


**Exercise 4 (for grade)** ~ Wednesday, September 28, 2022 ~ CPSC 535 Fall 2022

Write one submission for your entire group, and write all group members' names on that submission. Turn in your submission before the end of class. The  symbol marks where you should write answers.

Recall that our recommended problem-solving process is:

1. **Understand** the problem definition. What is the input? What is the output?
2. **Baseline** algorithm for comparison
3. **Goal** setting: improve on the baseline how?
4. **Design** a more sophisticated algorithm
5. **Inspiration** (if necessary) from patterns, bottleneck in the baseline algorithm, other algorithms
6. **Analyze** your solution; goal met? Trade-offs?

Follow this process for each of the following computational problems. For each problem, your submission should include:

- a. State are the input variables and what are the output variables
- b. Pseudocode for your baseline algorithm, that needs to include the data type and an explanation for any variable other than input and output variables
- a. The  $\Theta$ -notation time complexity of your baseline algorithm, with justification.

and if you manage to create an improved algorithm:

- c. Answer the question: how is your improved algorithm different from your baseline; what did you change to make it faster?
- d. Pseudocode for your improved algorithm, that needs to include the data type and an explanation for any variable other than input and output variables
- a. The  $\Theta$ -notation time complexity of your improved algorithm, with justification.

Today's problems are:

1.

Given a binary search tree with  $n$  nodes ( $n > 3$ ) and a query value  $q$ , design an algorithm that returns NONE if the query value is not in the tree, or the depth of the node that holds the query value. The depth of the root node is 0.

Baseline: time complexity  $O(n)$ , improved to  $O(\text{height})$  time complexity

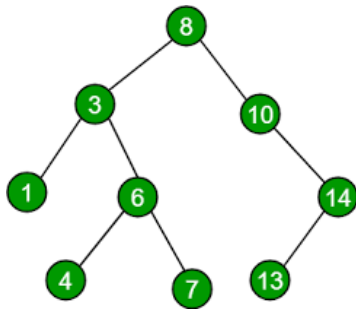


Fig. 1 Image taken from

<https://www.geeksforgeeks.org/check-if-given-sorted-sub-sequence-exists-in-binary-search-tree/>

- Example 1: Input: BST in Figure 1,  $q = 14$  Output: 2  
 Example 2: Input: BST in Figure 1,  $q = 7$  Output: 3  
 Example 3: Input: BST in Figure 1,  $q = 8$  Output: 0  
 Example 3: Input: BST in Figure 1,  $q = 9$  Output: NONE

2.

a) Run the Floyd-Warshall algorithm on the weighted, directed graph shown below. Show the matrix  $D^{(k)}$  that results for each iteration of the outer loop. Show your work at every step. I will deduct points if you do not show your work and you skip steps.

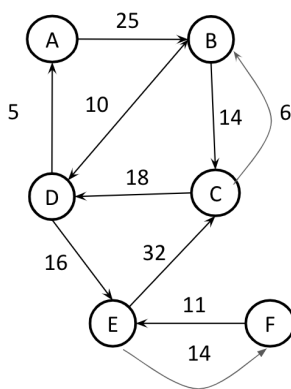


Fig. 2 Example of weighted, directed graph

b) State the dimensionality of the Floyd-Warshall algorithm as defined for a dynamic programming algorithm.

3.

Compute the transitive closure of the graph above. Show your work at every step. I will deduct points if you do not show your work and you skip steps.

## Names

Write the names of all group members below.

❌ Lency Lakhani

Hetal Patel

Vyshnavi Reddy

## Exercise 1: Solve and provide answer

❌

```
//Take input from the users//
Scanner sc=new scanner(System.in);
Int query=sc.nextInt();
System.out.println("enter your query to find the height);
```

```
BinarySearch(query){
height=0;
current=root;
if(query==current)
    Return current;
Else if(query<=current)
    current=current->left;
    Return current;
Else
    current=current->right;
    Return current;
Return NONE;
Do{
current.height=current.height+1
system.out.println(current.height);
}
while(current.isleaf)
```

incorrect algorithm; it returns the node, not its depth

### ALGORITHM

Step1:take the query input from the user.

Step2:assign root node as current node, if query is same as current node then assign query value to current.

Step3:if query value is less than the current node then go to the left on the tree,else go to the right on tree.

Step4:if query does not match with the current value the return NONE.

Step5:find the height of that query value so, increment the height variable till the leaf node

Step6: print the value of height till leaf node.

## Exercise 2: Solve and provide answer



Q2 (A) :-

K

	A	B	C	D	E	F
A	0	25	$\infty$	$\infty$	$\infty$	$\infty$
B	$\infty$	0	14	10	$\infty$	$\infty$
C	$\infty$	6	0	18	$\infty$	$\infty$
D	5	$\infty$	$\infty$	0	16	$\infty$
E	$\infty$	$\infty$	32	$\infty$	0	14
F	$\infty$	$\infty$	$\infty$	$\infty$	11	0

A

	A	B	C	D	E	F
A	0	25	$\infty$	$\infty$	$\infty$	$\infty$
B	$\infty$	0	14	10	$\infty$	$\infty$
C	$\infty$	6	0	18	$\infty$	$\infty$
D	5	30	$\infty$	0	16	$\infty$
E	$\infty$	$\infty$	32	$\infty$	0	14
F	$\infty$	$\infty$	$\infty$	$\infty$	11	0

B

	A	B	C	D	E	F
A	0	25	39	35	$\infty$	$\infty$
B	$\infty$	0	14	10	$\infty$	$\infty$
C	$\infty$	6	0	16	$\infty$	$\infty$
D	5	30	44	0	16	$\infty$
E	$\infty$	$\infty$	32	$\infty$	0	14
F	$\infty$	$\infty$	$\infty$	$\infty$	11	0

C

	A	B	C	D	E	F
A	0	25	39	35	$\infty$	$\infty$
B	$\infty$	0	14	10	$\infty$	$\infty$
C	$\infty$	6	0	16	$\infty$	$\infty$
D	5	30	44	0	16	$\infty$
E	$\infty$	38	32	48	0	14
F	$\infty$	$\infty$	$\infty$	$\infty$	11	0

(-0.75 points): last step is missing; there are 6 nodes not 3

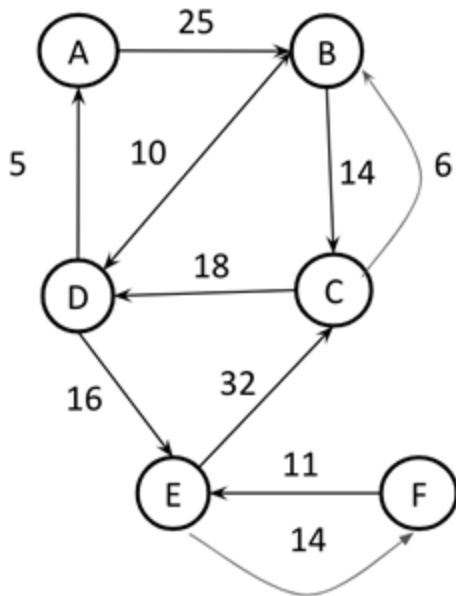
B. floyd warshall algorithm is used derive the shortest path between matrix. There are distance in between the node i and node j. clgo is used to derived weight between nodes does not include negative weight. If there is no distance then it is count as infinite. Floyd warshall uses dynamic programming approach in which problem is divided into the small subproblem find the solution of each of them and combine at the end. time complexity is  $O(n^3)$ .

(-0.25 points): incorrect dimensionality

### Exercise 3: Solve and provide answer

✖ Input: matrix of the graph determined by reachability

Transitive closure property: 1 if a directed edge between nodes x and y exists and is reachable, 0 if otherwise



	A	B	C	D	E	F
A	1	1	0	0	0	0
B	0	1	1	1	0	0
C	0	1	1	1	0	0
D	1	0	0	1	1	0
E	0	0	1	0	1	1
F	0	0	0	0	1	1

The resulting matrix is given below:

$T^{(0)}$ :

1	1	0	0	0	0
0	1	1	1	0	0
0	1	1	1	0	0
1	0	0	1	1	0
0	0	1	0	1	1
0	0	0	0	1	1

CxR1= (A,A) (A,B) (D,A) (D,B)

$T^{(1)}$ :

1	1	0	0	0	0
0	1	1	1	0	0
0	1	1	1	0	0
1	0	0	1	1	0
0	0	0	0	1	1
0	0	0	0	0	0

$T^{(2)}$ :



1	1	1	0	0	0
0	0	1	1	0	0
0	1	0	1	0	0
1	0	0	0	1	0
0	0	1	0	0	1
0	0	0	0	1	0

$T^{(3)}$ :

1	1	1	1	1	0
0	0	1	1	0	0
0	1	0	1	0	0
1	0	0	0	1	0
0	0	1	0	1	1
0	0	0	0	1	0

$T^{(4)}$ :

1	1	1	1	1	1
0	0	1	1	0	0

0	1	0	1	0	0
1	0	0	0	1	0
0	0	1	0	0	1
0	0	0	0	1	0

$T^{(4)}$ :

1	1	1	1	1	1
0	0	1	1	0	0
0	1	0	1	0	0
1	0	0	0	1	0
0	0	1	0	0	1
0	0	0	0	1	0

$T^{(5)}$ :

1	1	1	1	1	1
1	0	1	1	0	0
0	1	0	1	0	0
1	0	0	0	1	0

0	0	1	0	0	1
0	0	0	0	1	0

$T^{(6)}$ :

1	1	1	1	1	1
1	1	1	1	0	0
0	1	0	1	0	0
1	0	0	0	1	0
0	0	1	0	0	1
0	0	0	0	1	0

$T^{(7)}$ :

1	1	1	1	1	1
1	1	1	1	1	0
0	1	0	1	0	0
1	0	0	0	1	0
0	0	0	0	1	1
0	0	0	0	0	0

$T^{(8)}$ :

1	1	0	0	0	0
0	1	1	1	0	0
0	1	1	1	0	0
1	0	0	1	1	0
0	0	0	0	1	1
0	0	0	0	0	0

(-0.25 points): incorrect transitive closure