Name: Pourna Sengupta

ID: 109086577

**CSCI 3104, Algorithms** **Escobedo & Jahagirdar**
**Homework 5B (60 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: 109086577

**CSCI 3104, Algorithms** **Escobedo & Jahagirdar**
**Homework 5B (60 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 → 22.1, 22.2, 22.3
Graph Algorithms SSSPs: Ch. 24 → 24.3

Name: Pourna Sengupta

ID: 109086577

**CSCI 3104, Algorithms** **Escobedo & Jahagirdar**
**Homework 5B (60 points)** **Summer 2020, CU-Boulder**

1. (7.5 pts) Give an example of a simple directed, weighted graph $G$ and identify two vertices $s$ and $t$ such that Dijkstra's algortihm started at $s$ does not find the shortest $s \to t$ path.

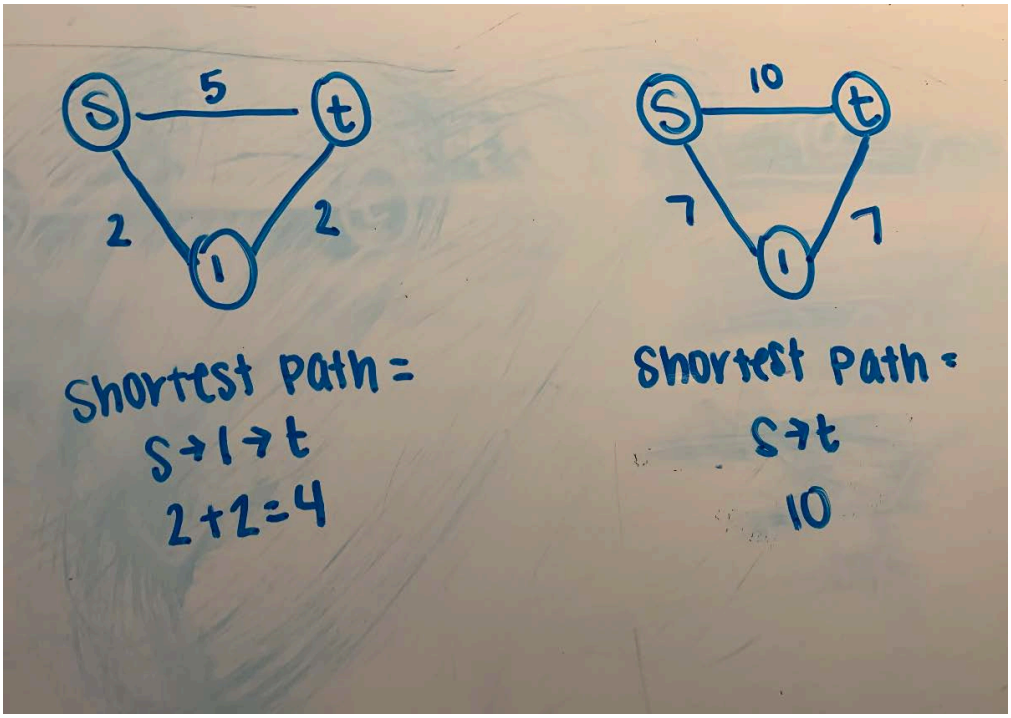
Name: Pourna Sengupta

ID: 109086577

**CSCI 3104, Algorithms** **Escobedo & Jahagirdar**
**Homework 5B (60 points)** **Summer 2020, CU-Boulder**

2. (7.5 pts) Suppose that you have calculated the shortest paths to all vertices from a fixed vertex $s \in V$ of an undirected graph $G = (V, E)$ with positive edge weights. If you increase each edge weight by 5, will the shortest paths from $s$ change? Prove that it cannot change or give a counterexample if it changes.

In the example graph shown below, we see that the shortest path from s to t is "s $\rightarrow$ 1 $\rightarrow$ t" with a distance of 4, which is less than "s $\rightarrow$ t" which has a distance of 5. When the edge weight is increased by 5, we see that the shortest path also changes. It is now "s $\rightarrow$ t" with a distance of 10, which is less than "s $\rightarrow$ 1 $\rightarrow$ t" which has a distance of 14.



Name: Pourna Sengupta
ID: 109086577



3. (20 pts) You are given an undirected tree with and a source vertex $v_0$. You have to the count of the number of vertices in the tree (excluding the source vertex $v_0$) which are at a distance less than or equal to $k$ from the source vertex $v_0$.

   The inputs to your algorithm are an undirected tree in the form of an adjacency list or adjacency matrix and a source vertex $v_0$.
   The output should be the count of the vertices that are at a distance less than or equal to $k$ from the vertex $v_0$.

   For example consider the following tree with a vertex $v_0$ and $k = 2$.

   HW5/tree.png

   The output for the above tree should be the value **8**, as there are 4 neighbouring nodes at unit distance from $v_0$ and 4 nodes at distance of two units from $v_0$.

   (a) (5 pts) Provide a 3-4 sentence description of how your algorithm works, including how you would maintain the distance from the source vertex $v_0$.

Name: Pourna Sengupta

ID: 109086577

**CSCI 3104, Algorithms** **Escobedo & Jahagirdar**
**Homework 5B (60 points)** **Summer 2020, CU-Boulder**

**Answer:** To solve this problem, we use the DFS method to traverse the tree and find the distance between vertices. The DFS algorithm takes the input K and the starting node. From this, we can find the nodes a distance K from the starting node. The algorithm also takes the input of -1, which indicates the parent of the node and the tree. The algorithm recursively moves through the tree by moving through nodes between distance 0 and K and the neighboring nodes of the starting node.

(b) (15 pts) Provide a well commented pseudo-code or actual code to solve the above problem.

```
//recursive DFS function to traverse the tree using neighboring nodes
void DFS(int k, int n, int p, vector<vector<int> >& tree){
    //Base Case: k > 0
    if (k < 0){
        exit;
    }
    int count = 0;
    count++;

    //traverse the tree
    for(int i: tree[n]){
        //if the node is not the parent node
        if(i != p){
            //change node i to the parent of child node
            //recursive call
            DFS(k - 1, i, n, tree);
        }
    }
}
//Helper function that finds nodes between 0 and distance k
int DFSHelper(struct arr graph[], int n, int k, int vert, int edge){
    //set tree into a vector
    vector<vector<int> > tree(v+1, vector<int>());
    //set edges
    for(int i = 0l i < edge; i++){
        int a = graph[i].a;
        int b = graph[i].b;

        tree[a].push_back(b);
        tree[b].push_back(a);
    }
    //call DFS to find all nodes
    return DFS(k, n, -1, tree);
}
```

4. (25 pts) You have two batteries $b_1$, $b_2$, with capacities $c_1$ mAh, $c_2$ mAh respectively. Both the batteries initially have no charge. You have a charging station with infinite supply of electricity and also an earthing device that can be used to remove charge from the battery. You have a battery transfer device which has a source battery position and a target battery position. When you place two batteries in the transfer device, it instantaneously transfers as many mAh from the source battery to the target battery as possible. Thus, this device stops the transfer either when the source battery has no mAh remaining or when the destination battery is fully charged (whichever comes first). The above device can also to fully charge/discharge a battery using the charging station/earthing device.

   The goal in this problem is to determine whether there exists a sequence of transfers amogst the batteries such that in the end, a charge amount of $k$ remains in one of the batteries, where $k \leq c_1$ and $k \leq c_2$.

   For example, consider the case where $c_1 = 4$, $c_2 = 3$ and $k = 2$.
   Let's represent the charge in both the batteries in the form of a tuple $(x, y)$, where $x$ is the charge of $b_1$ and $y$ is the charge of $b_2$. The following sequence of transfers are possible.

   - (0, 0) initially both the batteries have no charge.
   - (4, 0) charged $b_1$ using the charging station.
   - (1, 3) transferred the charge from $b_1$ to $b_2$.
   - (1, 0) discharged $b_2$ using earthing device.
   - (0, 1) transferred the charge from $b_1$ to $b_2$.
   - (4, 1) charged $b_1$ using the charging station.
   - (2, 3) transferred the charge from $b_1$ to $b_2$.

Name: Pourna Sengupta

ID: 109086577

**CSCI 3104, Algorithms** **Escobedo & Jahagirdar**
**Homework 5B (60 points)** **Summer 2020, CU-Boulder**

(a) (5 pts) Rephrase this is as a graph problem. Give a precise definition of how to model this problem as a graph, including how to define a vertex and identify it's neighbouring vertices, and state the specific question about this graph that must be answered.

(b) (5 pts) Provide a 3-4 sentence description of how your algorithm works

(c) (15 pts) Provide a well commented pseudo-code or actual code to solve the above problem.

Name: Pourna Sengupta

ID: 109086577

**CSCI 3104, Algorithms** **Escobedo & Jahagirdar**
**Homework 5B (60 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/course-schedule-ii/

```
class Solution {
    public int[] findOrder(int numCourses, int[][] prerequisites) {
        List<Integer>[]graph = new LinkedList[numCourses];
        int[] degree = new int[numCourses];

        for(int i = 0; i < numCourses; i++){
            graph[i] = new LinkedList<>();
        }

        for(int j = 0; j < prerequisites.length; j++){
            graph[prerequisites[j][1]].add(prerequisites[j][0]);
            degree[prerequisites[j][0]]++;
        }
        Queue<Integer> queue = new LinkedList<>();
        for(int m = 0; m < numCourses; m++){
            if(degree[m] == 0){
                queue.add(m);
            }
        }

        int[] sched = new int[numCourses];
        int n = 0;
        while(!queue.isEmpty()){
            int k = queue.remove();
            sched[n] = k;
            n++;

            for(int val:graph[k]){
                degree[val]--;
                if(degree[val] == 0){
```

```
                    queue.add(val);
                }
            }
        }

        if(n != numCourses){
            sched = new int[0];
        }
        Arrays.stream(sched).forEach(x -> System.out.print(x + " "));
        System.out.println();
        return sched;
    }
}
```