95-771 Data Structures and Algorithms
Fall 2011

100/100 PLEASE RECORD ON BB

Midterm Exam

Lists (15 points)

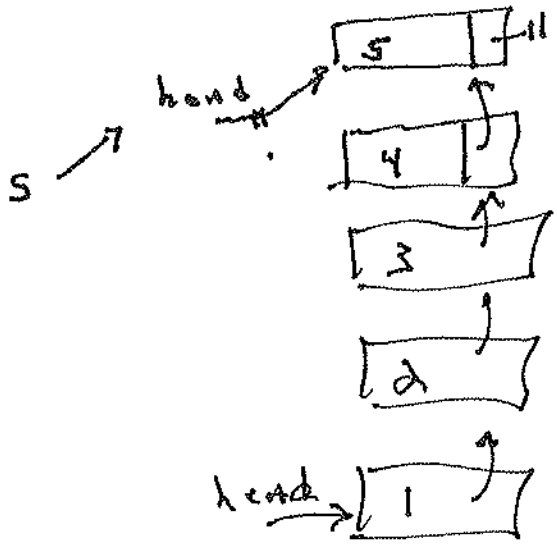
Name KEY

1. Consider the following Node and List classes.

```
class Node {

  private Object data;
  private Node next;
  public  Node(Object d, Node n) { data = d; next = n; }
  public  Node getNext() { return next; }
  public  Object getData() { return data; }
  public  void setNext(Node n) { next = n; }
  public  void setData(Object x) { data = x; }
  public  String toString() {
          String t = "";
          t = t + data;
          if(next == null) t = t + ">>>";
          else t = t + "---";
          return t;
        }
}

public class List {

  private Node head;
  public List() { head = null; }
  public void add(Object t) { head = new Node(t,head); }
  public Object remove() {
          Object t = head;
          head = head.getNext();
          return t;
  }
  private Node process(Node ptr) {
                  Node t;
                  Node p = null;
                  while(ptr != null) {
                          t = ptr.getNext();
                          ptr.setNext(p);
                          p = ptr;
                          ptr = t;
                  }
                  return p;
        }
        public void process() {
                  head = process(head);
  }
  public boolean isEmpty() { return head == null; }
  public String toString() {
          String result = "";
          Node p = head;
          while(p != null ) {
                          result = result + p;
             p = p.getNext();
          }
          return result;
  }
```

NUMBER WRONG ON PAGE 4

```
  public int listSize() {
      int count = 0;
      Node p = head;
      // Complete the method. See question 1 e.




      return count;
  }

  public static void main(String a[]) {

      List s = new List();
      for(int j = 5; j >= 1; j--) s.add(new Integer(j));
      // 1
      System.out.println(s);

      s.process();

      // 2
      System.out.println(s);

      s.process();

      // 3
      while(!s.isEmpty())
          System.out.print(s.remove());
  }
}
```



Key

Question 1.

(a) The program compiles and executes without error. What is the exact output of the print statement marked in the code with a 1? (3 Points)

1 --- 2 --- 3 --- 4 --- 5 >>>

(b) The program compiles and executes without error. What is the exact output of the print statement marked in the code with a 2? (3 Points)

5 --- 4 --- 3 --- 2 --- 1 >>>

(c) The program compiles and executes without error. What is the exact output of the print statements marked in the code with a 3? (3 Points)

1 -- 2 -- 3 -- 4 -- 5 >>>

(d) What is the run time complexity of the process() method in terms of Θ.(1 Point) Θ(N)

(e) Complete the listSize() method. You will begin work from the declarations already present. Currently the method is not correct. You need to fill in the correct Java code. Write your solution right here on the exam. (3 Points)

```
public int listSize() {
    int count = 0;
    Node p = head;
    // Complete the method here
    while (p != null){
        count++;
        p = p.getNext();
    }
    return count;
}
```

(f) Is it correct to say that your listSize() method is O(2^N)?
Circle Yes or No. (1 point)

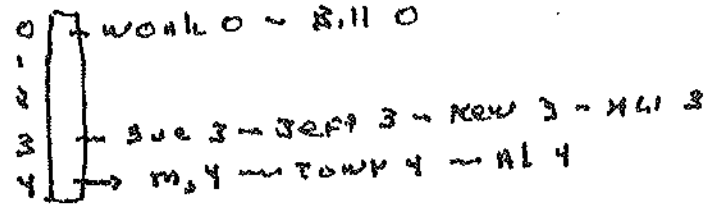
(g) Is it correct to say that the toString() method of the List class is Ω(1) ?
Circle Yes or No. (1 point)

Note - 0
The ↓
-0

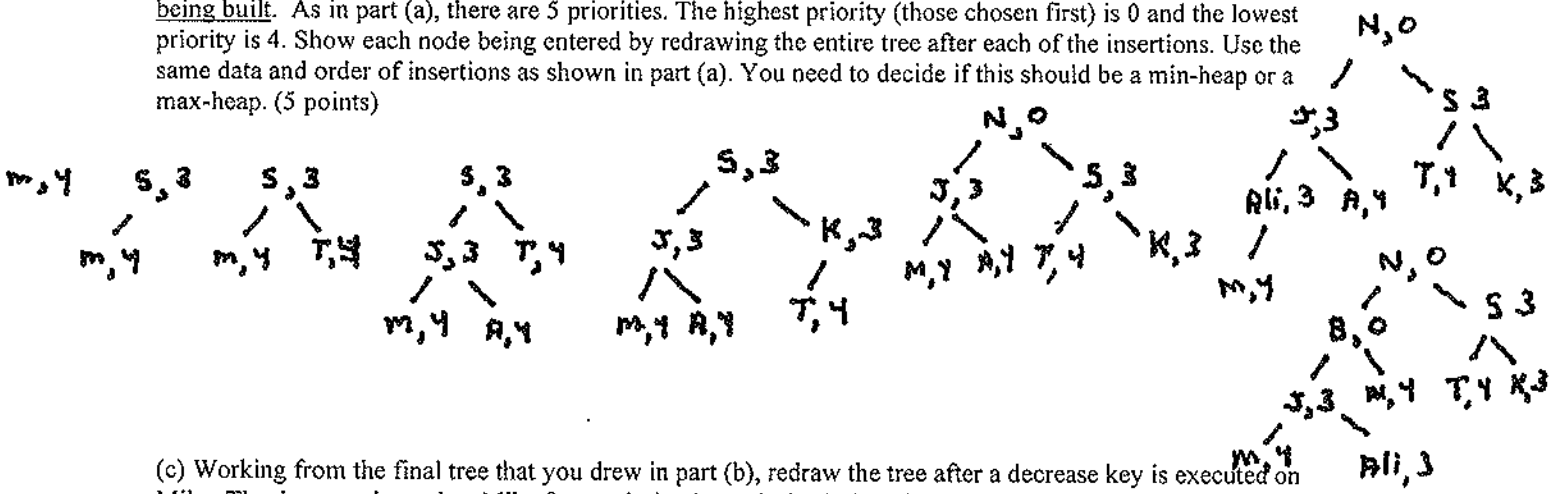
Key

Priority Queues: (15 points)

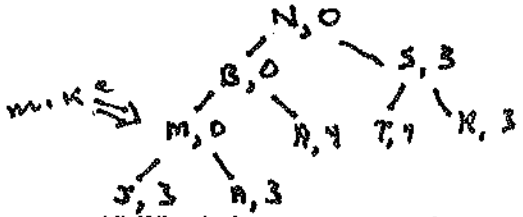
2) (a) Sue has implemented a priority queue as an array of linked lists. Each list is a queue and has an enQueue and deQueue method. These methods run in θ(1). The enQueue method adds data at the rear of the queue and the deQueue method removes data from the front of the queue. Each array cell holds a pointer to the front of the list and a pointer to the rear of the list. There are 5 priorities and so the array has 5 elements. The highest priority is 0 and the lowest is 4. Draw a picture of this data structure after the following data (with priorities shown) has been entered: Mike 4, Sue 3, Tony 4, Jeff 3, Al 4, Ken 3, Noah 0, Ali 3, Bill 0. (5 Points)



(b) Bill has implemented a priority queue as a heap in an array. Draw a picture of the heap as a tree as it is being built. As in part (a), there are 5 priorities. The highest priority (those chosen first) is 0 and the lowest priority is 4. Show each node being entered by redrawing the entire tree after each of the insertions. Use the same data and order of insertions as shown in part (a). You need to decide if this should be a min-heap or a max-heap. (5 points)



(c) Working from the final tree that you drew in part (b), redraw the tree after a decrease key is executed on Mike. The decrease key takes Mike from priority 4 to priority 0. (3 points)



(d) What is the worst-case runtime complexity of the decrease key operation? Use Big Theta. (1 Points)

Θ(log N)

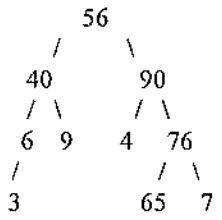
(e) What is the best-case runtime complexity of the decrease key operation? Use Big Theta. (1 Points)

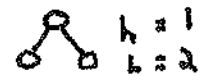
Θ(1)

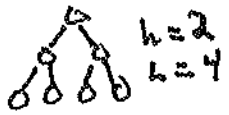
key

Trees (20 points)

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







(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)

56, 40, 6, 3, 9, 90, 4, 76, 65, 7

(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)

3, 6, 40, 9, 56, 4, 90, 65, 76, 7

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

3, 6, 9, 40, 4, 65, 7, 76, 90, 56

(d) In general, if a binary tree is perfectly balanced (unlike the tree pictured here) and the tree is of height h then how many leaves will the tree have? (2 points) 2^h

(e) In general, if a ternary tree (with a maximum of three children per node) is perfectly balanced and the tree is of height h then how many leaves will the tree have? (2 points) 3^h

(f) What is the runtime complexity of the in-order traversal if the tree is perfectly balanced? Use Big Theta(1 points) $\Theta(N)$

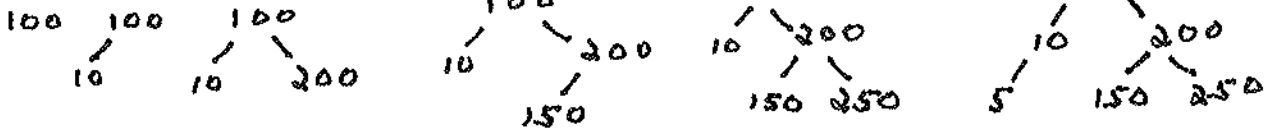
(g) What is the runtime complexity of the level-order traversal if the tree is perfectly balanced? Use Big Theta(1 point)

$\Theta(N)$

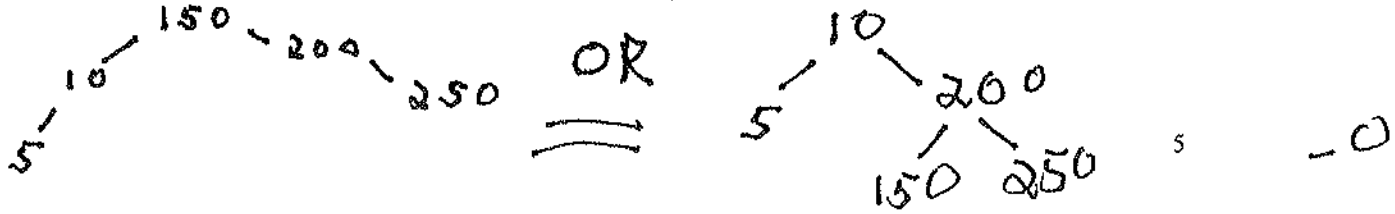
(h) In general, if a binary tree is perfectly balanced (unlike the tree pictured here) and the tree has k leaves (nodes with no children) then how many internal nodes will the tree have? (2 points)

K - 1

4. (a) Insert he following numbers into a Binary Search Tree. Draw the tree after each insertion. (3 Points)
100, 10, 200, 150, 250, 5



(b) Delete 100 from the final tree that you drew in 4 (a). Draw this final tree. (2 Points)



-0

KEY

Project Questions (10 points)

5. Recall the Merkle-Hellman cryptosystem that we worked with in Project 1 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 = {4, 16, 3, 9, 2, 8, 13, 100, 41, 8} and **k** = 36. Is there a subset of X which sums to k?
Yes (Yes/No (2 points)

(b) Suppose Alice sends messages to Bob encrypted with Bob's Merkle-Hellman public key. Circle the one statement that is true? (2 Points)

1. Alice encrypts with a super increasing sequence.
2. Alice decrypts with a super increasing sequence.
3. Bob decrypts with a super increasing sequence.
4. Bob decrypts with a set such as X in part a.
5. Alice decrypts with a set such as X in part a.

(c) Write a method in Java that returns true if there is a subset of X which sums to k and false otherwise. The method signature and pre-conditions are provided: (4 points)

```
public static boolean subSetSum(int[] x, int n, int k) {
// pre: x is a superincreasing sequence with x[0] < x[1]<x[2] <...<x[n-1].
// pre: n is the size of x.
// pre: all integers involved are small enough that overflow is not of concern
// post: returns true if some subset of x sums to k, false otherwise.
```

```
FOR ( J = N-1 ; J >= 0 ; J-- ) {
    IF ( X[J] <= K ) K = K - X[J]
}
return K == 0
```

Project 3 was written to make good use of a Red Black Tree.

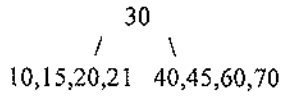
(d) The tree was used for (circle one answer). (2 Points)

1. Maintain a stack of variable names.
2. Maintain a stack of values.
3. Maintain a set of variables quickly searched by value.
4. Maintain a set of name value pairs quickly searched by name.
5. Maintain a set of name value pairs quickly searched by value.

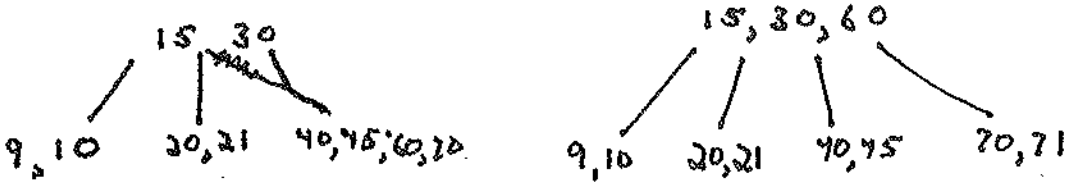
Key

Balanced Search Trees (15 points)

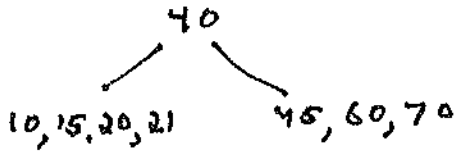
6. Consider the following B-Tree with a minimum of two and a maximum of four.



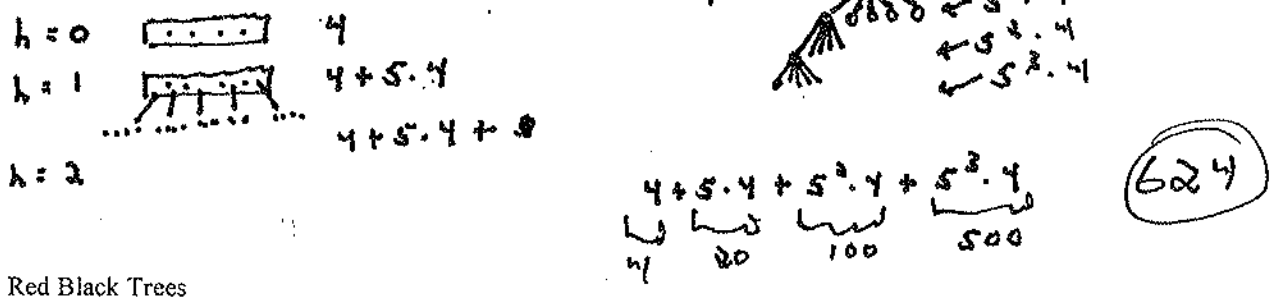
(a) Redraw the tree after inserting 9 and then 71. (3 points)



(b) Delete the value 30 from the B-Tree shown above. Begin work from the original tree, not the tree with the values 9 and 71. (2 points)



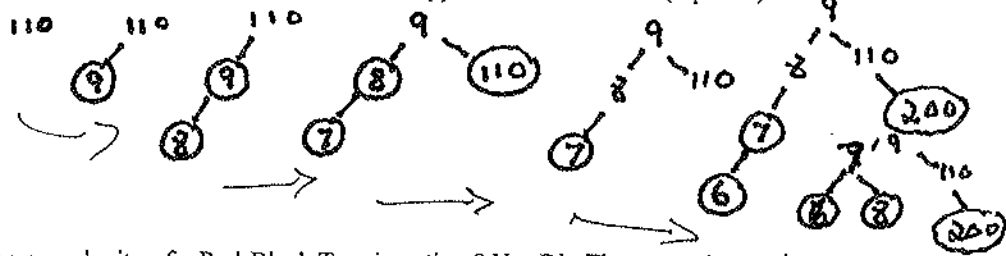
(c) The tree in question 6a has a height of 1. What is the maximum number of keys that this type of tree (min = 2, max=4) could hold with a height of 3? (3 points).



7. Red Black Trees

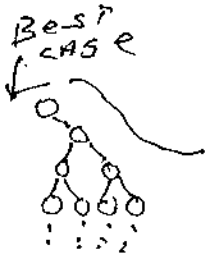
(a) Insert the following numbers, one by one, into a Red-Black Tree. Show the tree after each insertion. red vertices should be circled and black vertices should appear without circles. (5 points)

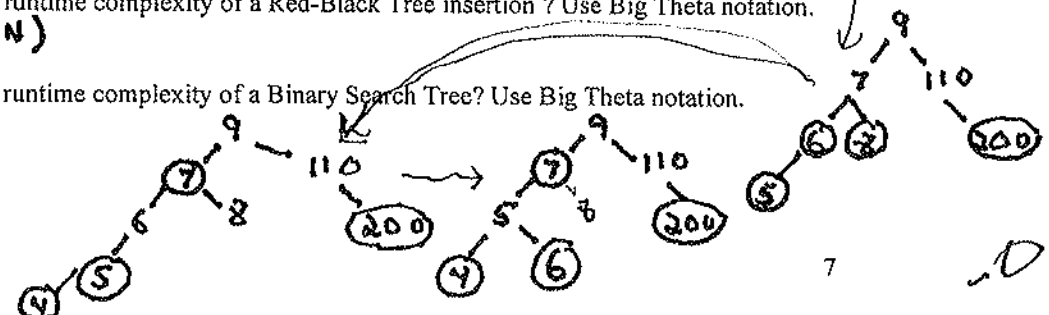
110, 9, 8, 7, 200, 6, 5, 4



(b) What is the best-case runtime complexity of a Red-Black Tree insertion ? Use Big Theta notation. (1 Point) $\Theta(\log N)$

(c) What is the best-case runtime complexity of a Binary Search Tree? Use Big Theta notation. (1 point) $\Theta(1)$





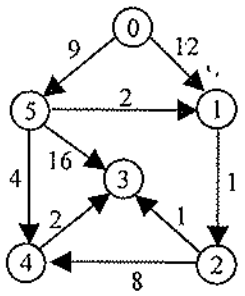
10

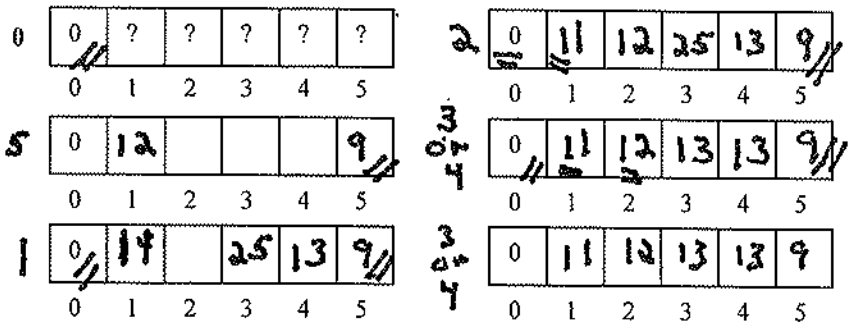
KEY

Graph Algorithms(25 points)

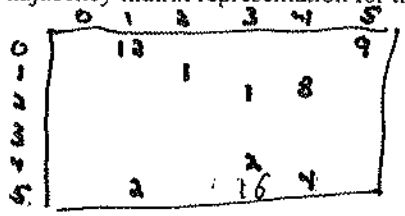
8. (a) What is the distance of the shortest path from node 0 to node 2 in the graph immediately below? Your path must be a list of ordered pairs.(1 point) (0,5), (5,1), (1,2)

(b) Draw the contents of the distance array for each iteration of Dijkstra'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. That is, complete the first column of arrays before the second column of arrays. (5 Points)

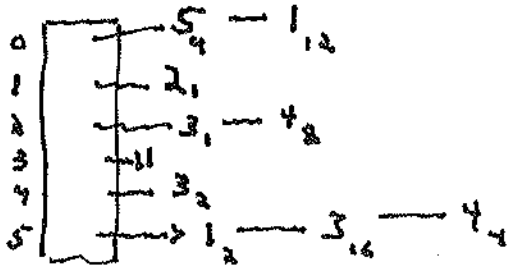




(c) Draw an adjacency matrix representation for the graph shown above. (2 points)



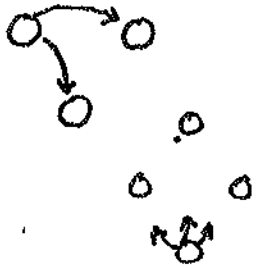
(d) Draw an adjacency list representation of the graph shown immediately above. (2 Points)



(e) The graph shown above is a simple, directed graph. It contains no loops or multiple edges. Suppose we have such a graph with V vertices. What is the maximum number of edges that such a graph can contain? Your answer should be an exact formula. Answers in Big Theta notation don't count. (1 Point)

V · (V - 1)

$V^2 - V$

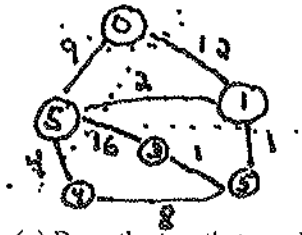


V	edges = E
1	0
2	2
3	3 · 2 = 6
4	4 · 3 = 12
V	V · (V-1)

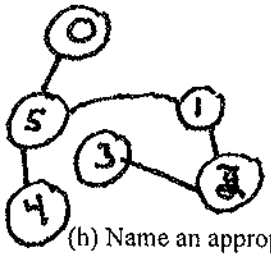
-0

Key

(f) Redraw the graph in question 8 b as an undirected graph. That is, replace each arrow by an undirected edge. The weight of each edge in the undirected graph will be the same as the respective edge weight in the directed graph of 8 b. (1 Point)



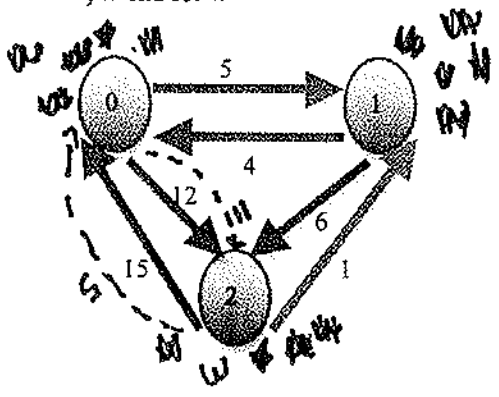
(g) Draw the tree that would be computed by Prim using the graph that you drew in question 8 f as input. This is the tree computed by Prim of the undirected, weighted graph you created in 8 f. (5 points)



(h) Name an appropriate and fast algorithm to determine whether or not there is a path from one vertex to another in a directed or undirected graph. Either BFS or DFS (2 points)

(i) 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. (6 Points)

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

// draw 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
  // draw the cost matrix in the next box
} // end for w
```



Cost

	0	1	2
0	0	5	12
1	4	0	6
2	15	1	0

	0	1	2
0		5	12
1	4		6
2	15	1	

	0	1	2
0		5	11
1	4		6
2	5	1	

	0	1	2
0		5	11
1	4		6
2	5	1	

No change

D