 \rfloor) + b,$$

Now  $1 \leq \lfloor n/2 \rfloor \leq \lfloor (n+1)/2 \rfloor \leq n$  for all  $n \geq 2$ . Hence, by inductive assumption we have,

$T(\lfloor (n+1)/2 \rfloor) - T(\lfloor n/2 \rfloor) \geq 0$ . Hence, the above equation shows that  $T(n+1) - T(n) \geq 0$ , for all  $n \geq 2$ . We have already shown that  $T(2) - T(1) \geq 0$ . Hence,  $T(n+1) - T(n) \geq 0$ , for all  $n \geq 1$ .

1.(d). (2 pts) Can you extend the complexity  $T(n) = \Theta(n)$  to all  $n$ ? State any relevant theorem you use to justify your answer.

Yes.  $f(n) = n$  is smooth and  $T(n)$  is nondecreasing, hence, by smooth function theory, we conclude  $T(n) = \Theta(n)$  for all  $n \geq 1$ .

2 (15 pts)

2.(a). (10 pts)

(i) (8 pts) The following is the dynamic programming recursive equation for solving the coin changing problem where  $1 = d_0 < d_1 < d_2 < \dots < d_k$  are the denominations of coins.

$f(a)$ : Minimum number of coins needed to change the amount  $a$ .

$$f(0) = 0$$

$$f(a) = +\infty \text{ if } a < 0.$$

$$f(a) = \min \{1+f(a-d_i) : i=0,1,2, \dots, k\}$$

$$= \min \{1+f(a-d_0), 1+f(a-d_1), 1+f(a-d_2), \dots, 1+f(a-d_k)\}.$$

$p(a)$  = listing (as a set) of minimum # of coins that make change for amount  $a$ .

Consider the following denominations and the amount:

$$d_0 = 1 < d_1 = 4 < d_2 = 7 \text{ and } a = 8.$$

$d_0$	$d_1$	$d_2$
1	0	1
0	2	0

Apply the algorithm to compute the number of coins and their denominations to change amount  $a$ . Compute all solutions. Show all computations.

(ii) (2 pts) Now apply the greedy algorithm to the above data and obtain the solution. Compare this with the dynamic programming solution.

**$d_0 = 1 < d_1 = 4 < d_2 = 7$  and  $a = 8$ .**

$$f(0) = 0$$

$$f(1) = \min\{1+f(1-d_0)\} = \min\{1+f(1-1)\} = \min\{1+f(0)\} = \min\{1+0\} = 1.$$

$$p(1) = \{\text{one 1-coin}\}$$

$$f(2) = \min\{1+f(2-d_0)\} = \min\{1+f(2-1)\} = \min\{1+f(1)\} = \min\{1+1\} = 2.$$

$$p(2) = \{\text{two 1-coins}\}$$

$$f(3) = \min\{1+f(3-d_0)\} = \min\{1+f(3-1)\} = \min\{1+f(2)\} = \min\{1+2\} = 3.$$

$$p(3) = \{\text{three 1-coins}\}$$

$$f(4) = \min\{1+f(4-d_0), 1+f(4-d_1)\} = \min\{1+f(4-1), 1+f(4-0)\} = \min\{1+f(3), 1+f(0)\} = \min\{1+3, 1+0\} = \min\{2, 1\} = 1.$$

$$p(4) = \{\text{one 4-coin}\}$$

$$f(5) = \min\{1+f(5-d_0), 1+f(5-d_1)\} = \min\{1+f(5-1), 1+f(5-4)\} = \min\{1+f(4), 1+f(1)\} = \min\{1+1, 1+1\} = \min\{2, 2\} = 2.$$

$$p(5) = \{\text{one 1-coin, one 4-coin}\}$$

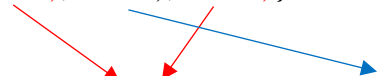
$$f(6) = \min\{1+f(6-d_0), 1+f(6-d_1)\} = \min\{1+f(6-1), 1+f(6-4)\} = \min\{1+f(5), 1+f(2)\} = \min\{1+2, 1+2\} = \min\{3, 3\} = 3.$$

$$p(6) = \{\text{two one-coin, one 4-coin}\}$$

$$f(7) = \min\{1+f(7-d_0), 1+f(7-d_1), 1+f(7-d_2)\} = \min\{1+f(7-1), 1+f(7-4), 1+f(7-7)\} = \min\{1+f(6), 1+f(3), 1+f(0)\} = \min\{1+3, 1+3, 1+0\} = \min\{3, 2, 1\} = 1.$$

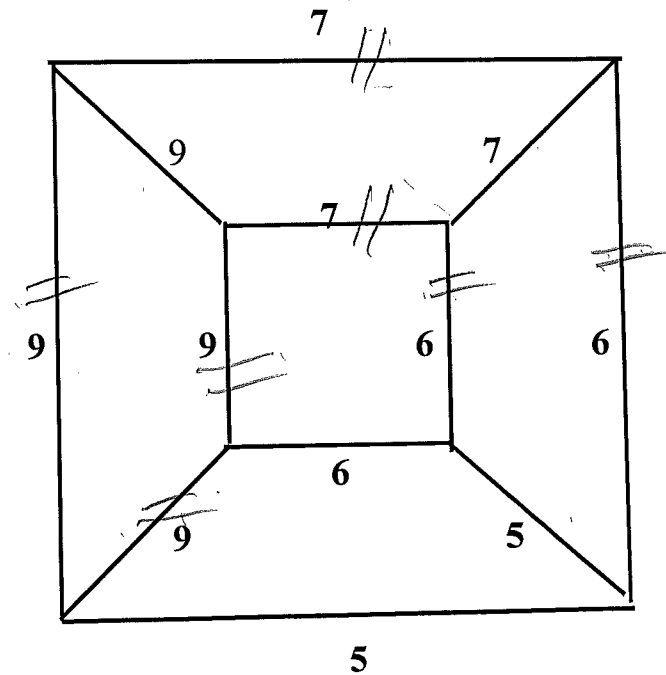
$$p(7) = \{\text{one 7-coin}\}$$

$$f(8) = \min\{1+f(8-d_0), 1+f(8-d_1), 1+f(8-d_2)\} = \min\{1+f(8-1), 1+f(8-4), 1+f(8-7)\} = \min\{1+f(7), 1+f(4), 1+f(1)\} = \min\{1+1, 1+1, 1+1\} = \min\{2, 2, 2\} = 2.$$

$$p(8) = \{\text{one 1-coin, one 7-coin}\} \text{ or } \{\text{two 4-coins}\}$$


2.(b) (5 pts) Can you use greedy algorithm to find a **maximum** cost spanning tree in the following network? If so, find the tree and its cost, and explain your algorithm briefly. If not, explain why.

Apply Prim's or Kruskal's algorithm choosing largest cost arcs.



$$27 + 14 + 12 = 53$$

3. (15 pts) Consider the following version of the Merge-Sort algorithm applied to sort  $n$  data elements. It divides the list into two sub-lists, one of size  $n/3$  and the other of size  $2n/3$ , approximately. It then sorts the two lists recursively and merges them. The algorithm is invoked by the top-level call `merge-sort (1, n)`.

The following code is exactly the same as that of the regular merge-sort algorithm where the lists are divided into two almost equal parts with the exception of one change shown in bold letters that calculates `mid` in a different way.

```
algorithm merge-sort (low, high)
```

```
  // A (low: high) is an array to be sorted
```

```
  if low < high then
```

```
    mid  $\leftarrow \lfloor (2 * \text{low} + \text{high}) / 3 \rfloor$  //mid =  $\lfloor \text{low} + (1/3) (\text{high} - \text{low}) \rfloor$  , one third
```

```
    //divide problem into two sub-problems
```

```
    merge-sort (low, mid)
```

```
    merge-sort (mid + 1, high)
```

```
  //merge the 2 sorted lists, A (low, mid), and A (mid + 1, high) to
```

```
  // form a sorted array A (low, high)
```

```
  //Use algorithm merge in the document "Divide and Conquer" on BB as below.
```

```
    merge (low, mid, high)
```

```
  endif
```

algorithm merge (low, mid, high) //As in document “Merging ordered lists” on BB

// A (low, mid), A (mid+1, high) are sorted arrays

//combine them to form a sorted array A (low, high)

$p \leftarrow \text{low}, q \leftarrow \text{low}, r \leftarrow \text{mid} + 1$

while ( $p \leq \text{mid}$ ) and ( $q \leq \text{high}$ ) do

    If  $A(p) \leq A(r)$  then

$B(q) \leftarrow A(p), p \leftarrow p+1$

    else

$B(q) \leftarrow A(r), r \leftarrow r+1$

    endif

$q \leftarrow q+1$

endwhile

if  $p > \text{mid}$  then

    for  $k = r$  to high do

$B(q) \leftarrow A(k), q \leftarrow q+1$

    endfor

else

    for  $k = p$  to mid do

$B(q) \leftarrow A(k); q \leftarrow q+1;$

    endfor

endif

for  $k = \text{low}$  to high do  $A(k) \leftarrow B(k)$  endfor

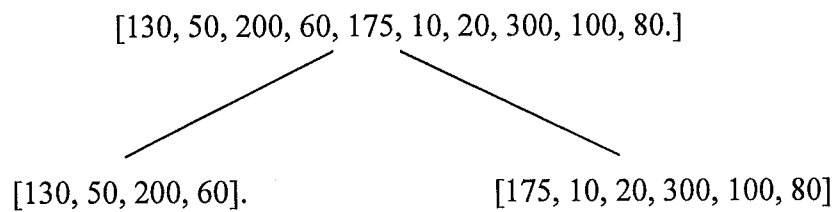
The algorithm is invoked by the top-level call merge-sort (1, n). Note that  $\text{low} = 1, \text{mid} = \lfloor (2+n)/3 \rfloor$  and  $\text{high} = n$ .

The above algorithm is applied to the following data of 10 elements.

130, 50, 200, 60, 175, 10, 20, 300, 100, 80.

Show the tree of recursive calls of both divide and merge actions. For your convenience, the following shows the first step. Note  $\text{mid} = \lfloor (2*1 + n)/3 \rfloor = \lfloor (2*1 + 10)/3 \rfloor = 4$ . Please complete the tree.

Tree of recursive calls:

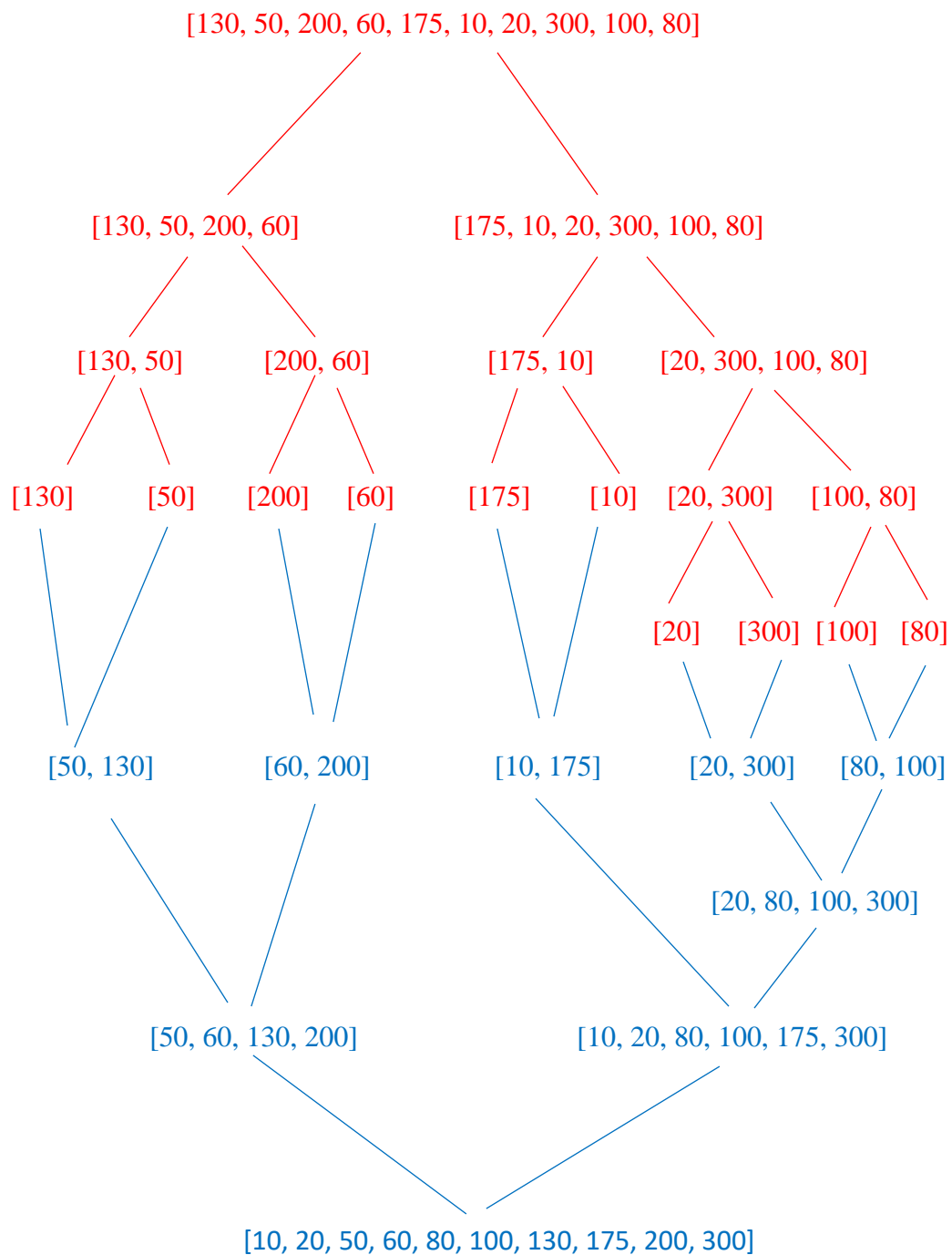




Apply merge-sort to the following data: [A(1) :A(10)] =

[130, 50, 200, 60, 175, 10, 20, 300, 100, 80]

In the following decision tree, **red numbers and lines indicate divide activity** and **blue numbers and lines indicate merging into ordered list**.



4. (15 pts)

The following is a list of 10 elements.

A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	A(7)	A(8)	A(9)	A(10)
3	8	6	4	7	10	2	9	1	5

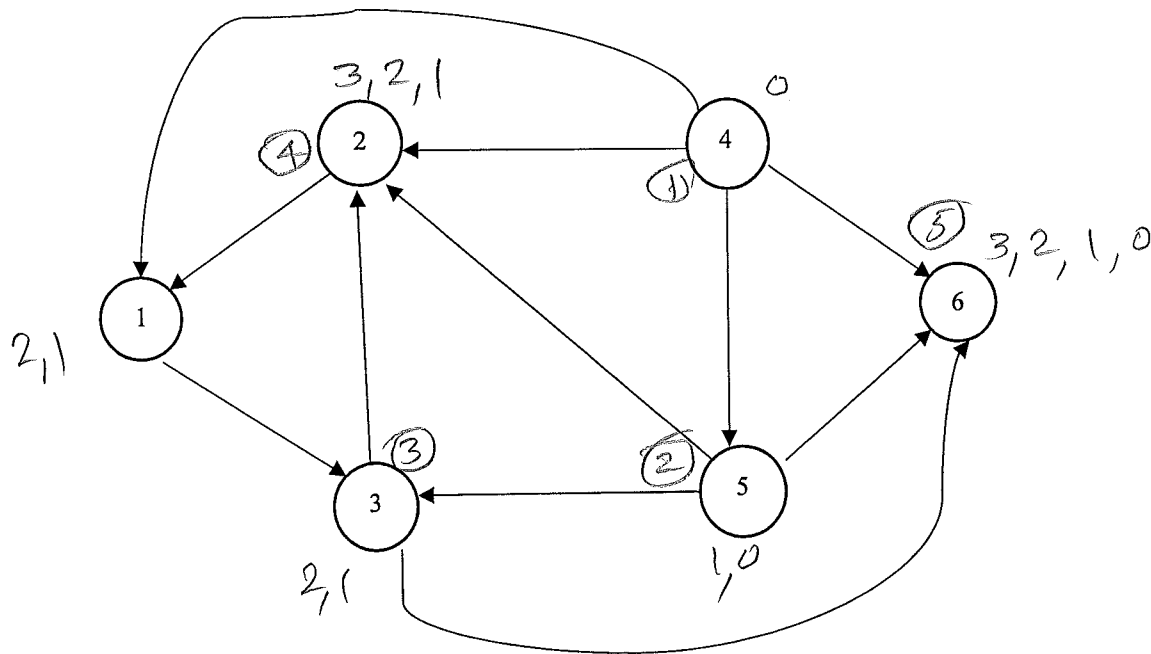
(a). (10 pts) Apply the quicksort algorithm covered in class to the list letting the partitioning element to be the last element of the list, i.e., 5. Perform steps until the partitioning element is in the correct position.

(b). (5 pts) The partitioning element 5 divides the list into two parts. Apply the quicksort algorithm to each of the two parts until their partitioning elements are in the correct position.

3 8 6 4 7 10 2 9 1 5  
 3 4 6 8 7 10 2 9 1 5  
 3 4 2 8 7 10 6 9 1 5  
 3 4 2 1 7 10 6 9 8 5  
 3 4 2 1 5 10 6 9 8 7  
  
 3 4 2 1      10 6 9 8 7  
 1 4 2 3      6 10 9 8 7  
 1 4 2 3      6 7 9 8 10

5.(a). (7 pts) Consider the network shown below. Use an algorithm (not by inspection) to determine if the network is acyclic or not. State your conclusion clearly.

*Apply topological ordering*

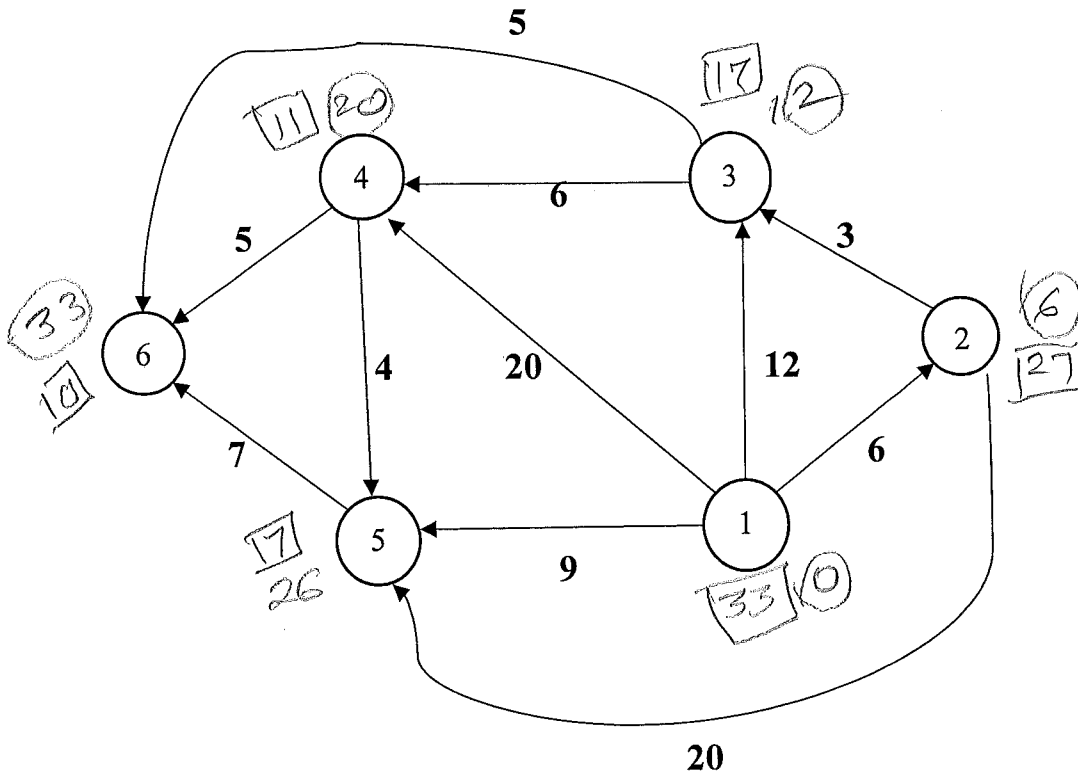


*Numbers indicate the indegrees.*

*The algorithm cannot be completed because there is no node with indegree 0.*

*Not acyclic.*

5.(b). (8 pts) Consider the following acyclic digraph shown below. It is topologically ordered. The bold numbers on arcs are arc lengths.

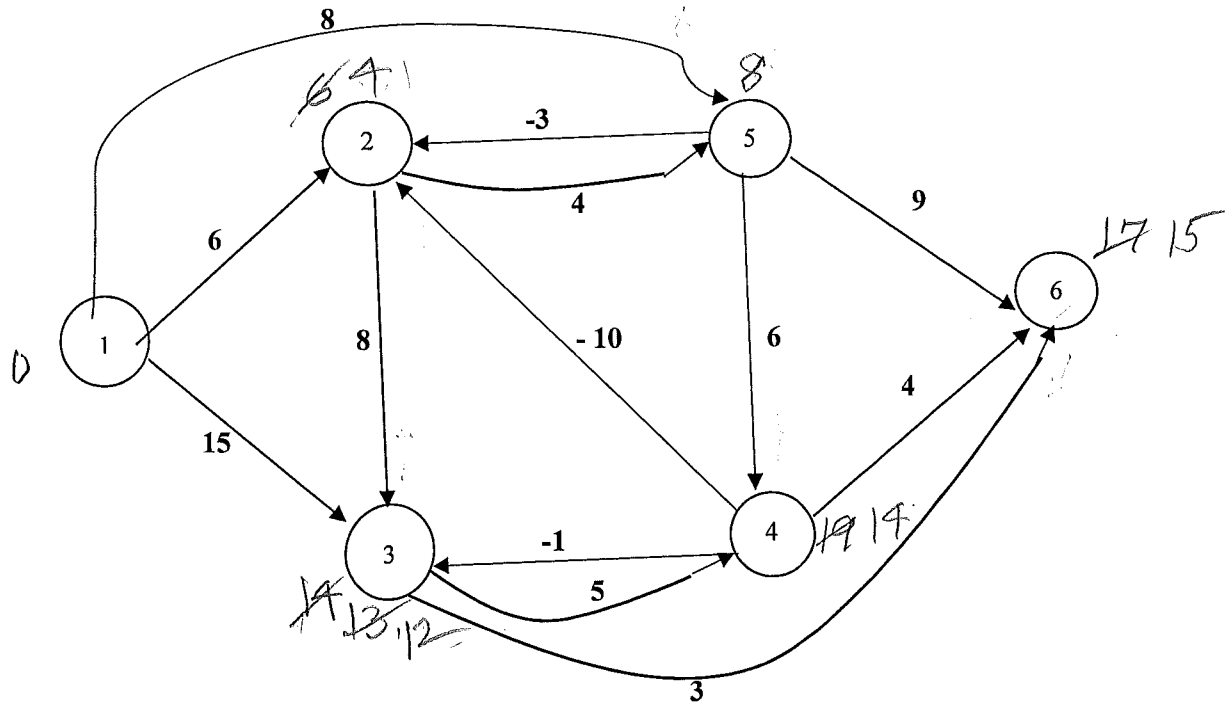


- i). (3 pts) Find the longest path from node 1 to each node. numbers in
- ii). (3 pts) Find the longest path from each node to node 6.
- iii). (2 pts) If you add the two numbers from part (i) and part (ii) at each node, what is the interpretation of the sum so obtained at each node?

Shortest path from node 1 to nodes  
passing through a given node.

6. (15 pts)

6.(a). (11 pts) Consider the network shown in the figure below. Apply Yen's algorithm to find the shortest path from node 1 to all nodes. Tabulate your results in the tableau below. Each entry in the tableau under a given iteration gives the shortest path from node 1 to a given node in that iteration.



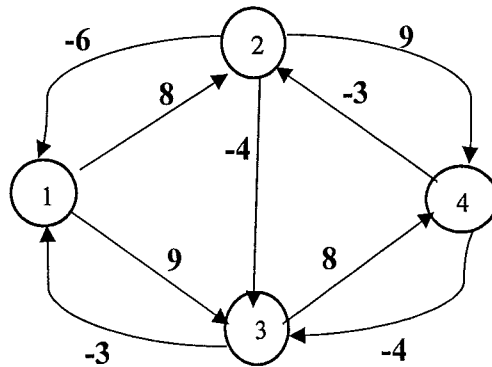
Node#	Iteration #					
	0	1	2	3	4	5
1	0	0	0	0	0	
2	$\infty$	6	4	4	4	
3	$\infty$	14	13	12	12	
4	$\infty$	19	14	14	14	
5	$\infty$	8	8	8	8	
6	$\infty$	17	17	15	15	

6.b.(4 pts) What is the length of the shortest path from node 1 to node 6? What is the shortest path? (List the nodes on the shortest path.)

1 5 4 2 3 6

7 (15 pts)

7(a) (10 pts) Consider the application of Floyd's algorithm for finding shortest paths between any pair of nodes to the directed network given below. In this respect set up the initial tableau, and make next two iterations of the algorithm completing the tableau below.



Distance/cost

	1	2	3	4
1	0	8	9	<del>∞</del>
2	-6	0	-4	9
3	-3	∞	0	8
4	<del>∞</del>	-3	-4	0

Policy

	1	2	3	4
1	-1	-1	-1	-1
2	-1	-1	-1	-1
3	-1	-1	-1	-1
4	-1	-1	-1	-1

	1	2	3	4
1	0	8	9	<del>∞</del>
2	-6	0	-4	9
3	-3	(5)	0	8
4	<del>∞</del>	-3	-4	0

	1	2	3	4
1	-1	-1	-1	-1
2	-1	-1	-1	-1
3	-1	(1)	-1	-1
4	-1	-1	-1	-1

	1	2	3	4
1	0	8	4	17
2	-6	0	-4	9
3	-3	5	0	8
4	-9	-3	-7	0

	1	2	3	4
1	-1	-1	2	2
2	-1	-1	-1	-1
3	-1	1	-1	-1
4	2	-1	2	-1

7(b)(5 pts) The following tableau are obtained at the end of an application of Floyd's algorithm to a directed network of 7 nodes. Some entries were missing and are shown by blanks. From the available entries determine the cost of the shortest path from node 3 to node 5, and all the nodes on this path in a sequence from node 3 to node 5.

Distance/cost

	1	2	3	4	5	6	7
1	0				6		
2	13	0		7	19		9
3	21	8	0	15	27	3	17
4	6			0	12		2
5					0		
6	18	5		12	24	0	14
7	4				10		0

Policy

	1	2	3	4	5	6	7
1					-1		
2			4	-1	7		4
3		6			2	-1	
4	7						-1
5			2			1	
6		-1		2			
7	-1				1		

3 -6 -2 -4 -7 -1 -5



8. (15 pts) Consider the sequence defined by

$$T_n = 6 T_{n-1} - 9 T_{n-2}, n \geq 2,$$

$$T_0 = 0, T_1 = 1$$

Apply the generating function method to determine the value of  $T_n$ .

8. (15 pts) Consider the sequence defined by

$$T_n = 6 T_{n-1} - 9 T_{n-2}, n \geq 2, (*)$$

$$T_0 = 0, T_1 = 1$$

Apply the generating function method to determine the value of  $T_n$ .

We create the generating functions of the given recurrence equations.

$$G(z) = \{\sum T_n z^n: n \geq 0\} = T_0 + T_1 z + T_2 z^2 + \dots + T_n z^n + \dots$$

We multiply linear recurrence (\*) by  $z^n$  to get,

$$z^n T_n = 6 z^n T_{n-1} - 9 z^n T_{n-2}, n \geq 2.$$

We sum the above equation for all  $n \geq 2$  to get,

$$\{\sum T_n z^n: n \geq 2\} = 6z \{\sum T_{n-1} z^{n-1}: n \geq 2\} - 9z^2 \{\sum T_{n-2} z^{n-2}: n \geq 2\}$$

Substituting  $G(z)$  in the previous equation, we get

$$G(z) - T_0 - T_1 z = 6z (G(z) - T_0) - 9z^2 G(z).$$

Since  $T_0 = 0$  and  $T_1 = 1$ , we have

$$G(z) - z = 6z G(z) - 9z^2 G(z).$$

This gives

$$G(z) = z / (1 - 6z + 9z^2)$$

Note that  $(1 - 6z + 9z^2) = (1 - 3z)^2$ . Hence,

$$G(z) = z / (1 - 6z + 9z^2) = z / ((1 - 3z)^2) = (1/3) (3z / ((1 - 3z)^2))$$

Recall that

$$1 / (1 - x) = 1 + x + x^2 + \dots + x^n \dots, |x| < 1.$$

Differentiating the above, we get

$$1 / (1 - x)^2 = 1 + 2x + 3x^2 + \dots + nx^{n-1} \dots, |x| < 1.$$

Thus,

$$x/(1-x)^2 = x + 2x^2 + 3x^3 + \dots + nx^n \dots, |x| < 1.$$

Substituting  $x = 3z$  in the above, we get

$$3z/(1-3z)^2 = 3z + 2(3z)^2 + 3(3z)^3 + \dots + n(3z)^n \dots, |3z| < 1.$$

Hence,

$$\begin{aligned} G(z) &= (1/3) (3z/((1-3z)^2)) \\ &= (1/3) (3z + 2(3z)^2 + 3(3z)^3 + \dots + n(3z)^n \dots), |3z| < 1. \\ &= z + 2 \cdot 3 \cdot z^2 + 3 \cdot 3^2 z^3 + \dots + n3^{n-1}z^n \dots, |3z| < 1. \end{aligned}$$

Comparing with

$$G(z) = T_0 + T_1 z + T_2 z^2 + \dots + T_n z^n + \dots$$

we have

$$T_n = n3^{n-1}.$$