# Graph algorithms for competitive programming

5 Dec 2022

Name: ______________________________

Reg. no: ______________________________

. . . . . . . . . . . . . . . . . . . . For instructors' use . . . . . . . . . . . . . . . . . . . .

| Question: | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Points: | 2 | 2 | 2 | 2 | 2 | 3 | 3 |
| Score: | | | | | | | |

| Question: | 8 | 9 | 10 | 11 | 12 | 13 | Total |
|---|---|---|---|---|---|---|---|
| Points: | 3 | 3 | 3 | 5 | 6 | 14 | 50 |
| Score: | | | | | | | |

## Multiple-choice questions

1. *(2 points)* Which of the two traversals (BFS and DFS) may be used to find if there is a path between two vertices in a given graph?

   (a) Only BFS

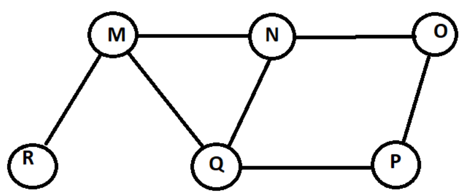
   (b) Only DFS

   (c) Both BFS and DFS

   (d) Neither BFS nor DFS

   **Answer:**
   (c)

2. *(2 points)* The Breadth-First Search algorithm is run on the following graph. Which of the options are valid traversal orders?




   (a) NQMPOR

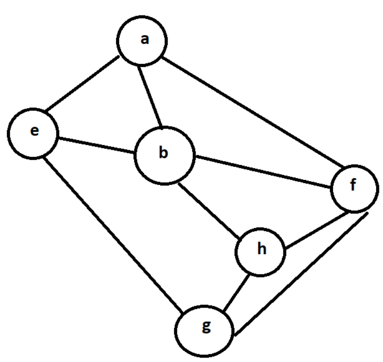
   (b) QMNPOR

   (c) QMNPRO

   (d) MNOPQR

   **Answer:**
   (b) and (c)

3. *(2 points)* The Depth-First Search algorithm is run on the following graph. Which of the options are valid traversal orders?



(a) a b e g h f

(b) a b f e h g

(c) a b f h g e

(d) a f g h b e

**Answer:**

(b) and (c)

4. *(2 points)* Consider a simple undirected graph of 6 vertices. If the graph is disconnected, then the maximum number of edges it can have is

   (a) 10

   (b) 6

   (c) 8

   (d) 15

**Answer:**

(a)

5. *(2 points)* A graph has $n > 4$ vertices and is made up of three connected components. What is the least number of edges it can have?

   (a) $n$
   (b) $n - 1$
   (c) $n - 2$
   (d) $n - 3$
   (e) None of the above.

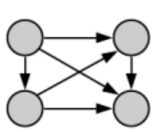
**Answer:**

(a)

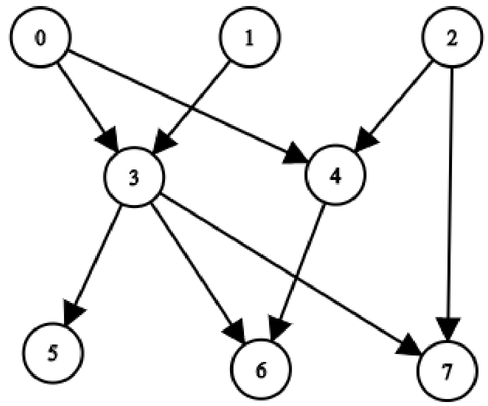
## Short answers

6. *(3 points)* Draw an acyclic, directed graph having four vertices and six edges.

**Answer:**



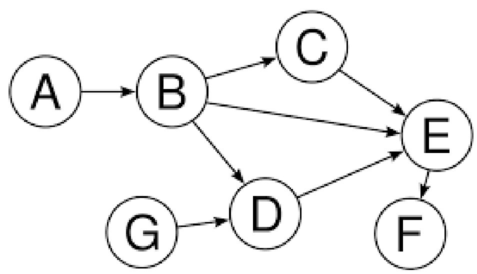
7. *(3 points)* Copy the below-given graph to your answer sheet. Write the in-degree of each node next to each node.




**Answer:**

0:0, 1:0, 2:0, 3:2, 4:2, 5:1, 6:2 & 7:2

8. *(3 points)* Give a topological ordering of the graph shown below. If there are multiple orderings write down the lexicographically smallest one.




**Answer:**

A,B,C,G,D,E,F

9. *(3 points)* Suppose $G$ is a simple directed graph in which there is a directed path from any node to any other node. If the number of nodes in $G$ is $n$, what is the least number of edges $G$ can have?

**Answer:**

Ans: $n$.

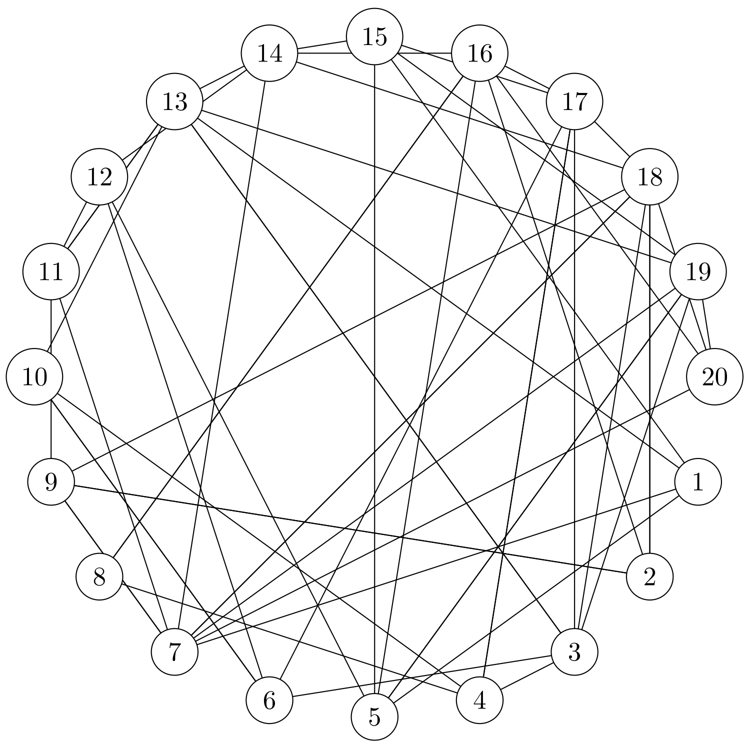
10. *(3 points)* Which of these statements is/are true?

(a) Topological sort can be used to detect a directed cycle in a directed graph.
(b) Topological sort can be used to find the longest path in a directed acylic graph.
(c) Suppose we take a directed acyclic graph and replace each directed edge by an undirected edge. The resulting graph will be a tree.

> **Answer:**
> (a) and (b)

## Problems

11. *(5 points)* Consider the graph given below that has 20 nodes. The length of the shortest path between a pair of vertices $u, v$ is called the *distance* between $u$ and $v$.



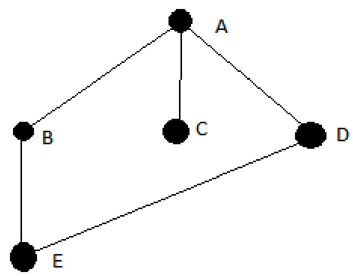

(a) Write down the list of vertices that are at distance two from node 1.

(b) Write down the list of vertices that are at distance three from node 1.

In both cases, write the list of vertices in ascending order.

> **Answer:**
> (a) 3,8,10,11,12,14,16,18,19 and 20.
> (b) 2,4,6,8 and 17.

12. *(6 points)* Consider the following undirected graph $G$ and its adjacency list representation as stored in the memory.




```
Adjacency list representation
=============================
A -> [D,C,B]
B -> [A,E]
C -> [A]
D -> [A,E]
E -> [B,D]
```

Given below is the pseudo-code which does a traversal on a graph. The variable `st` is a stack. Stack is a last-in-first-out data structure. The variable `adj[v]` refers to the adjacency list given above. We run this code on the graph instance $G$.

List the vertices in the order in which they get visited.

---

```
Initialize seen(v) to false for all v
st = stack()

v0 = 'A'
push(st, v0)
seen(v0) := true
while (!empty(st))
  v := pop(st)
  visit(v)
  for each y in adj[v]:
    if not seen(y)
      push(st, y)
      seen(y) := true
```

---

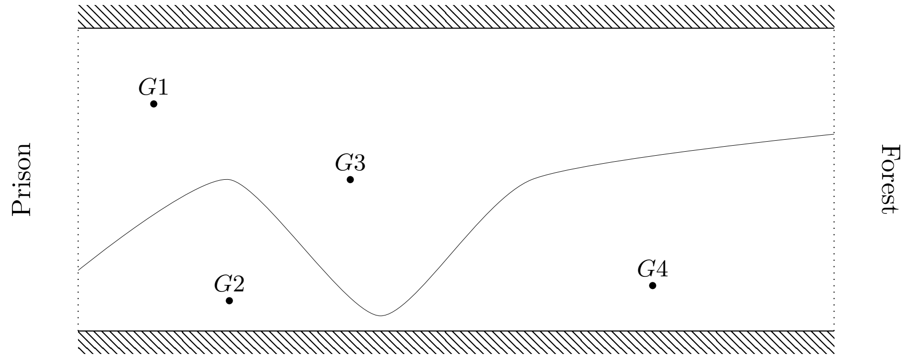
**Answer:**

A, B, E, C and D.

13. *(14 points)* A prisoner is planning to escape from his prison. There is a forest near the prison and the prisoner will be free if he reaches the forest. In the picture below, the prison is on the left edge and the forest is on the right edge.

    There is an open rectangular field that separates the prison and the forest. The length of the field is 5000 m (x-axis in the picture). The width of the field is 1000 m. (y-axis, dotted vertical line in the picture). The length of the field is lined by walls on either side.

    There are $n$ guards at different locations on the field. Each guard's range of view is limited to 10m. In order to escape, the prisoner has to pass through the field and reach the forest. He can escape if he maintains a distance of more than 10m from the nearest guard at all times.




(a) (4 points) Suppose you're free to pick the positions of the guards. What is the minimum number of guards needed to ensure that the prisoner cannot escape?

(b) (10 points) You are given the number $n$ and the locations of the $n$ guards as input. Each location is specified by an $(x, y)$ coordinate. All the locations are contained inside the field so $x \leq 5000$ and $y \leq 1000$. Design an algorithm to determine if the prisoner can pass through the field and escape. Your algorithm must run in $O(n^2)$ time.

The input corresponding to the picture has $n = 4$ since there are four guards. The prisoner can pass through the field maintaining a distance of more than 10m from every guard. This escape route is shown by the curved line in the picture. The prisoner can start from anywhere on the left edge and each anywhere on the right edge.

```
Sample Input

4
500 800
1000 200
2200 500
3600 220

Output

Yes
```

**Answer:**

(a) 50 guards. Put them in a vertical line 20 m apart. (b) Model this as a graph. Each guard is a vertex. Two guards are adjacent if they are at most distance of 20 m apart (there is no way to go between them).

Add a vertex $s$ for the top boundary of the canyon and a vertex $t$ for the bottom boundary. A guard is adjacent to $s$ if it is within 10m of the top border. A guard is adjacent to $t$ if it is within 10m of the bottom border. *Claim*: If there is a sequence of guards that connect the top border to the bottom border in this graph, there is no way to escape. Otherwise, we can find a route that goes between the guards. Hence, the solution just requires checking whether $t$ is reachable from $s$ in this graph.