CST 370 Design and Analysis of Algorithms
Spring A 2020
Final Exam

Name: Matthew Robertson

Four-digits Class ID: 1618

---

1. (10 points) Choose one of the 4 options for each question.

(a) Let P be a QuickSort Program to sort numbers in ascending order using the first element as pivot. Let t1 and t2 be the number of comparisons made by P for the inputs {4, 1, 5, 3, 2} and {1, 2, 3, 4, 5} respectively. Which one of the following holds?

(A) t1 = 5
(B) t1 < t2
(C) t1 > t2
(D) t1 = t2

(B) t1 < t2

(b) Consider the following code snippet in C++. The function BSTk() receives root of a **Binary Search Tree (BST)** and a positive integer k as arguments. In the "node" data structure, "left" points to the left child and "right" points to the right child.

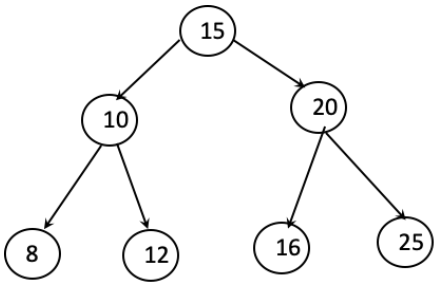
```
// A BST node
struct node {
    int data;
    struct node *left, *right;
};

int count = 0;

void BSTk(struct node *root, int k)
{
    if (root != NULL && count <= k)
    {
        BSTk(root->right, k);
        count++;
        if (count == k)
          printf("%d ", root->data);
```

```
        BSTk(root->left, k);
    }
}
```

What is the printed output of BSTk(root, 2) where root represents the root of the following BST?

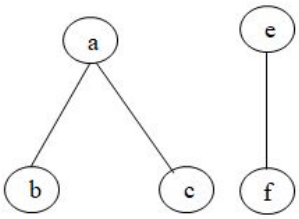


(A) 10
(B) 16
(C) 20
(D) 20 10

(B) 16

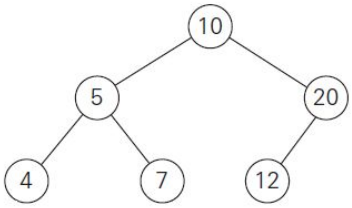
2. (10 points)

(a) Based on our class's definition, this is a graph. (True/False)



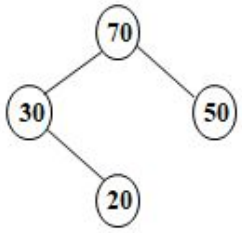
True, a graph needn't be connected.

(b) This is an AVL tree. (True/False)



True,

(c) This is a max heap. (True/False)



False, the tree should be filled, there is a missing node as the left child of 30 that should be populated before its right child.

3. (5 point) Consider the following master theorem:
$T(n) = aT(n/b) + f(n)$ where $f(n) \in \Theta(n^d)$, $d \geq 0$

Master Theorem: If $a < b^d$, $T(n) \in \Theta(n^d)$ *Case 1*
If $a = b^d$, $T(n) \in \Theta(n^d \log n)$ *Case 2*
If $a > b^d$, $T(n) \in \Theta(n^{\log_b a})$ *Case 3*

Based on the theorem, determine the time efficiency of the following formula T(n).
You need to indicate the values for a, b, and d. Also, indicate which case of the Master Theorem applies.

(a) $T(n) = 2 * T(n/2) + n^4$

a = 2
b = 2
d = 4
a ? b^d ➜ $2 < 2^4$ ➜ Case 1
$T(n) \in \Theta(n^d) = \Theta(n^4)$

(b) $T(n) = 16 * T(n/4) + n^2$

a = 16
b = 4
d = 2
a ? b^d ➜ $16 = 4^2$ ➜ Case 2
$T(n) \in \Theta(n^d \log n) = \Theta(n^2 \log n)$

4. (5 points) What is the time complexity of the following function fun()? Choose one of the 4 options and describe what the basic operation is here.

```
int fun(int n)
{
  int count = 0;
  for (int i = n; i > 0; i /= 2)
     for (int j = 0; j < i; j++)
        count += 1;
  return count;
}
```

(A) O(nLog(n))
(B) O(n)
(C) $O(n^2)$
(D) O(nLog(n)Log(n))

(C) the time complexity is $O(n^2)$, given that:

- $n + n/2 + n/4 + ... 1 = O(n)$
- inner loops that cap on the outer loop's iterator are expanded into a $n + n/2^2 .. n/2^i$ sequence.

The basic operation is the line:

```
count += 1;
```

5. (10 points) Suppose you have three jars, A, B, and C, in a room. Jar A has 5 large black balls, 3 large red balls, and 2 large green balls. Jar B has 4 small black balls, 3 small red balls, and 2 small green balls. Jar C is empty. Thus, there are **total 19 balls**. Now, you will pick a few balls from the jar A in the dark and place them in the jar C. After that, you will pick a few balls from the jar B in the dark and place them in the jar C. Note that the color of the selected balls at the jars A and B can not be confirmed because the surroundings are dark. Also, the numbers of balls selected from the jars A and B need not always be the same. Once you're done, you can turn on the lights in the room and see the balls in the jar C.

(a) Assuming the **worst case occurs**, what is the minimum number of balls you have to choose to **get a matching pair**? Here, a matching pair means that there must be one large ball and one small ball of the same color in the jar C. But the **color** itself of the pair **is not important**. Present the total number of balls chosen and how they are chosen (number of balls from each jar)

In the worst case, 10 balls are chosen, 9 from jar A and 1 from jar B.
Those balls from jar A would be: 5 large black, 3 large red and 1 large green.
Those balls from jar B would be: 1 small green.

(b) Assuming the **best case occurs**, what is the minimum number of balls you have to choose to **get three matching pairs of each color (= black, red, green)**? In other words, you should have one pair of large and small black balls, one pair of large and small red

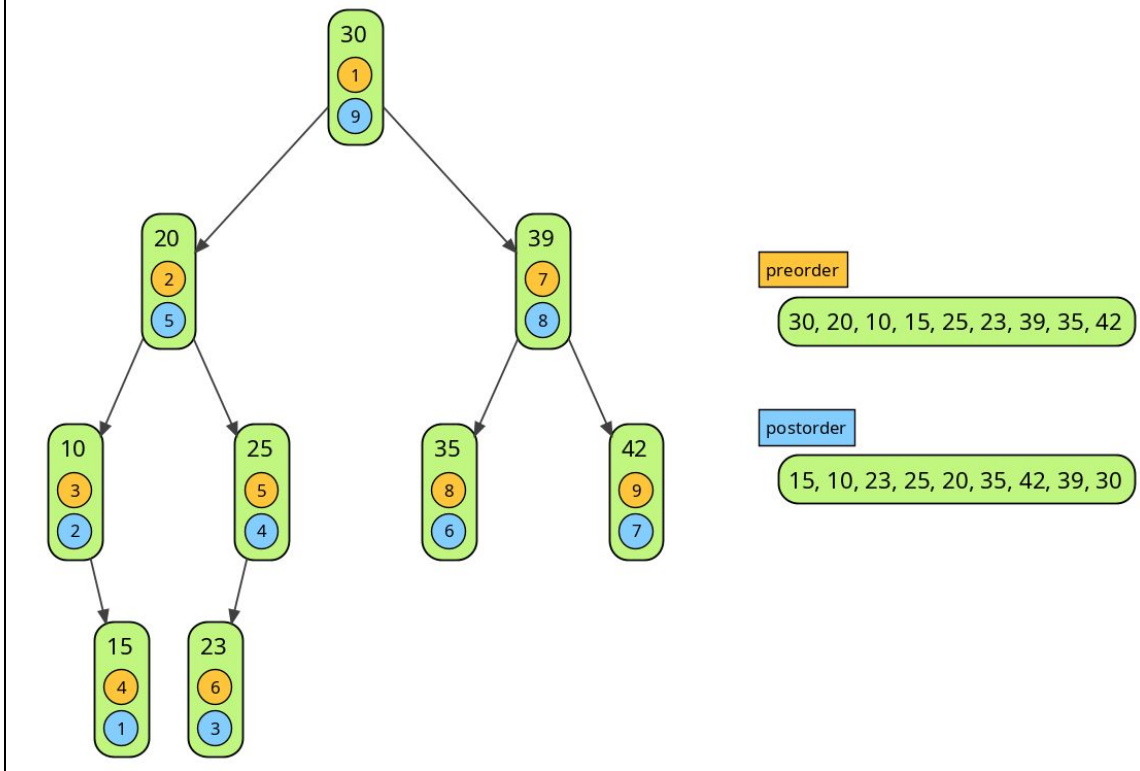
balls, and one pair of large and small green balls. Present the total number of balls chosen and how they are chosen (number of balls from each jar)

In the best case, 6 balls are chosen, 3 from jar A and 3 from jar B. Those balls from jar A would be: 1 large black, 1 large red and 1 large green. Those balls from jar B would be: 1 small black, 1 small red and 1 small green.

6. (5 points) The preorder traversal sequence of a binary search tree is 30, 20, 10, 15, 25, 23, 39, 35, 42. Which one of the following is the postorder traversal sequence of the same tree? Explain your answer.

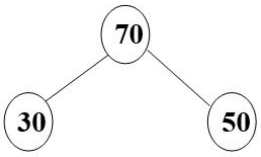
a) 10, 20, 15, 23, 25, 35, 42, 39, 30
b) 15, 10, 25, 23, 20, 42, 35, 39, 30
c) 15, 20, 10, 23, 25, 42, 35, 39, 30
d) 15, 10, 23, 25, 20, 35, 42, 39, 30

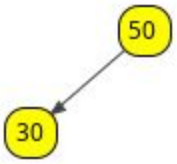
The postorder traversal sequence of the tree is:
d) 15, 10, 23, 25, 20, 35, 42, 39, 30



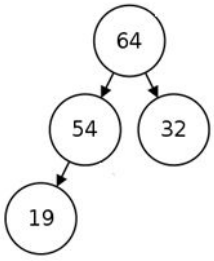
7. (10 points)

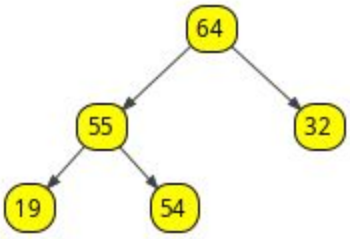
(a) Remove the max value from the following heap. Perform necessary operations to make it a heap again (heapify). Only need to present the final tree which should be a heap.



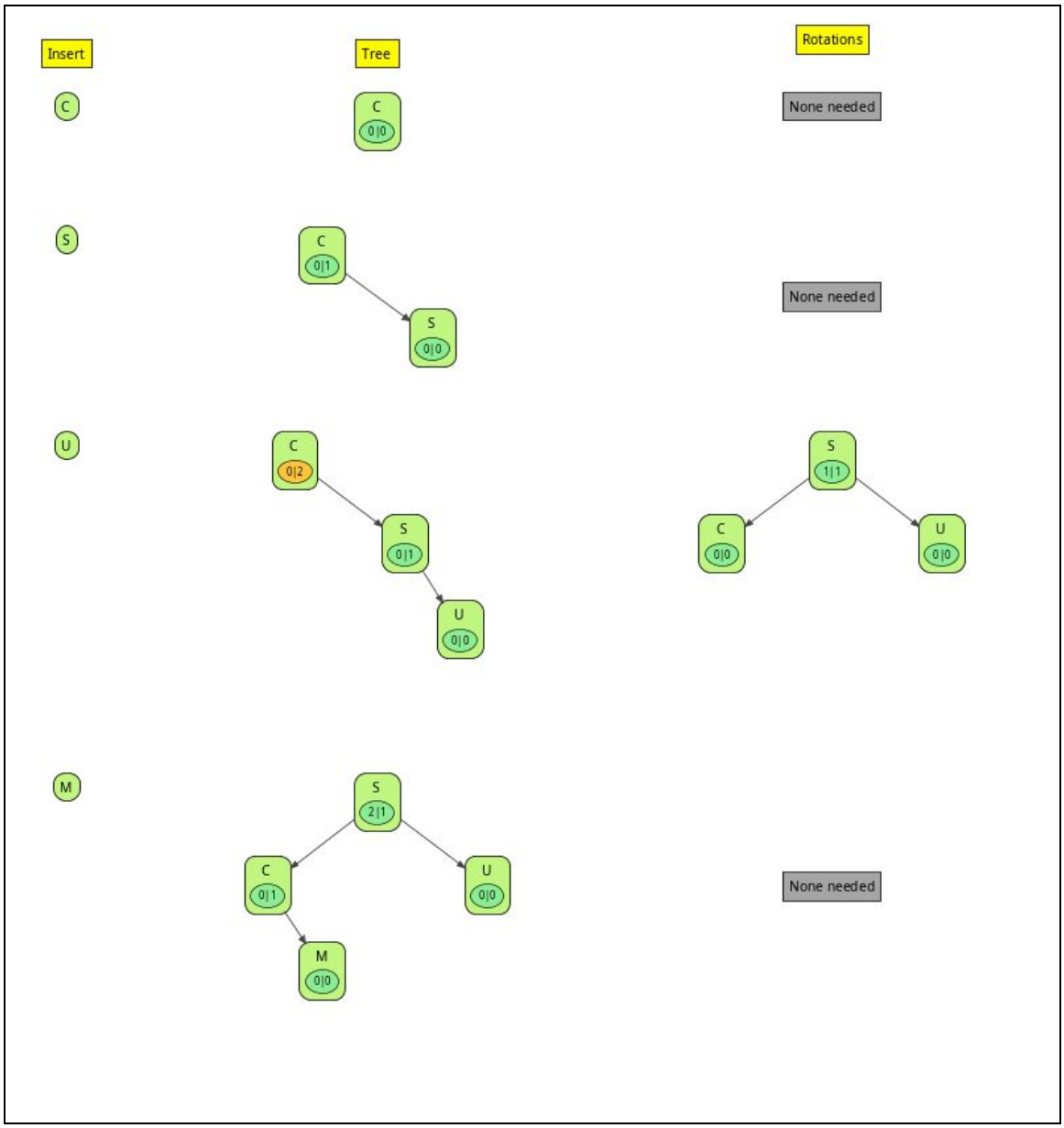


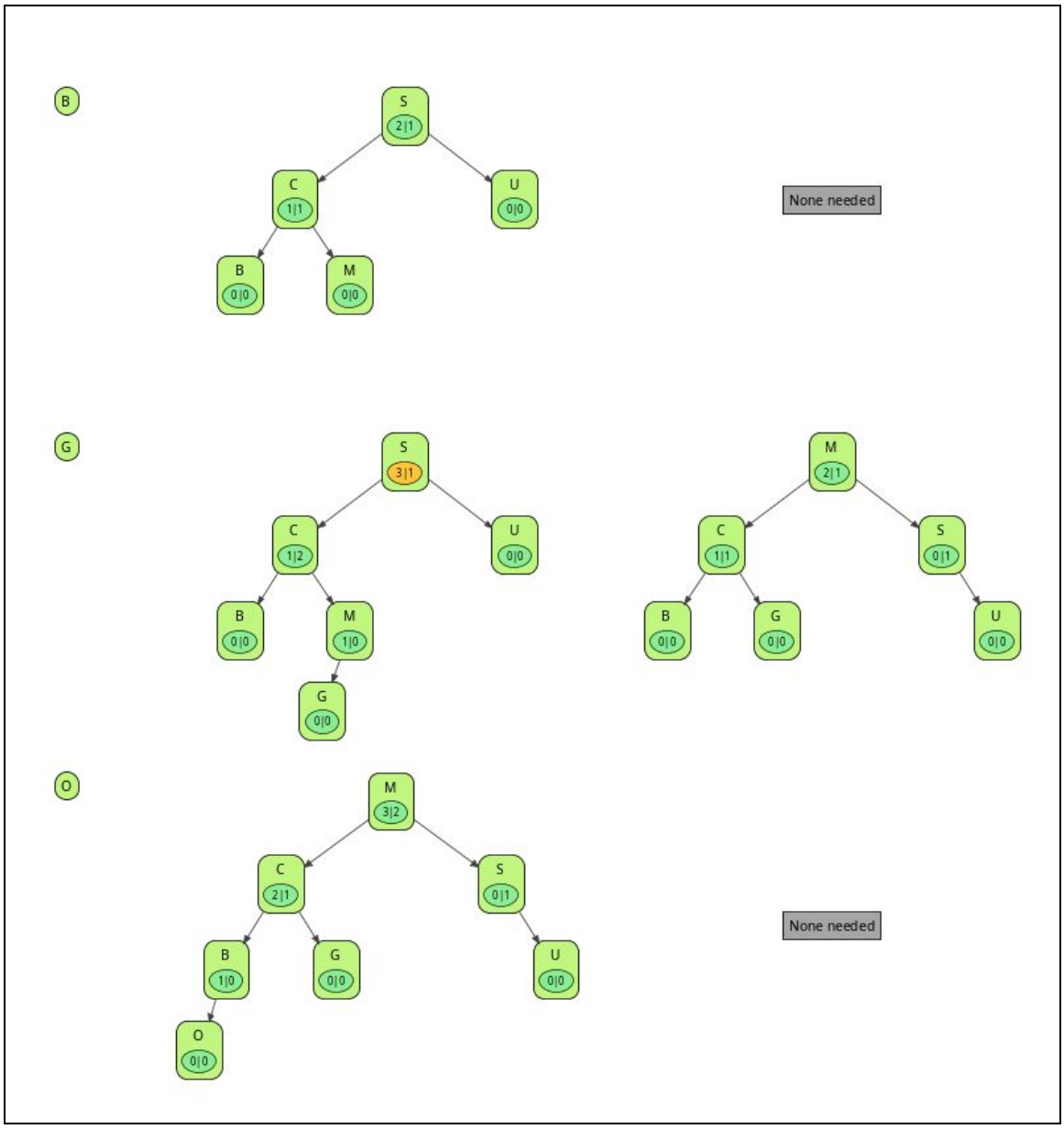
(b) Add 55 to the following heap. Perform necessary operations to make it a heap again (heapify). Only need to present the final tree which should be a heap.



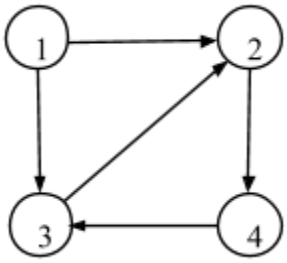


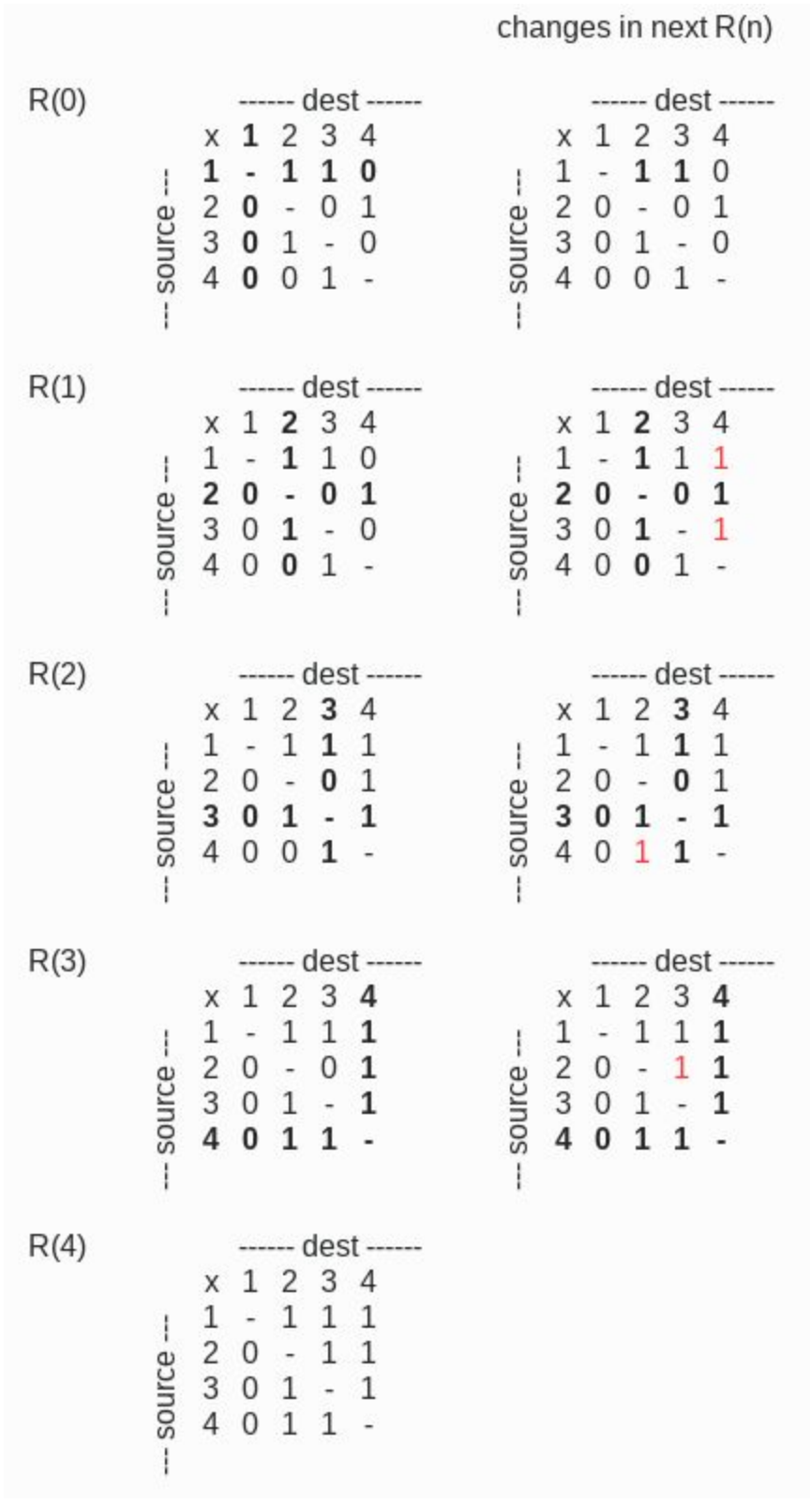
8. (5 points) Construct an AVL tree for the list **C, S, U, M, B, G, O**. Insert each letter successively starting with the empty tree. Use alphabetical order when inserting a letter. Your answer should present the rotation operations (if necessary) for each letter insertion.


Insert
Tree
Rotations
C
C
0|0
None needed
S
C
0|1
S
0|0
None needed
U
C
0|2
S
0|1
U
0|0
S
1|1
C
0|0
U
0|0
M
S
2|1
C
0|1
U
0|0
M
0|0
None needed

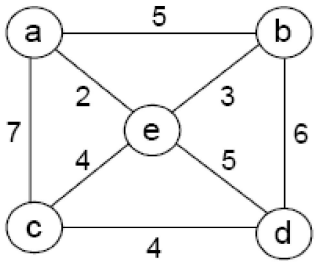


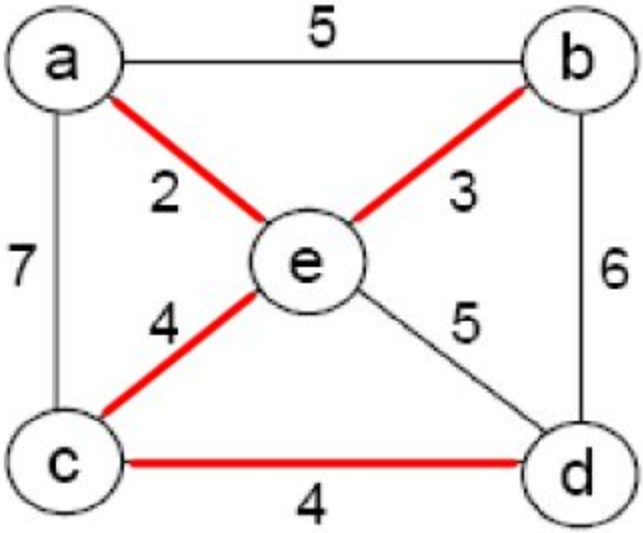
9. (10 points) Apply the Warshall's algorithm to get the transitive closure of the digraph defined by the following graph. Present $R^{(0)}$, $R^{(1)}$, $R^{(2)}$, $R^{(3)}$, and $R^{(4)}$ as we discussed in the class.



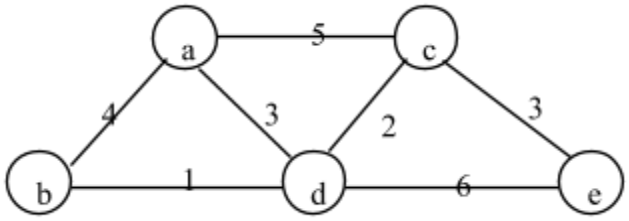

changes in next R(n)
R(0)
--- source ---
------ dest ------
x 1 2 3 4
1 - 1 1 0
2 0 - 0 1
3 0 1 - 0
4 0 0 1 -
--- source ---
------ dest ------
x 1 2 3 4
1 - 1 1 0
2 0 - 0 1
3 0 1 - 0
4 0 0 1 -
R(1)
--- source ---
------ dest ------
x 1 2 3 4
1 - 1 1 0
2 0 - 0 1
3 0 1 - 0
4 0 0 1 -
--- source ---
------ dest ------
x 1 2 3 4
1 - 1 1 1
2 0 - 0 1
3 0 1 - 1
4 0 0 1 -
R(2)
--- source ---
------ dest ------
x 1 2 3 4
1 - 1 1 1
2 0 - 0 1
3 0 1 - 1
4 0 0 1 -
--- source ---
------ dest ------
x 1 2 3 4
1 - 1 1 1
2 0 - 0 1
3 0 1 - 1
4 0 1 1 -
R(3)
--- source ---
------ dest ------
x 1 2 3 4
1 - 1 1 1
2 0 - 0 1
3 0 1 - 1
4 0 1 1 -
--- source ---
------ dest ------
x 1 2 3 4
1 - 1 1 1
2 0 - 1 1
3 0 1 - 1
4 0 1 1 -
R(4)
--- source ---
------ dest ------
x 1 2 3 4
1 - 1 1 1
2 0 - 1 1
3 0 1 - 1
4 0 1 1 -

10. (10 points) Assume that you are going to solve the **MST** (Minimum Spanning Tree) problem using the **Prim's algorithm** for the following graph. Draw the **final MST**. For the problem, you have to start from the vertex **a**. You must also provide the **sequence of vertices** to be added to the "visited" set as we covered in the class.



Sequence of vertices to be added: a-e (2), e-b (3), e-c (4), c-d (4)


11. (10 points) Assume that you are going to solve the single-source shortest-paths problem using the **Dijkstra's algorithm** for the following graph. For the problem, you should start from the vertex **a**. Fill out the table as you learned in the class.



V	a	b	c	d	e
a	0_a	4_a	5_a	3_a	∞
d		$3+1=4_d$	$3+2=5_d$	3_a	$3+6=9_d$
b		4_d	5_d		9_d
c			5_d		$5+3=8_c$
e					8_c

12. (10 points) Assume that you want to calculate 2^n where n is a positive integer using a **decrease-and-conquer** algorithm. For example, if the algorithm receives an input value 3 for n, it should return 8 (= 2^3).

(a) Describe the basic idea of your decrease-and-conquer algorithm **in English**.

To decrease and conquer this exponentiation problem, one would recursively multiply 2 by 2 for n-1 iterations.
So:
2^3 would be: 2 * (2 * 2)
2^4 would be 2 * (2 * (2 * 2))
and so on.

(b) Based on the basic idea of (a), **write a pseudocode** of your algorithm

```
function two_exp(n)
    return 2 * two_exp(n-1)
```