95-771 Data Structures and Algorithms
Spring 2014

Midterm Exam
Carnegie Mellon University Key

Print Full Name Key

Andrew USER ID ____________

There are 110 points on this exam. Your score will be computed as a percentage of ~~110~~. 124

Coding Lists and Trees (18 points)

1. The following program contains two println() statements. You will be asked to describe the output of the program when each of the println() statements is executed. You will also be asked to describe the run time complexity Big Θ of two methods. See below. (8 points):

```
class Node {
   private int data;
   private Node next;

   public Node(int data, Node next) {
      this.data = data;
      this.next = next;
   }

   public int getData() {
      return data;
   }

   public void setData(int data) {
      this.data = data;
   }

   public Node getNext() {
      return next;
   }

   public void setNext(Node next) {
      this.next = next;
   }
}
public class DSAMidtermProject {

  private Node list = null;

  public void push(int x) {
     list = new Node(x,list);
  }

  public int pop() {
     int x = list.getData();
     list = list.getNext();
     return x;
  }

  public boolean isEmpty() {
     return list == null;
  }
```

Kai

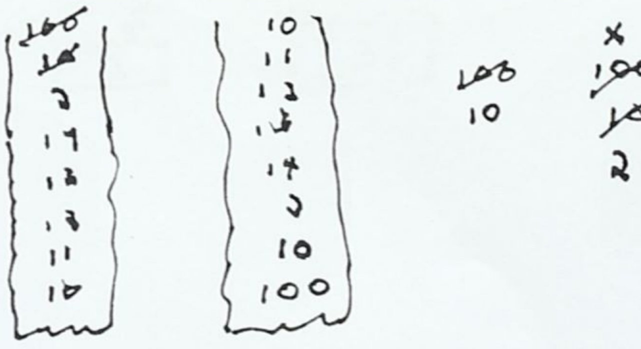
```
  public String toString() {
    String v = "";
    Node c = list;
    while (c != null) {
      v = v + c.getData();
      if(c.getNext()!= null) v = v + "->";
      else v = v + "--||";
      c = c.getNext();
    }
    return v;
  }

  public int retieveTheValue() {
    DSAMidtermProject temp = new DSAMidtermProject();
    int x = 1000000;
    while(!this.isEmpty()) {
      int y = this.pop();
      if(y < x) x = y;
      temp.push(y);
    }
    while(!temp.isEmpty()) {
      this.push(temp.pop());
    }
    return x;
  }

  public static void main(String args[]) {

    DSAMidtermProject dsa = new DSAMidtermProject();

    for (int j = 10; j < 15; j++) {
      dsa.push(j);
    }
    dsa.push(2);
    dsa.push(10);
    dsa.push(100);

    // 1.a
    System.out.println(dsa.retieveTheValue());
```

2

```
    // 1.b
    System.out.println(dsa);
```

100 -> 10 -> 2 -> 14 -> 13 -> 12 -> 11 -> 10

```
  }
}
```



Key

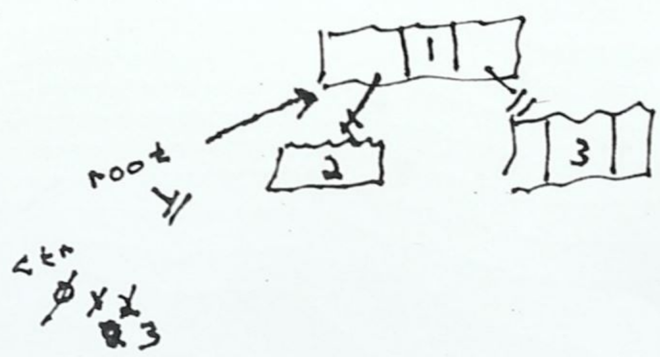
1.a What will the program display at the println marked 1.a? 2

1.b What will the program display at the println marked 1.b? 100 -> 10 -> 2 -> 14 -> 13 -> 12 -> 11 -> 10 -||

1.c What is the run time complexity (Big Θ) of retieveTheValue()? Θ (N)

1.d Is it correct to say that the toString() method runs in O(2^n) ? Circle [✓ True] or False

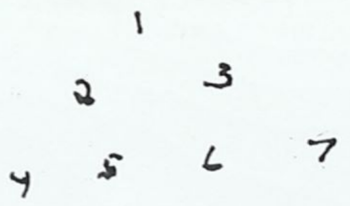
2. Study the execution of the following program. Questions appear below. (10 points):



```
class Node {
   public int data;
   public Node lc;
   public Node rc;
   public Node(Node lc, int x, Node rc) {
      this.lc = lc;
      this.data = x;
      this.rc = rc;
   }
}
public class SimpleTree {

   public Node root;
   public int ctr;

   public SimpleTree() {
      root = null;
      ctr = 0;
   }

   private Node add(Node t){

      if (t == null) {
         ctr = ctr + 1;
         return new Node(null,ctr,null);
      }
      t.lc = add(t.lc);
      t.rc = add(t.rc);
      return t;
   }

   public void add() {
      if (root == null) {
         ctr = ctr + 1;
         root = new Node(null,ctr,null);
      }
      else {
         add(root);
      }
   }
```

Key

```
public void traversal(Node t) {

  if(t != null) {
     traversal(t.lc);
     System.out.print(t.data + ":");
     traversal(t.rc);
  }
}
public void traversal() {
   traversal(root);
}

public static void main(String[] args) {
   SimpleTree st = new SimpleTree();
   st.add();
   st.add();

   // Question 2.a
   st.traversal();
   System.out.println();

   SimpleTree coolTree = new SimpleTree();
   coolTree.add();
   coolTree.add();
   coolTree.add();

   // Question 2.b
   coolTree.traversal();

  }
}
```



2.a What will the program display at the traversal marked Question 2.a?

2: 1: 3:

2.b What will the program display at the traversal marked Question 2.b?

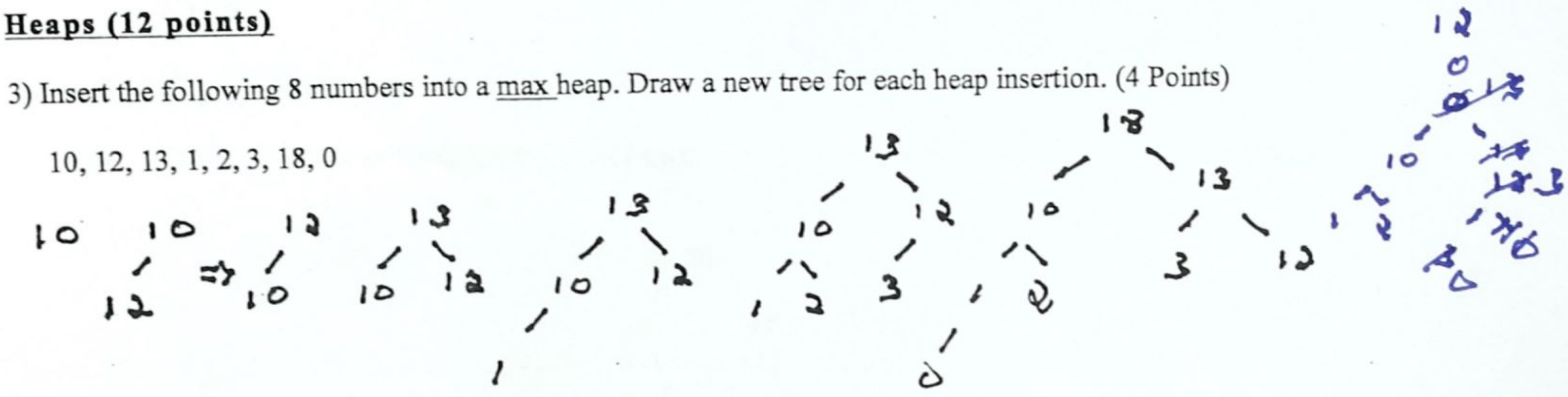
4 2 5 1 6 3 7

Key

Heaps (12 points)

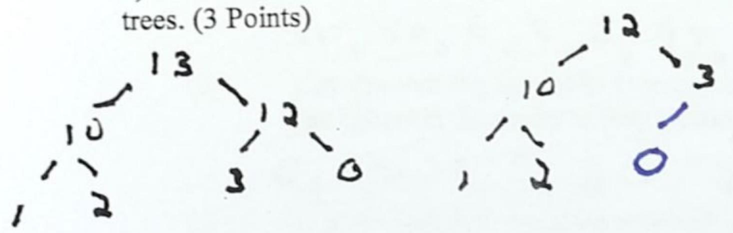
3) Insert the following 8 numbers into a max heap. Draw a new tree for each heap insertion. (4 Points)

10, 12, 13, 1, 2, 3, 18, 0



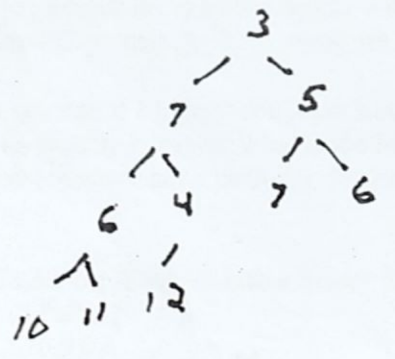
4) What is the height of the tree that you drew in question 3? (2 Points) 3

5) Perform exactly two deleteMax() operations on the heap that you drew in question 3. Draw the resulting trees. (3 Points)



6) Consider the following min heap implemented in an array. It is not quite correct. To make it a proper max heap exactly one swap must occur. What two numbers (child and parent) need to be swapped in order to make this a max heap? (3 points). PLACE CHECK MARKS NEXT TO THE TWO NUMBERS THAT NEED SWAPPED.

3
7 ✓
5
6
4 ✓
7
6
10
11
12



Key

Binary Trees (16 points)

7. Parts (a), (b), (c) refer to the following binary tree:

```
              20
           /     \
         40        97
        / \       / \
       6   9     2   76
           /
          8
```

(a) List the data that would be accessed by a pre-order traversal on the given tree by writing out the values in the nodes as they would be accessed, separated by commas. (3 points)

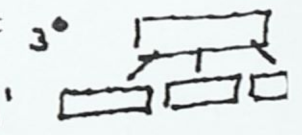
20, 40, 6, 9, 8, 97, 2, 76

(b) List the data that would be accessed by an in-order traversal on the given tree by writing out the values in the nodes as they would be accessed, separated by commas. (2 points)

6, 40, 8, 9, 20, 2, 97, 76

(c) List the data that would be accessed by a level-order traversal on the given tree by writing out the values in the nodes as they would be accessed, separated by commas. (2 points)

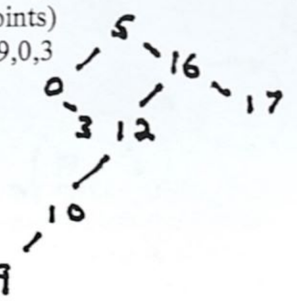
20, 40, 97, 6, 9, 2, 76, 8



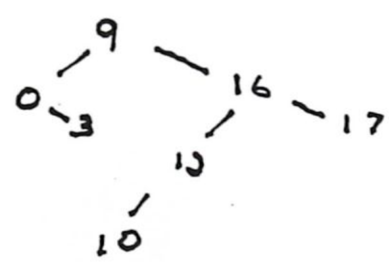
(d) In general, if a ternary (at most three children per node) tree is perfectly balanced (unlike the tree pictured here) and complete with height h, how many leaves, in terms of h, will the tree have? (2 points) 3^h Note, this tree has a perfectly flat bottom.

(e) In general, if a binary tree is perfectly balanced (unlike the tree pictured here) and complete with exactly k leaves. What is the height (in terms of k) of this tree? (2 points) $\log_2 K$ Note, this tree has a perfectly flat bottom.

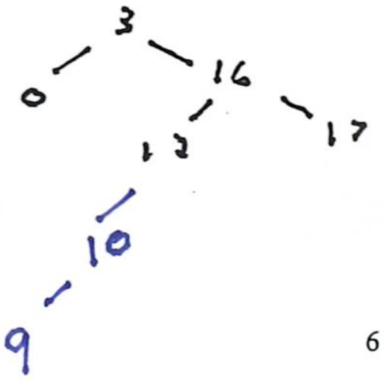
8. (a) Insert he following numbers into a Binary Search Tree. Draw the tree after all insertions are complete. (3 Points)
5,16,17,12,10,9,0,3



(b) Delete 5 from the final tree that you drew in 8 (a). Draw this final tree. (2 Points)
There are two possible answers. Either one is fine.



xor



Key

Project Questions (18 points)

9. Recall the Merkle-Hellman cryptosystem that we worked with in Project 1, the spell checker application in Project 2, and the calculator problem from Project 3.

Project 1 was based on the subset sum problem which is known to be NP-Complete. The problem itself can be described as follows: given a set of numbers *X* and a number **k**, is there a subset of *X*, which sums to **k**?

(a) Suppose X = {3,9,12,4, 6, 2} and **k** = 11. Is there a subset of X which sums to **k**?
Yes Yes/No (2 points)

(b) The type of problem you were asked to solve in question 9 (a) is (Circle one answer): (2 Points)

1. an optimization problem.
2. a problem that is impossible to solve.
3. a problem that has been proven to take exponential time to solve.
4. a problem that has been proven to take factorial time to solve.
[✓ 5.] a decision problem.

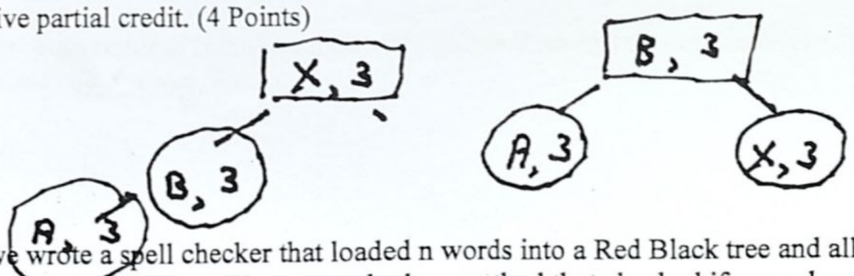
(c) Suppose Alice sends a message M encrypted to the integer (K) to Bob. K is computed using Bob's Merkle-Hellman public key combined with the message M. From Bob's perspective, he can easily solve for M given K because (2 pts)

1. M is prime.
2. Bob's public key is super increasing.
[✓ 3.] Bob can easily solve the subset sum problem with a super increasing sequence.
4. Bob can easily solve the subset sum problem with a random sequence.
5. M = K mod 13.

(d) Recall that a modular inverse of an integer b mod m is the integer b^{-1} such that $(b * b^{-1}) \bmod m=1$.
What is the modular inverse of 3 mod 7? 5 (4 Points)

(e) In Project 3, we wrote a calculator that processed RPN expressions and used a Red Black Tree. Draw what the Red Black Tree would look like after the following user interaction.
Draw RED nodes with a circle and BLACK nodes with a rectangle. If you show each step clearly, you will receive partial credit. (4 Points)

X 3 =
B X =
A X =



(f) In Project 2 we wrote a spell checker that loaded n words into a Red Black tree and allowed a user to make queries against the tree. We wrote a lookup method that checked if a word was present. Which of the following is true in the best case for the lookup method? Circle all of those that are true. (You may or may not have more than one answer.) (4 Points)

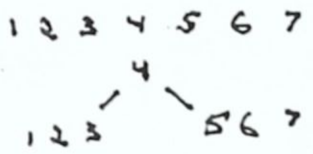
In the best case lookup is:
[✓ 1.] O(LogN)
[✓ 2.] O(1)
3. $\Omega(N^2)$
[✓ 4.] O(N)
5. Θ(LogN)
[✓ 6.] $O(2^n)$
[✓ 7.] $\Omega(1)$

Key

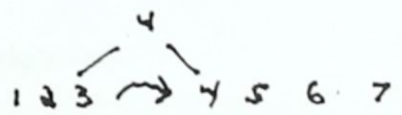
Balanced Trees (21 points)

21

10. a) Insert the following numbers into a B-Tree with a minimum of 3. 1,2,3,4,5,6,7. Draw the final tree. (5 Points)



b) Insert the following numbers into a B+ Tree with a minimum of 3. 1,2,3,4,5,6,7. Draw the final tree. (5 Points)



101
100
10100
+100
10200

c) Consider a B-Tree with a minimum of 50. What is the exact maximum number of keys such a tree could hold if the tree were of height 1? 10,200 (5 Points)

100
+ 100 × 101

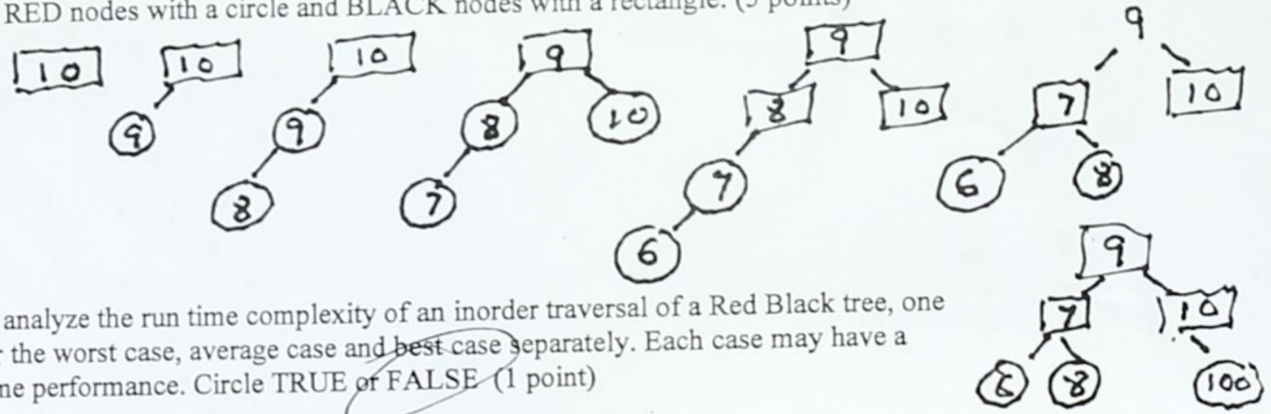
d) Suppose we have a B-Tree with a minimum of 50 stored on disk. Suppose too that we have 10,000 keys stored in the tree. How many disk accesses (in the worst case) would we need to find a given key? (Hint: solve 10 c. before doing 10 d.). 2 (6 Points)

11. Red Black Trees

8

(a) Insert the following numbers, one by one, into a Red-Black Tree. Show the tree after each insertion. Draw RED nodes with a circle and BLACK nodes with a rectangle. (5 points)

10,9,8,7,6,100



(b) When trying to analyze the run time complexity of an inorder traversal of a Red Black tree, one should consider the worst case, average case and ~~best~~ case separately. Each case may have a different run time performance. Circle TRUE or [✓ FALSE] (1 point)

(c) What is the best-case runtime complexity of a Red Black Tree insert operation? Use Big Theta notation. (2 points) $\Theta(\log N)$

0

Key

12. Dynamic Programming. (6 Points)

6

A top-down approach to computing n ! might look something like this :

```
int fact(int n) {
        if (n == 0) return 1 ;
        else return n * fact(n-1) ;
}
```

~~1 1 2 3 5 8~~

Write a method to compute n ! using a bottom-up approach.

~~INT fact(INT N) {~~
~~IF N == 0 || N == 1 return 1;~~
~~// N ≥ 2~~
~~K = 1~~
~~J = 1~~
~~t = 2~~
~~while t <= N {~~
~~sum = K + J~~
~~K = J~~
~~J = sum~~
~~t = t + 1~~
~~}~~
~~return sum~~

1	1	2	6	24	120
0	1	2	3	4	5

```
INT fact(INT N) {
    IF N == 0 || N = 1 return 1
    i = 2
    F = 1
    while (i ≤ N) {
        F = F * i
        i = i + 1
    }
    return F;
```

i	N	F
~~2~~	5	~~1~~
~~3~~		~~2~~
~~4~~		~~6~~
~~5~~		~~24~~
		~~120~~

i	N	F
~~2~~	5	~~1~~
~~3~~		~~2~~
~~4~~		~~6~~
~~5~~		~~24~~

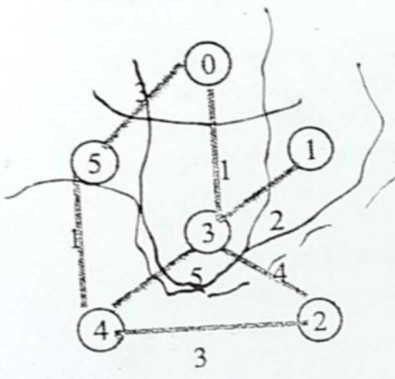
Key

Graph Algorithms(25 points)

13. (a) Consider the weighted and undirected graph below. When starting at vertex 0, breadth first search (BFS) could be used to enumerate (visit) each of the vertices. Show an ordering that BFS might produce. Starting from vertex 0, visit all the nodes. List the nodes as BFS would visit.
(3 points) 0, 5, 3, 1, 4, 2

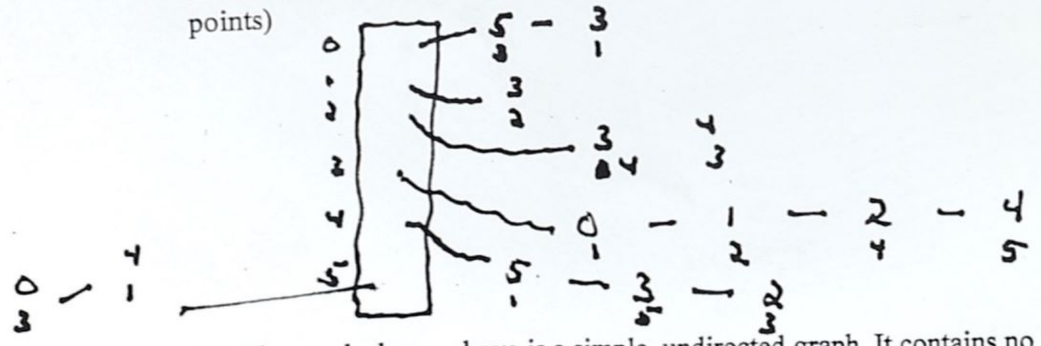
(b) If we want to learn if a path exists between vertex A and vertex B, we can simply run either BFS or DFS to find out. Circle [✓ True] or False. (3 points)

(c) Draw the contents of the distance array for each iteration of Prim's Algorithm as it works on this graph. The initial state is given. Mark the node to be selected next to the left of the array (note how 0 is marked to the left of the first array.) Fill in each array cell working downward. (8 points)

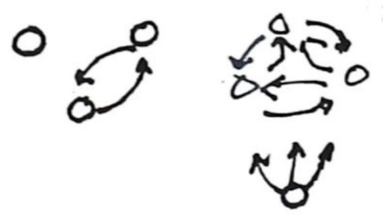


	0	1	2	3	4	5
0	0	?	?	?	?	?
3	0			1		3
1	0//	2	4	1//	5	3
5	0//	2//	4	1//	5	3
4	0//	2//	4	1//	1	3//
2	0//	2//	3	1//	1//	3//

(d) Draw an adjacency list representation of the graph shown immediately above. You need to think about how you will represent the distances associated with the edges. Draw a very clear picture of this array of linked lists. The picture that you draw will make it clear how the edge weights are being represented. (2 points)



(e) The graph shown above is a simple, undirected graph. It contains no loops or multiple edges. Suppose instead that we have a simple <u>directed</u> graph G with no loops or multiple edges. Suppose too that G has V vertices. What is the maximum number of edges that such a graph can contain (in terms of V)? Your answer should be an exact formula. Answers in Big Theta notation don't count. (1 point) $V \cdot (V-1)$



V	edges
1	0
2	2
3	6
4	12
V	$V \cdot (V-1)$

Key

(f) Using the Floyd Warshall algorithm shown below, draw a matrix each time the comment "draw cost matrix" is encountered in the code. Use the graph below for input. You may assume that the original graph is represented in an adjacency matrix c. Also, you must assume that the loops proceed through the graph in numerical order. That is, the first vertex is vertex 0 and the second is vertex 1 and so on. (8 points)

```
for each vertex u in G do {
  for each vertex v in G do {
    cost[u,v] = c[u,v]
  }
}
```

// Now, draw the cost matrix in the first box

```
for each vertex w in G do {
  for each vertex u in G do {
    for each vertex v in G do {
      cost[u,v] = min(cost[u,v],cost[u,w] + cost[w,v]
    } // end for v
  } // end for u
  // Now, draw the cost matrix in the next box
} // end for w
```

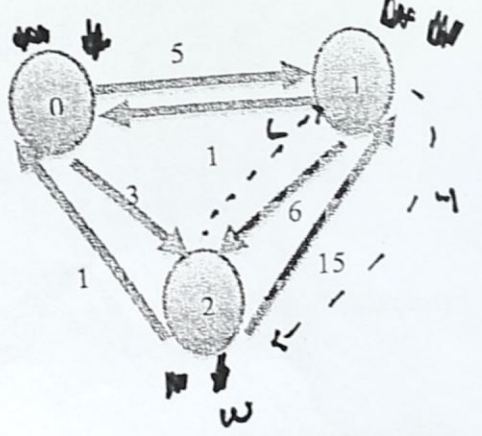
Cost

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 0 & 5 & 3 \\ 1 & 1 & 0 & 6 \\ 2 & 1 & 15 & 0 \end{array}$$

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 0 & 5 & 3 \\ 1 & 1 & 0 & 4 \\ 2 & 1 & \cancel{5}\ 6 & 0 \end{array}$$

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 0 & 5 & 3 \\ 1 & 1 & 0 & 4 \\ 2 & 1 & 6 & 0 \end{array}$$

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 0 & 5 & 3 \\ 1 & 1 & 0 & 4 \\ 2 & 1 & 6 & 0 \end{array}$$



w,
O

O
v

O u