Tutorial

Instructor: Meng-Fen Chiang

COMPCSI220: WEEK 10

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

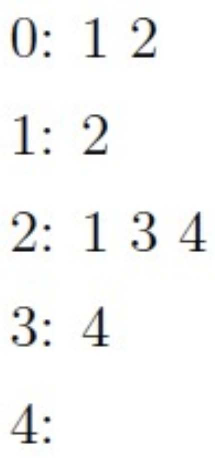
Course Website: https://ankechiang.github.io/cs220_swu.html#week10

OUTLINE

- Question 1: Graph Traversal
- Question 2: Graph Traversal
- Question 3: Graph Traversal
- Question 4: Graph Traversal
- Question 5: Graph Traversal
- Question 6: Graph Traversal
- Question 7: Topological Sorting
- Question 8: Girth Computation

Question 1.1-1.3

1. Draw the digraph G.
2. Draw the sub-digraph induced by {1, 2, 3}.
3. Draw the underlying graph of G.



0: 1 2

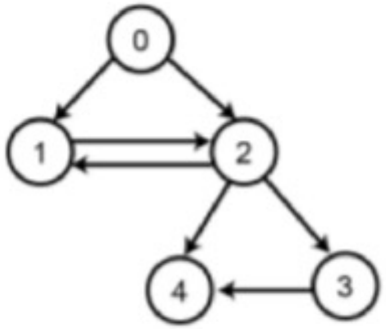
1: 2

2: 1 3 4

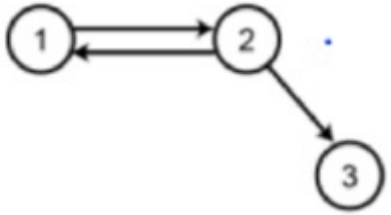
3: 4

4:

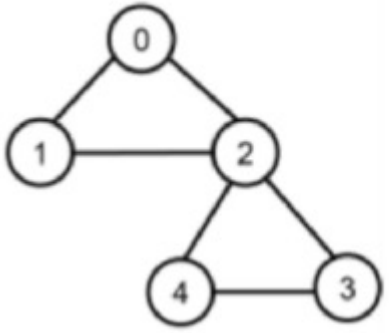
digraph G



sub-digraph induced by {1,2,3}



underlying graph



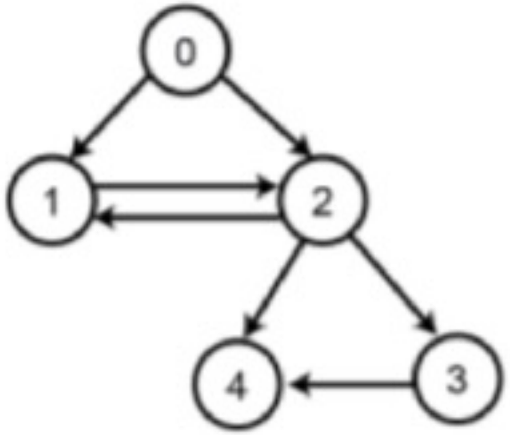
Question 1.4

- 4. If the search forest F contains exactly the following arcs (0, 1), (1, 2), (3, 4), what is the order of nodes visited?

Step1: The order is, we call the visit function on node 3, and visit 3 and 4.

Step2: Then, another run of visit function is called on node 0, where we visit 0, then 1 and 2.

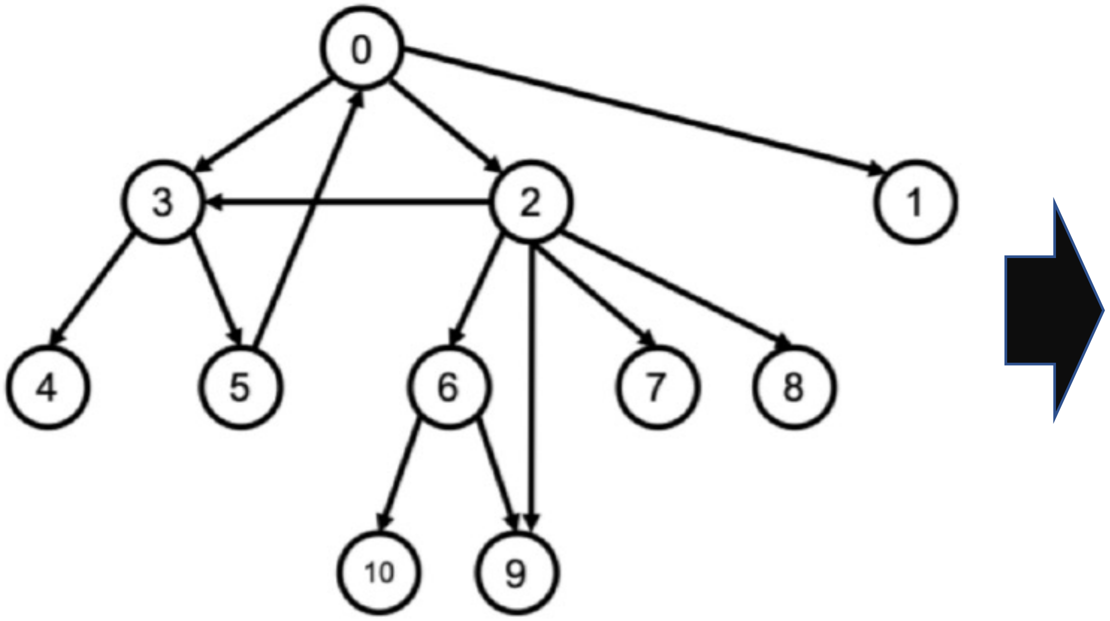
Step3: Therefore, the order we visit the nodes is 3, 4, 0, 1, 2.

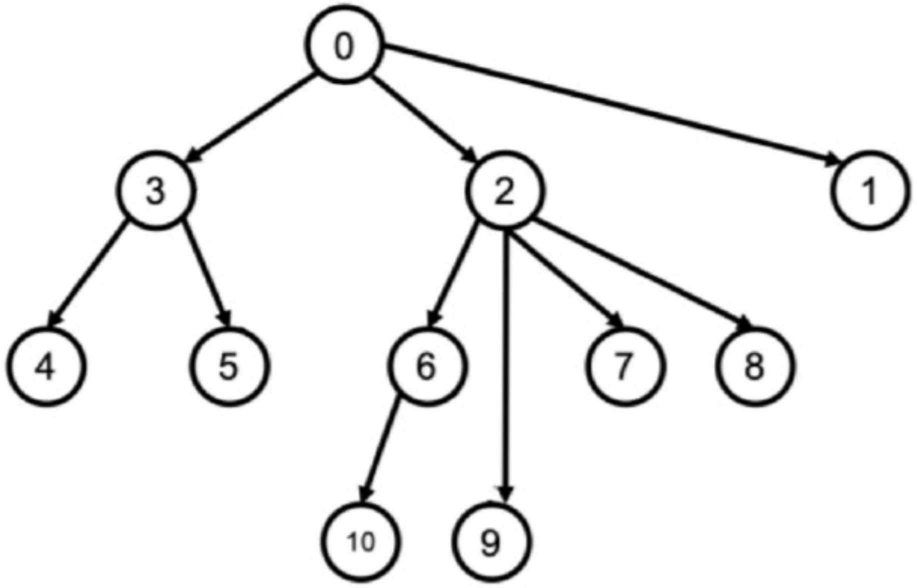


Question 2

- Perform the general traversal algorithm define in the coursebook/lecture slides. Using the convention that nodes are chosen in ascending numeric order when there is a choice of nodes. Show the resulting ***pred* array** and the **search forest**.

pred = [null, 0, 0, 0, 3, 3, 2, 2, 2, 2, 6]





Question 3

- Is priority first search a generalisation of depth-first and breadth-first search?

- Yes. Both BFS and DFS can be instantiated from a priority-first search by choosing how the priority queue prioritizes nodes.

- When we put highest priority to the newly inserted node, the priority queue will follow the last in first out policy, and thus PFS becomes DFS.

- When we put highest priority to nodes with smallest depth, the priority queue will follow the first in first out policy, and thus PFS becomes BFS.

Question 4 (BFS)

- Carry out BFS and DFS on the digraph with adjacency list given below. Show the state of the queue/stack after each change in its state.

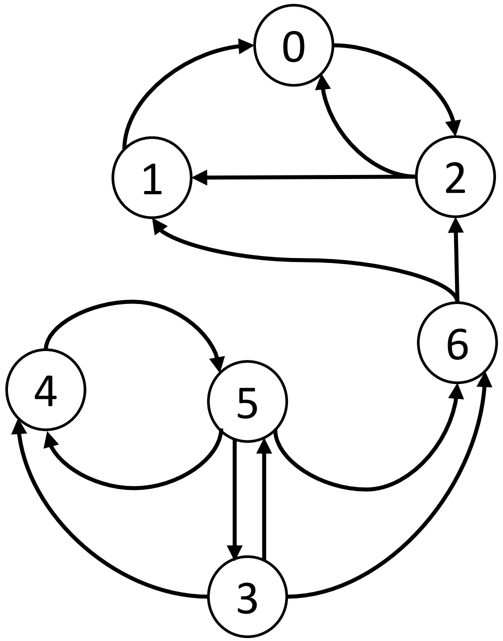
1. The states of the queue for the first search tree:

   [],[0],[0, 2],[2],[2, 1],[1],[].

2. The state of the queue for the second search tree:

   [], [3], [3, 4], [3, 4, 5], [3, 4, 5, 6], [4, 5, 6], [5, 6], [6], []

0 : 2
1 : 0
2 : 0 1
3 : 4 5 6
4 : 5
5 : 3 4 6
6 : 1 2



Question 4 (DFS)

- Carry out BFS and DFS on the digraph with adjacency list given below. Show the state of the queue/stack after each change in its state.

1. The states of the stack for the first search tree:

   [], [0], [0, 2], [0, 2, 1], [0, 2], [0], [].

2. The states of the stack for the second search tree:

   [], [3], [3, 4], [3, 4, 5], [3, 4, 5, 6], [3, 4, 5], [3, 4], [3], []

$0 : 2$

$1 : 0$

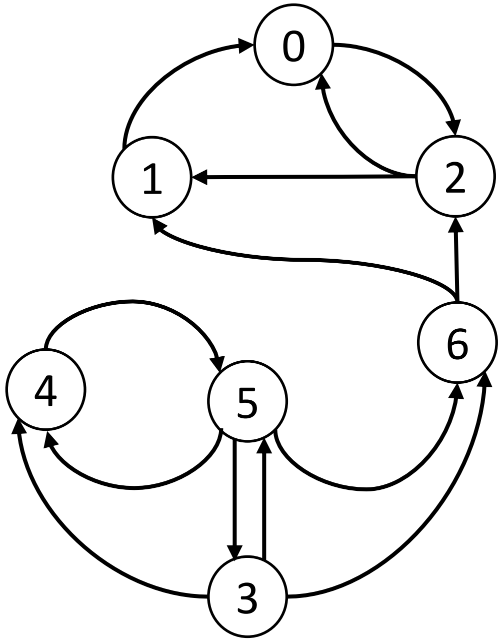
$2 : 0\ 1$

$3 : 4\ 5\ 6$

$4 : 5$

$5 : 3\ 4\ 6$

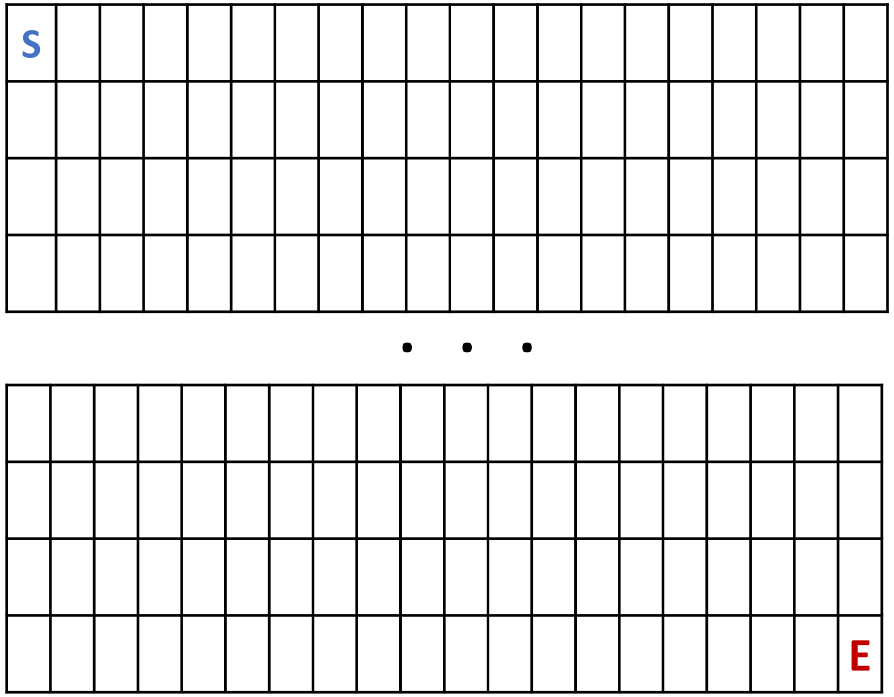
$6 : 1\ 2$



Question 5

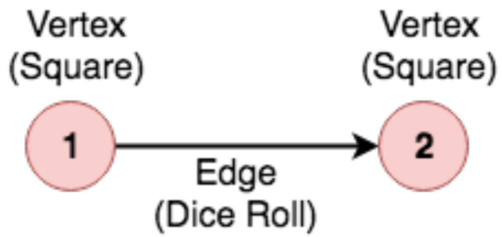
- Consider the grid graph defined as follows. Which ordering will cause the DFS to visit the fewest nodes before terminating?

- A: right, down, up, left
- B. up, down, left, right
- C. left, right, down, up
- D. right, down, left, up
- E. right, left, down, up



Question 6 (Extended Version)

- Run BFS to calculate the minimum number of throws
  - move forward by sum of dices
  - climb up to the top of the ladder from the foot
  - stop at a snake's head and drop down to lower level
  - starts at square 1, reaches the last square n to win



Vertex (Square)

Vertex (Square)

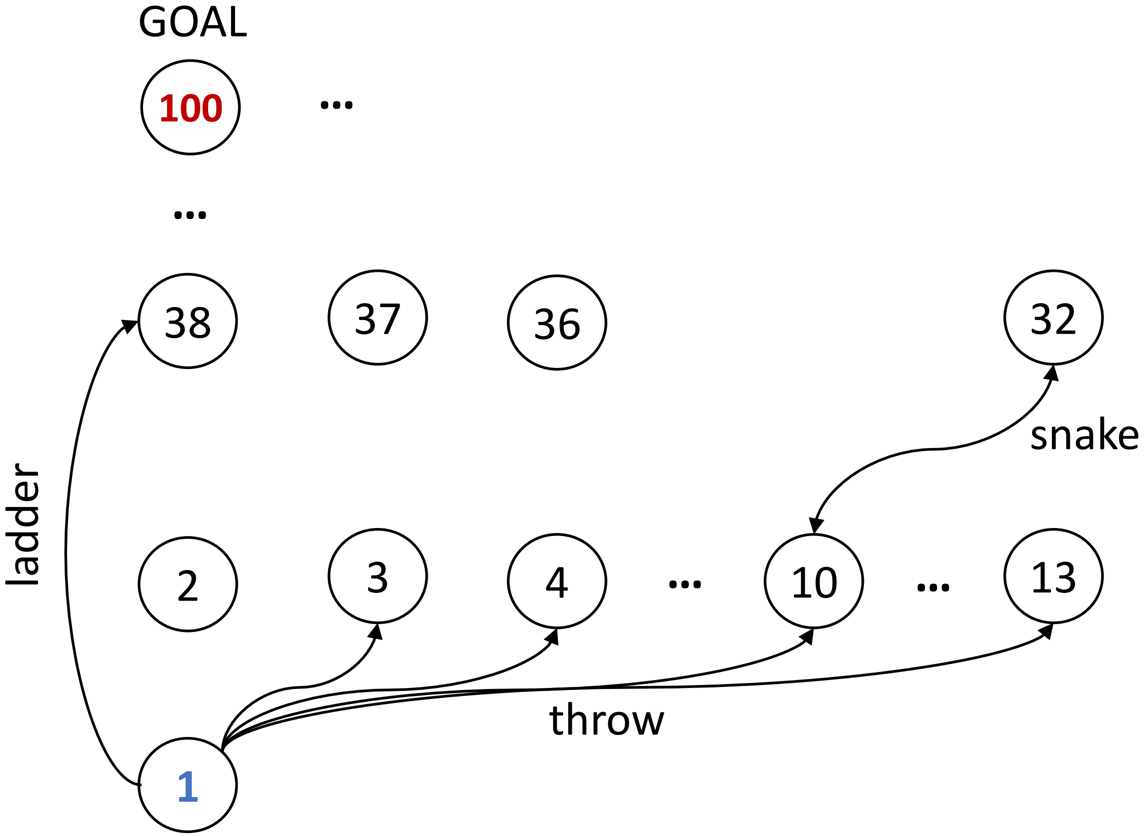
1

2

Edge (Dice Roll)

1. Create (u, v) by 2 to12 moves (the minimum and maximum steps we can move for a single throw), or
2. Create (u,v) at the foot of ladder or at the head of snake connecting the two

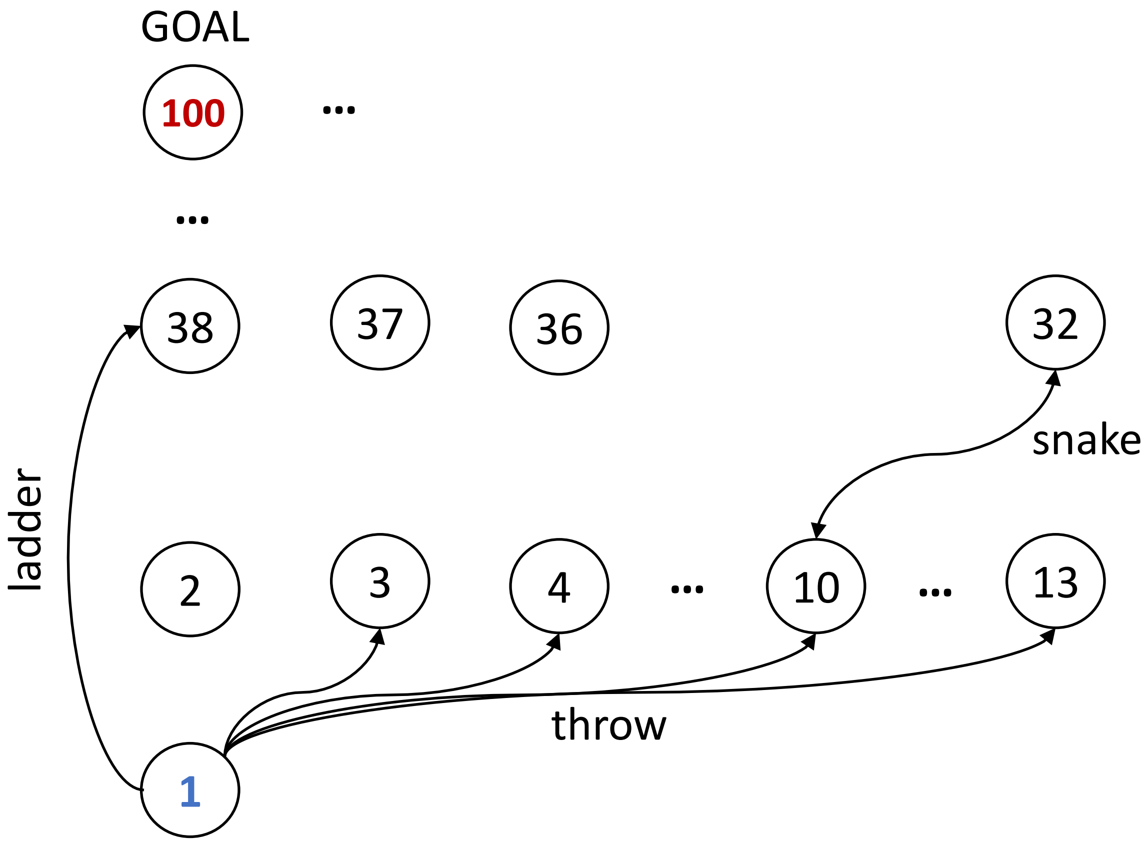
Question 6 (Extended Version)



Question 6 (Extended Version)

- Run BFS on this graph, and then it would take all the paths in the search tree from the initial node to the final node.

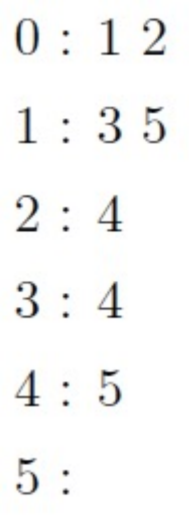
- Return one with the smallest number of edges of the first type



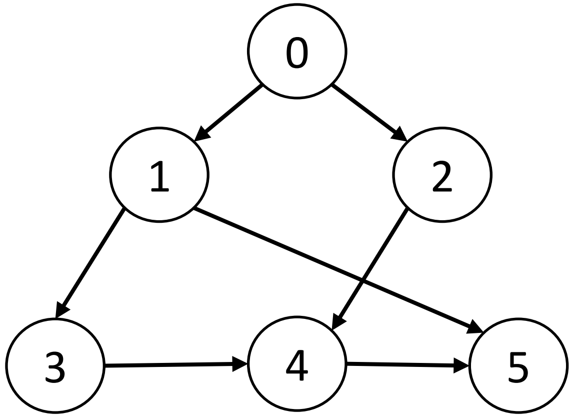
Question 7

- Do the graphs below have a topological sorting if so give one, if not why not?

- Topological orders can be: (0, 1, 3, 2, 4, 5), (0, 1, 2, 3, 4, 5) or (0, 2, 1, 3, 4, 5).



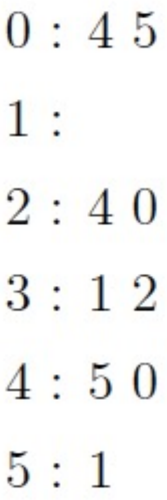
0 : 1 2
1 : 3 5
2 : 4
3 : 4
4 : 5
5 :



Question 7 (Contd.)

- Do the graphs below have a topological sorting if so give one, if not why not?

- There is no topological order as there is a cycle created by the two arcs (0, 4) and (4, 0).



0 : 4 5

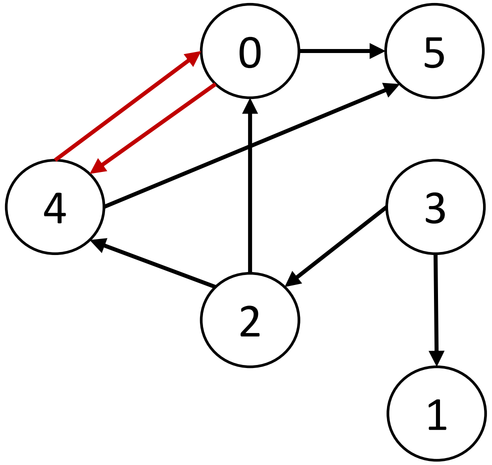
1 :

2 : 4 0

3 : 1 2

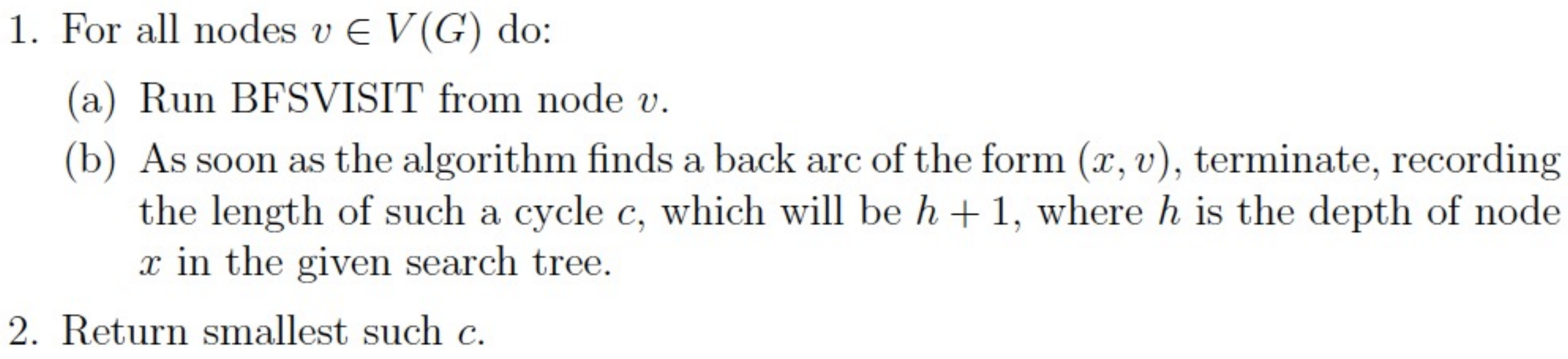
4 : 5 0

5 : 1



Question 8.1

- Why is there no need to continue to the end of the level before halting the traversal?

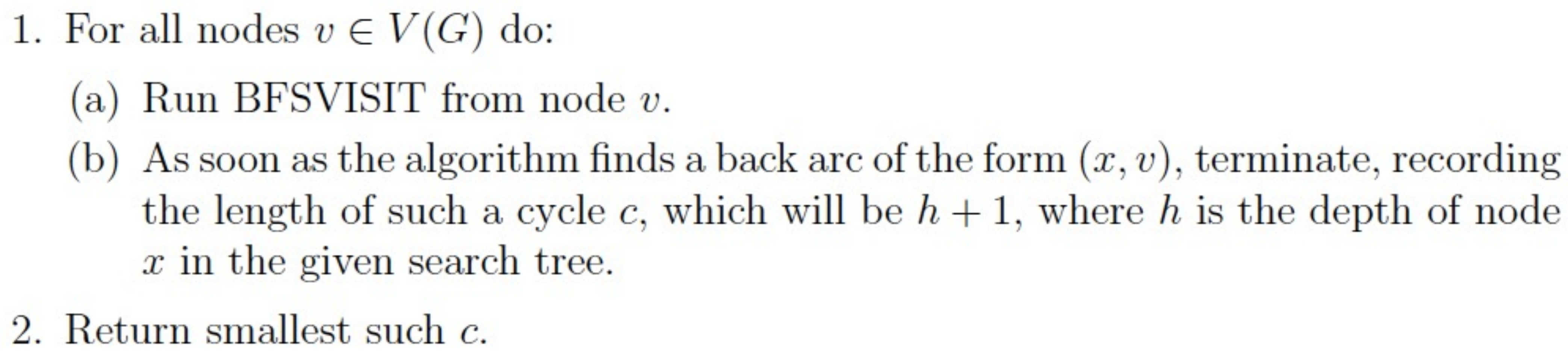


```
1. For all nodes $v \in V(G)$ do:
    (a) Run BFSVISIT from node $v$.
    (b) As soon as the algorithm finds a back arc of the form $(x, v)$, terminate, recording
        the length of such a cycle $c$, which will be $h + 1$, where $h$ is the depth of node
        $x$ in the given search tree.
2. Return smallest such $c$.
```

- **Girth computation:** we need to check all nodes at the same level because there could be shorter cycle found by the algorithm at the same level.
- **Above**: all cycles found at the same level has the same length, h + 1

Question 8.2

- What is the running time for doing so?



```
1. For all nodes $v \in V(G)$ do:
    (a) Run BFSVISIT from node $v$.
    (b) As soon as the algorithm finds a back arc of the form $(x, v)$, terminate, recording
        the length of such a cycle $c$, which will be $h + 1$, where $h$ is the depth of node
        $x$ in the given search tree.
2. Return smallest such $c$.
```

- **Loop:** we may need to traverse the whole graph in the worst case, that is $\boldsymbol{O(n+m)}$ for adjacency list, and $\boldsymbol{O(n^2)}$ for adjacency matrix
- **Overall:** worst-case running time complexity is $\boldsymbol{O(n(n+m))}$ for adjacency list and $\boldsymbol{O(n^3)}$ for adjacency matrix.

Resources

- Course Website
  - https://ankechiang.github.io/cs220_swu.html#week10
- Lecture Notes
  - https://github.com/ankechiang/ankechiang.github.io/tree/master/cs220_lecture
- Lecture Recordings
  - https://github.com/ankechiang/ankechiang.github.io/tree/master/cs220_recording