Name: Pourna Sengupta

ID: 10906577

CSCI 3104, Algorithms **Escobedo & Jahagirdar**

Homework 5A (40 points) **Summer 2020, CU-Boulder**

Advice 1: For every problem in this class, you must justify your answer: show how you arrived at it and why it is correct. If there are assumptions you need to make along the way, state those clearly.

Advice 2: Verbal reasoning is typically insufficient for full credit. Instead, write a logical argument, in the style of a mathematical proof.

Instructions for submitting your solution:

- The solutions **should be typed**, we cannot accept hand-written solutions. Here's a short intro to **Latex**.
- In this homework we denote the asymptomatic *Big-O* notation by $\mathcal{O}$ and *Small-O* notation is represented as o.
- We recommend using online Latex editor **Overleaf**. Download the **.tex** file from Canvas and upload it on overleaf to edit.
- You should submit your work through **Gradescope** only.
- If you don't have an account on it, sign up for one using your CU email. You should have gotten an email to sign up. If your name based CU email doesn't work, try the identikey@colorado.edu version.
- Gradescope will only accept **.pdf** files (except for code files that should be submitted separately on Canvas if a problem set has them) and **try to fit your work in the box provided**.
- You cannot submit a pdf which has less pages than what we provided you as Gradescope won't allow it.

Name:	Pourna Sengupta
ID:	10906577

CSCI 3104, Algorithms **Escobedo & Jahagirdar**
Homework 5A (40 points) **Summer 2020, CU-Boulder**

Piazza threads for hints and further discussion

Piazza Threads
Question 1
Question 2
Question 3
Question 4

Recommended reading:
Graph Algorithms Intro: Ch. 22 $\rightarrow$ 22.1, 22.2, 22.3
Graph Algorithms SSSPs: Ch. 24 $\rightarrow$ 24.3

Name: Pourna Sengupta

ID: 10906577

CSCI 3104, Algorithms **Escobedo & Jahagirdar**

Homework 5A (40 points) **Summer 2020, CU-Boulder**

1. (5 pts) Consider the following unweighted directed graph.

HW5/graph_question.png

(a) (2.5 pts) Write down the order in which nodes are visited if Depth First Search (DFS) is called on the above graph with **S** as the source node.

Answer:
1: S
2: A
3: B
4: E
5: F
6: C
7: D

(b) (2.5 pts) Write down the order in which nodes are visited if Breadth First Search (BFS) is called on the above graph with **S** as the source node.

Name: Pourna Sengupta

ID: 10906577

CSCI 3104, Algorithms **Escobedo & Jahagirdar**

Homework 5A (40 points) **Summer 2020, CU-Boulder**

Answer:
1: S
2: A
3: C
4: B
5: F
6: D
7: E

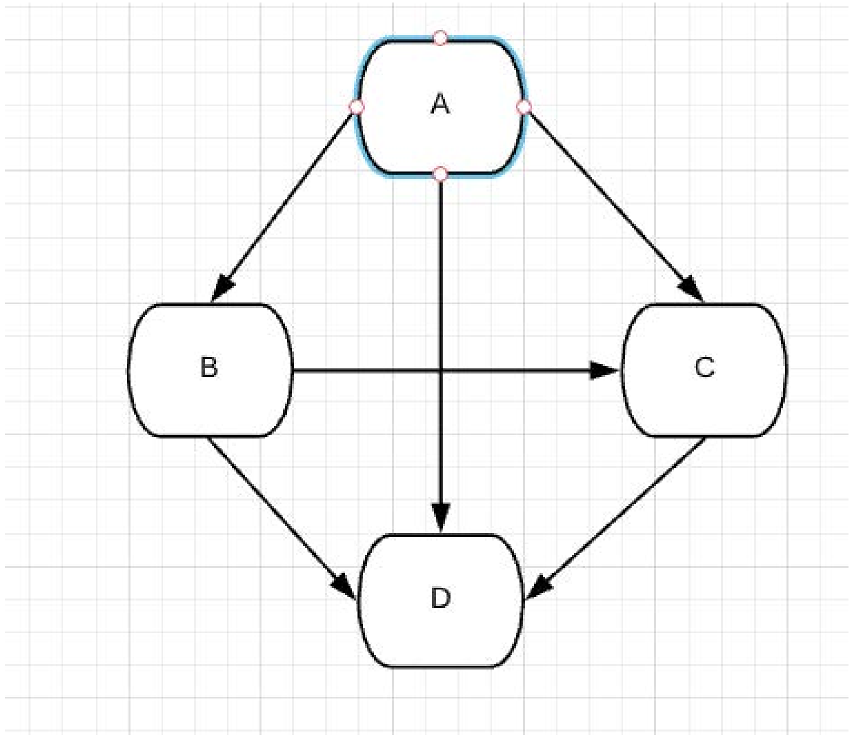
Name: Pourna Sengupta

ID: 10906577

CSCI 3104, Algorithms **Escobedo & Jahagirdar**
Homework 5A (40 points) **Summer 2020, CU-Boulder**

2. (5 pts) A simple path $s \to t$ in a graph G is a path starting at s, ending at t, and never visiting the same vertex twice. Give an example graph (simple, undirected, unweighted) $G = (V; E)$ and vertices $s, t \in V$ such that DFS finds a path from s to t which is neither a shortest path nor a longest simple path. Detail the execution of DFS (list the contents of the stack at each step, and which vertex it pops off the stack), show the final $s \to t$ path it finds, show a shorter $s \to t$ path, and a longer simple $s \to t$ path.

The DFS for the graph below to travel from A to D is as follows: We travel from A to B first. Then we travel to the children of B which are C and D. Therefore, by traveling to D from B, we have reached the end goal with a path length of 2. The shortest path from A to D is 1 and the longest is 3, traveling A to B to C to D.



Name: Pourna Sengupta

ID: 10906577

CSCI 3104, Algorithms **Escobedo & Jahagirdar**
Homework 5A (40 points) **Summer 2020, CU-Boulder**

3. (20 pts) Assume you're given an integer matrix that representing islands in an ocean. A value of 0 in a particular cell indicates water and a value of 1 indicated land. An island is region of land connected vertically, horizontally, or diagonally. The size of an island is the total number of connected land cells. Write an algorithm to compute the sizes of all islands in the matrix. Example:

```
1 0 0 1
0 1 0 1
0 1 0 0
0 1 0 1
```

would output 1, 2, 4.

(a) (6 pts) State if the graph data structure that your algorithm will use for this problem will be i) be weighted or unweighted and ii) directed or undirected. In addition, iii) what makes two nodes have an edge/connection? Give an explanation for each of your answers.

i. The graph is unweighted. The matrix contains 0 and 1, so there is no need for weights for the graph.
ii. The graph is undirected. It is possible to move in all directions without running into anything. By being able to scan the matrix, it is obvious that all movement is allowed so the graph is undirected.
iii. Two nodes have an edge/connection when they are neighboring nodes. A land node is connected to another land node when there is one a unit away in any direction.

Name: Pourna Sengupta

ID: 10906577

CSCI 3104, Algorithms **Escobedo & Jahagirdar**

Homework 5A (40 points) **Summer 2020, CU-Boulder**

(b) (4 pts) Provide a 3-4 sentence description of how your algorithm works, including how the matrix is converted to the graph, how adjacent vertices are identified, and how the algorithm traverses the graph to identify connected vertices.

The Depth First Search recursively looks through the matrix until it finds a land node. It then counts neighboring island nodes that are connected in any direction. It then marks each node as visited and once finished, calculates the area of the island. The DFS function then prints the area and continues to recursively look for more island nodes.

Name: Pourna Sengupta

ID: 10906577

CSCI 3104, Algorithms **Escobedo & Jahagirdar**

Homework 5A (40 points) **Summer 2020, CU-Boulder**

(c) (10 pts) Write well commented pseudo or real code to solve this problem.

```
//helper function to check if the cell is in the matrix
//checks to see if it has been visited
bool isValid (int x, int y){
    return (x >= 0 && x < m && y >= 0 && y < n && visited[x][y] == 0);
}

int islandArea(island, x, y){
    //counter
    int c = 0;
    //if the cell can be visitedd
    if(isValid(x,y)){
        //if the cell is land
        if(island[x][y] == 1){
            //set the location as visited
            visited[x][y] = 1;
            c = 1;
            //add any land connected to the node to find the island size
            c += islandArea(island, x-1, y);
            c += islandArea(island, x+1, y);
            c += islandArea(island, x, y-1);
            c += islandArea(island, x, y+1);
            c += islandArea(island, x-1, y-1);
            c += islandArea(island, x-1, y+1);
            c += islandArea(island, x+1, y-1);
            c += islandArea(island, x+1, y+1);

            return c;
        }
        //if it is not land
        else{
            //return 0
            return 0;

        }
    }
    //the index is not valid
    else{
        return 0;
    }
}

//search the matrix for islands
void DFS(island, x, y){
    //if the cell can be visited
    if(isValid(x,y)){
        //if the cell is land
        if(island[x][y] == 1){
            //print the islands area
            cout << getArea(island, x, y) << endl;
        }
        //set the location as visited
        visited[x][y] == 1;
        //look through neighboring nodes
        DFS(island, x-1, y);
        DFS(island, x+1, y);
        DFS(island, x, y-1);
        DFS(island, x, y+1);
    }
}

int main(){
    int arr[][] = {[1,0,0,1],[0,1,0,1],[0,1,0,0],[0,1,0,1]};
    int visited[][] = {[0,0,0,0], [0,0,0,0], [0,0,0,0], [0,0,0,0]};
    int m = 4;
    int n = 4;
    //Call DFS to start recursive function
    //Starts search at 0
    DFS(a, 0, 0);
}
```

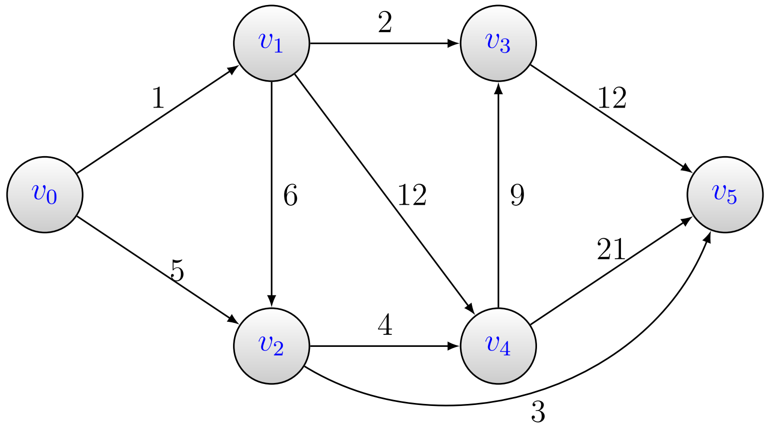
Name: Pourna Sengupta

ID: 10906577

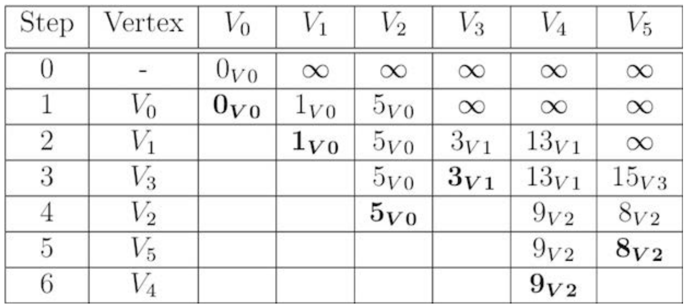
CSCI 3104, Algorithms | **Escobedo & Jahagirdar**

Homework 5A (40 points) | **Summer 2020, CU-Boulder**

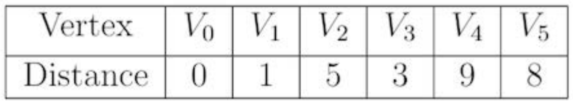
4. (10 pts) Using Dijkstra's algorithm, determine the length of the shortest path from v_0 to each of the other vertices in the graph. Clearly specify the distances from v_0 to each vertex **after each iteration** of the algorithm.



The shortest distance from V_0 to each vertex are shown in the table below. The algorithm finds the shortest distances from V_0 to the other vertices. The second table show us the shortest distances between V_0 and other vertices, such as the shortest path from V_0 to V_4 is shown to be 9.



Step	Vertex	V_0	V_1	V_2	V_3	V_4	V_5
0	-	0_{V0}	∞	∞	∞	∞	∞
1	V_0	$\mathbf{0_{V0}}$	1_{V0}	5_{V0}	∞	∞	∞
2	V_1		$\mathbf{1_{V0}}$	5_{V0}	3_{V1}	13_{V1}	∞
3	V_3			5_{V0}	$\mathbf{3_{V1}}$	13_{V1}	15_{V3}
4	V_2			$\mathbf{5_{V0}}$		9_{V2}	8_{V2}
5	V_5					9_{V2}	$\mathbf{8_{V2}}$
6	V_4					$\mathbf{9_{V2}}$	



Vertex	V_0	V_1	V_2	V_3	V_4	V_5
Distance	0	1	5	3	9	8

Name: Pourna Sengupta

ID: 10906577

CSCI 3104, Algorithms **Escobedo & Jahagirdar**
Homework 5A (40 points) **Summer 2020, CU-Boulder**

5. ***Extra Credit (5% of total homework grade)*** *For this extra credit question, please refer the leetcode link provided below or click here. Multiple solutions exist to this question ranging from brute force to the most optimal one. Points will be provided based on Time and Space Complexities relative to that of the most optimal solution.*

   Please provide your solution with proper comments which carries points as well.

   https://leetcode.com/problems/keys-and-rooms/

```
class Solution {
    public boolean canVisitAllRooms(List<List<Integer>> rooms) {
        int size = rooms.size();
        int[] visited = new int[size];
        visitingHelper(rooms, 0, visited);
        for(int key: visited){
            if(key == 0){
                return false;
            }
        }
        return true;
    }

    private void visitingHelper(List<List<Integer>> rooms, int i, int[] visited){
        if(visited[i] == 1){
            return;
        }
        visited[i] = 1;
        for(int temp : rooms.get(i)){
            visitingHelper(rooms, temp, visited);
        }
    }
};
```