Graph algorithms for competitive programming

6 Dec 2023

Name: ______________________________

Reg. no: ______________________________

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

Question:	1	2	3	4	5	6
Points:	2	2	2	2	2	5
Score:						

Question:	7	8	9	10	11	Total
Points:	6	10	3	6	10	50
Score:						

Multiple-choice questions

Note: More than one option may be correct.

1. *(2 points)* Ten people go to a party. Every person shakes hands with every other person. How many handshakes were there at the party?

   (a) 45

   (b) 50

   (c) 90

   (d) 100

   Answer:
   (b)

2. *(2 points)* A directed acyclic graph has 9 edges. What is the least number of vertices the graph can have?

   (a) 3

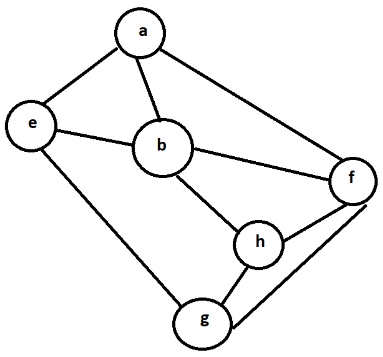
   (b) 4

   (c) 5

   (d) 6

   Answer:
   (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)* We take a directed acyclic graph of n vertices and replace each directed edge with an undirected edge. Which of the statements is true?

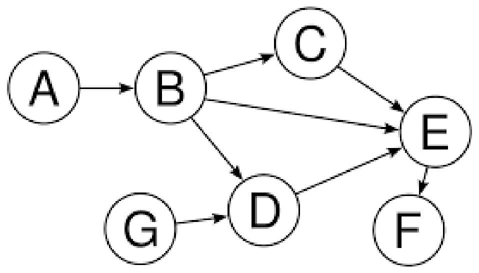
   (a) The resulting graph will always be a tree.

   (b) The resulting graph will be a tree if the number of edges is less than n.

   (c) The resulting graph may be a complete graph i.e. have $n(n-1)/2$ edges.

Answer:
(c)

5. *(2 points)* Identify all valid topological orderings of the graph shown below.



   (a) A,B,C,G,D,E,F

   (b) A,G,B,C,D,E,F

   (c) A,G,B,D,C,E,F

   (d) A,B,C,E,F,D,G

Answer:
(a),(b) and (c)

Short-answer questions

6. *(5 points)* Each of the kinds of problems below can be solved using BFS, DFS, or Topological sorting. Write which one is the most appropriate against each type of problem.

   (a) Find the shortest path from a source to other vertices in an unweighted graph.

   (b) Check if a vertex in a tree is an ancestor of some other vertex.

   (c) We have n tasks to do. The tasks are not independent, and executing one task is only possible if other tasks have already been executed. Find a way to order the tasks given the dependencies.

   (d) Finding a solution to a game with the least number of moves, if a vertex of the graph can represent each state of the game, and the moves from one state to the other are the edges of the graph.

(e) A bipartite graph is a graph whose vertices can be divided into two disjoint sets so that every edge connects two vertices from different sets (i.e., there are no edges that connect vertices from the same set). These sets are usually called sides. You are given an undirected graph. Check whether it is bipartite.

Answer:

(a) BFS

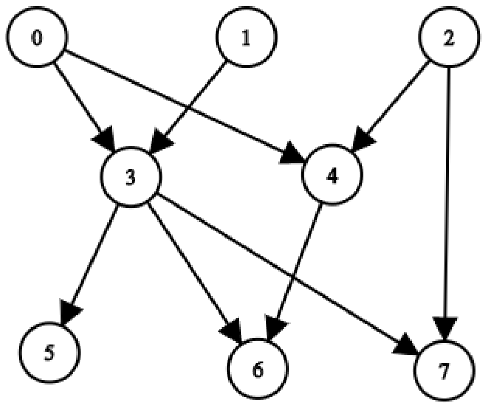
(b) DFS

(c) Toposort

(d) BFS

(e) DFS or BFS. (either of them is correct).

7. *(6 points)* Consider the directed acyclic graph shown below. The graph is stored in an adjacency list `adj`. The vertices in the adjacency list of any vertex are in increasing order. Initially, the vector `visited` values are set to false.



```
void dfs(int u, VVI &adj, vector<bool> &visited){
    visited[u] = true;
    for(int v: adj[u]){
        if( !visited[v] )
            dfs(v,adj,visited);
    }
    cout << u << " ";
}
```

Assume that lines (a), (b), and (c) below are called in a sequence. Write the output of each function call.

(a) `dfs(1,adj,visited)`

(b) `dfs(2,adj,visited)`

(c) `dfs(0,adj,visited)`

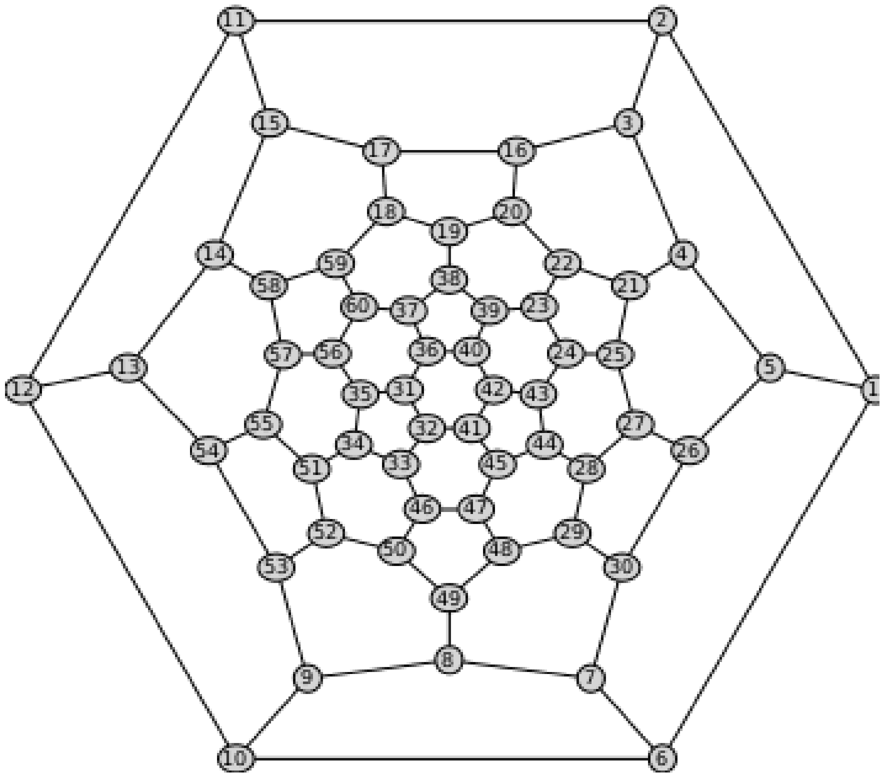
Answer:

(a) 5 6 7 3 1

(b) 4 2

(c) 0

8. Consider the graph shown below. The right-most node is labeled 1.



(a) *(4 points)* List the vertices, in increasing order, at the shortest distance 2 from the node labeled 1.

(b) *(6 points)* List the vertices, in increasing order, at the shortest distance 4 from node labeled 1.

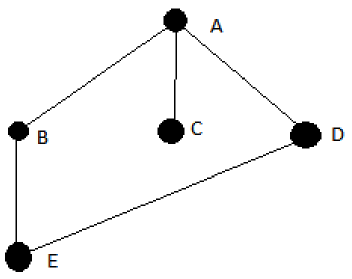
Image credit: Maplesoft.

Answer:

(a) 3,4,7,10,11, 26.

(b) 13, 14, 17, 20, 22, 25, 28, 29, 49, 53.

9. *(3 points)* Consider the following undirected graph G and its adjacency list representation as stored in the memory.



```
Adjacency list representation
=============================
A -> [B,C,D]
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 above mentioned adjacency list. 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, D, E, C and B.

10. *(6 points)* We want to print all the bitstrings of length `n` that have exactly `k` one bits. Fill in the blanks on line 9 and 10 to complete the program. An example of how the program might be run is given inside the `main` function.

```
 1 void dfs(int n,int k,string word){
 2     if( n < k ) return;
 3     if( k < 0 ) return;
 4 
 5     if(n==0 and k==0){
 6         cout << word << endl;
 7         return;
 8     }
 9     dfs(n-1,________);
10     dfs(n-1,________);
11 }
12 
13 int main(){
14     int n=4,k=2;
15     dfs(n,k,"");
16     return 0;
17 }
```

Answer:

```
 9     dfs(n-1,k,word+"0");
10     dfs(n-1,k-1,word+"1");
```

11. *(10 points)* A 6×6 grid (a prison) has five guards and a prisoner. The objective of the prisoner is to escape the prison in the fewest steps possible. The exit is at the bottom-right corner of the grid. The guards patrol the prison by moving one step at each time step. At time $t = 0$, the guards G_1 to G_5 and the prisoner are initially at positions given by the input. Guards G_1, G_2 and G_3 move one step to the right at each time step. Guards G_4 and G_5 move one step down at each time step. When the guards each the boundary of the grid, they (magically) wrap around. For example, when G_5 reaches the bottom of the grid his next step will take him to the top row. The prisoner can move one step in any of the four directions (wrapping around if needed). The prisoner is caught if the prisoner and a guard are in the same cell at some time step. You need to help the prisoner escape in the fewest time steps, without getting caught.

    We wish to solve the problem using BFS. You can access a function called `bfs(G,source,target)` that takes a graph as input and returns the shortest path from `source` to `target`. Answer the questions below on how you would build a graph G that can be used to solve the problem.

    (a) Explain the state space and what would be the nodes in the graph G.

    (b) What is the source node and the target node in your graph G?

    (c) If the grid size was $n \times n$ instead of 6×6 and there were still five guards, how many vertices (asymptotically) would the graph G have? Assume that the graph is built following your response to part (a). Try to make this number as small as possible.

    (d) Explain how to build the edges G. Will the edges be directed or undirected?

    (e) Can you think of a way to reduce the number of nodes in G less than $O(n^4)$?

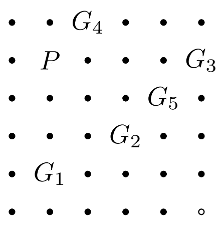


Figure 1: The guards are labelled from G_1 to G_5. The prisoner is labeled P. The bottom right corner is the exit with cell number $(5, 5)$. The top-left corner has the cell number $(0, 0)$.

Answer:

(a) Each node will be represented by a tuple $(g1_x, g1_y, g2_x, g2_y, g3_x, g3_y, g4_x, g4_y, g5_x, g5_y, p_x, p_y)$. The guards' positions are denoted by (gi_x, gi_y) and the prisoner's position by (p_x, p_y). Additionally, create a special node called `target` and connect all the nodes with $(p_x, p_y) = (5, 5)$ to that node.

(b) The source node encodes the initial positions of the guards and the prisoners given in the input. The target node is specified above.

(c) One of the coordinates of each guard is fixed. Hence, either gi_x or gi_y is a constant. We take up $O(n^7)$ nodes in the above representation. Five changing positions for the guards and two for the prisoner.

(d) There would be a directed edge from node specified in (a) to $(g1_x + 1, g1_y, g2_x + 1, g2_y, g3_x + 1, g3_y, g4_x, g4_y + 1, g5_x, g5_y + 1, p_x \pm 1, p_y \pm 1)$. Addition and subtraction is taken modulo 6, since the boards wraps around. We must delete nodes that have $(p_x, p_y) = (gi_x, gi_y)$ for some i. This condition ensures that the prisoner does not get caught.

(e) Given the initial locations and the location of one guard, we can find out the location of all other guards at any timestep. Hence, it suffices to keep track of the position of only the first guard in the state space. Keeping track of G_1's position takes $O(n)$ space. Keeping track of the prisoner's position takes $O(n^2)$ space. Hence, we need only $O(n^3)$ nodes in G.