CS 39000-ATA: Homework 3

Due on 2025-02-13 23:59

Prof. Kent Quanrud, Spring 2025

Mukul Agarwal and Kovid Tandon

Additional Collaborators: TA Cyril Sharma, Saad Sharief, Jayden Lee, and Maanas Karwa.

Problem 7.1

Each of the following problems describes a graph problem over an unweighted directed graph $G = (V, E)$ with m edges and n vertices. Each problem can be solved by constructing an auxiliary graph and calling a shortest path algorithm from this chapter as a black box. For each problem, (a) design an auxiliary graph and shortest path problem, (b) indicate which algorithm to apply and how, and (c) analyze the running time. No proof of correctness is required.

Notation 2.1: For this question we assume that every vertex is predefined a depth of $+\infty$, and BFS updates the depth of each vertex according to the distance from a given source vertex. Thus, if a walk is not possible, the depth returned from BFS will be $+\infty$. Note that pre-assigning every vertex a depth is a $\mathcal{O}(n)$ runtime operation, which never affects the final runtime.

3. **For two vertices s and t, compute the length of the shortest walk from s to t where the number of edges is either a multiple of 2, or a multiple of 3.**[1]

We define a useful helper algorithm.

Algorithm 1: Walk-Divisible$(G = (V, E), s, t, k)$: Given a directed graph G with vertices V and edges E, some $s, t \in V$, and some $k \in \mathbb{N}$, return the length of the shortest walk from s to t having length divisible by k (or $+\infty$ if no such walk exists).

```
Walk-Divisible($G = (V, E), s, t, k$)
    Let $G' = (V', E')$ be a directed unweighted graph
    // We label the vertices of $V$ as $v_1, v_2, ..., v_n$
    // WLOG, say $s = v_1$ and $t = v_2$
    for $i \in [n]$
        for $j \in \{0, 1, 2, ..., k-1\}$
            // A vertex corresponding to $v_i$ at path length $\equiv j \pmod{k}$
            Add to $V'$ a vertex $v_{i,j}$
        end for
    end for
    for $(v_\alpha, v_\beta) \in E$ do
        for $j \in [k]$ do
            Add an edge $\left(v_{\alpha,(j \bmod k)}, v_{\beta,((j+1) \bmod k)}\right)$ to $E'$
        end for
    end for
    return the depth of $v_{2,0}$ when performing BFS on $G'$ starting at $v_{1,0}$
end algorithm
```

Observation 2.1: Algorithm 1 constructs a new auxiliary graph $G' = (V', E')$ from $G = (V, E)$ which contains k layers. Each layer contains all n vertices from G, with each vertex $v_{i,j}$ representing the ith vertex from G in the jth layer of G'. Furthermore, for each edge $(v_\alpha, v_\beta) \in E$, j edges are added to E' such that there is an edge starting at the vertex corresponding to v_α at every layer, and ending at the vertex corresponding to v_β in the next

[1]For this and the other problems, if there is no such walk, then your algorithm should return $+\infty$.

corresponding layer (wrapping around from the kth layer to the 0th layer). Each layer can be thought to represent the path length $(\bmod k)$ from the 0th layer. Thus, for a path length to be divisible by k, the path must end at the 0th layer, since all multiples of $k \ (\bmod k) = 0$.

The actual algorithm follows directly from this.

Algorithm 2: Shortest-2,3-Walk$(G = (V, E), s, t)$: Given a directed, unweighted graph G with V vertices and E edges, and some vertices $s, t \in V$, return the length of the shortest walk from s to t in V where the length of the path is divisible by either 2 or 3.

```
Shortest-2,3-Walk$(G = (V, E), s, t)$
    **return** $\min\{$Walk-Divisible$(G, s, t, 2)$, Walk-Divisible$(G, s, t, 3))$
**end algorithm**
```

Consequently, we see the runtime of the algorithm.

Proposition 2.1 *(Runtime of Algorithm 2)*: The runtime of computing the length of the shortest walk from s to t where the number of edges is either a multiple of 2 or 3 when using Shortest-2,3-Walk is $\mathcal{O}(m + n)$, where $n := |V|$ and $m := |E|$.

Proof: Shortest-2,3-Walk returns the minimum of Walk-Divisible$(G, s, t, 2)$ and Walk-Divisible$(G, s, t, 3)$, which is an $\mathcal{O}(1)$ operation excluding the runtime taken to execute the Walk-Divisible calls.

Let us first consider Walk-Divisible$(G, s, t, 2)$. Note that constructing the auxiliary graph with 2 layers is a $\mathcal{O}(2(m + n))$ operation. Furthermore, since the auxiliary graph has $2n$ vertices and $2m$ edges, computing the depth using BFS on the auxiliary graph is a $\mathcal{O}(2(m + n))$ operation. Thus the total runtime of Walk-Divisible$(G, s, t, 2)$ is $\mathcal{O}(2(m + n)) + \mathcal{O}(2(m + n)) = \mathcal{O}(m + n)$. An identical argument for Walk-Divisible$(G, s, t, 3)$ will show that the runtime for this call is also $\mathcal{O}(m + n)$.

Thus, the final runtime of Shortest-2,3-Walk is the runtime of both Walk-Divisible calls with the $\mathcal{O}(1)$ operation to take the minimum. Thus, the final runtime is,

$$\mathcal{O}(m+n) + \mathcal{O}(m+n) + \mathcal{O}(1) = \mathcal{O}(m+n)$$

$\square$

4. **For two vertices s and t, compute the length of the shortest walk from s to t where the number of edges is of the form $5x + 8y$ for $x, y \in \mathbb{Z}_{\geq 0}$.**

Algorithm 3: Shortest-$5x + 8y$-Walk$(G = (V, E), s, t)$: Given a directed unweighted graph G with vertices V and edges E and some vertices return the length of the shortest walk from s to t which can be expressed in the form $5x + 8y$ for $x, y \in \mathbb{Z}_{\geq 0}$.

```
Shortest-$5x + 8y$-Walk$(G = (V, E), s, t)$
    Let $G' = (V', E')$ be a directed unweighted graph
    // We label the vertices of $V$ as $v_1, v_2, ..., v_n$
    // WLOG, say $s = v_1$ and $t = v_2$
    for $i \in [n]$
        for $j \in \{0, 1, 2, ..., 7\}$
            Add to $V'$ a vertex $v_{i,j}$
        end for
    end for
    for $(v_\alpha, v_\beta) \in E$ do
        for $j \in \{0, 1, 2, ..., 7\}$ do
            Add an edge $(v_{\alpha,j}, v_{\beta,(j+1) \bmod 8})$ to $E'$
            if $j = 4$ then
                Add an edge $(v_{\alpha,j}, v_{\beta,0})$
            end if
        end for
    end for
    return the depth of $v_{2,0}$ when performing BFS on $G'$ starting at $v_{1,0}$.
end algorithm
```

Observation 2.2: Algorithm 3 constructs a new auxiliary graph $G' = (V', E')$ from $G = (V, E)$ which contains 8 layers. Each layer contains all n vertices from G, with each vertex $v_{i,j}$ representing the ith vertex from G in the jth layer of G'. Furthermore, for each edge $(v_\alpha, v_\beta) \in E$, j edges are added to E' such that there is an edge starting at the vertex corresponding to v_α at every layer, and ending at the vertex corresponding to v_β in the next corresponding layer (wrapping around from the 7th layer to the 0th layer). Furthermore, for each edge $(v_\alpha, v_\beta) \in E$, an extra edge is also inserted from the 4th layer to the 0th layer. If we treat each layer as the path length, note that the path can wrap around from either the 4th or the 7th layer back to the 0th layer. This ensures that the length of the path at layer 0 must be of the form $5x + 8y$ for some $x, y \in \mathbb{Z}_{\geq 0}$.

Proposition 2.2 *(Runtime of Algorithm 3)*: The runtime of computing the length of the shortest walk with length $5x + 8y$ for some $x, y \in \mathbb{Z}_{\geq 0}$ between two vertices in a graph $G = (V, E)$ with Algorithm 3 has asymptotic time complexity $\mathcal{O}(m + n)$ where $m := |E|$ and $n := |V|$.

Proof: Note that creating the auxiliary graph is a $\mathcal{O}(m + n)$ operation. This is because adding all the vertices to V' is a $\mathcal{O}(7n)$ operation, while adding all the edges to E' is a $\mathcal{O}(7m)$ operation. Furthermore, since the auxiliary graph has $7n$ vertices and $8m$ edges, computing the depth using BFS on the auxiliary graph is a $\mathcal{O}(8m + 7n) = \mathcal{O}(m + n)$ operation. Thus, the final runtime for Algorithm 3 is,

$$\mathcal{O}(7n) + \mathcal{O}(7m) + \mathcal{O}(m + n) = \mathcal{O}(m + n)$$

□

5. **Suppose G is a patriotic directed graph where all the edges are colored either red, white, or blue. An *American walk* is a walk where the edges alternate in color in the order of the American dream: red, white, blue, red, white, blue... Here the first edge could be any color and then the colors have to cycle through the American dream thereafter. Compute the length of the shortest American walk from s to t.**

Algorithm 4: Walk-American$(G = (V, E), s, t)$: Given a directed graph G with vertices V and edges E, some $s, t \in V$, and, return the length of the shortest American walk from s to t (or $+\infty$ if no such walk exists)

```
Walk-American$(G = (V, E), s, t)$
    Let $G' = (V', E')$ be a directed unweighted graph
    // We label the vertices of $V$ as $v_1, v_2, ..., v_n$
    // WLOG, say $s = v_1$ and $t = v_2$
    for $i \in [n]$
        for $j \in \{\text{red}, \text{white}, \text{blue}\}$
            Add to $V'$ a vertex $v_{i,j}$ // outgoing edge colored $j$
        end for
    end for
    for $e = (v_\alpha, v_\beta) \in E$ do
        let $c$ be the color of $e$ and $c'$ be the color which should be subsequent
        Add an edge $(v_{\alpha,c}, v_{\beta,c'})$ to $E'$
    end for
    Add vertices $s^*, t^*$ to $V'$
    for $j \in \{\text{red}, \text{white}, \text{blue}\}$ do
        Add edges $(s^*, v_{1,j})$ and $(v_{2,j}, t^*)$ to $E'$
    end for
    let $d$ be the depth of $t^*$ when performing BFS on $G'$ starting at $s^*$
    if $d = +\infty$ then return $+\infty$
    else return $d - 2$
end algorithm
```

Observation 2.3: Algorithm 4 constructs a new auxiliary graph $G' = (V', E')$ from $G = (V, E)$ which contains 3 layers, each representing one of the colors red, white, or blue. Each layer contains all n vertices from G, with each vertex $v_{i,j}$ representing the ith vertex from G in the jth layer of G'. Furthermore, for each edge $(v_\alpha, v_\beta) \in E$, 1 edge is added to E' such that the edge starts at vertex corresponding to v_α in the layer of its color, and ends at the vertex corresponding to v_β in the layer of the next color in the American walk order. Furthermore, a super source, s^*, is added, and edges from s^* to each of the three starting nodes at each layer are added as well. Similarly, a super sink, t^* is added along with edges from all three destination nodes to t^*. The addition of the super source and sink adds 2 to every path from the source to the destination, thus the final result substracts 2 from the final shortest path.

Proposition 2.3 *(Runtime of Algorithm 4)*: The computation of the shortest American walk between any two vertices in $G = (V, E)$ using Walk-American has asymptotic time complexity $\mathcal{O}(m + n)$, where $m := |E|$ and $n := |V|$.

Proof: Note that creating the auxiliary graph is a $\mathcal{O}(m + n)$ operation. This is because adding all the vertices to V' is a $\mathcal{O}(3n)$ operation, while adding all the edges to E' is a $\mathcal{O}(m + 6)$ operation, since for each edge in G, 1 edge is added to E', and 3 edges are added for both the super source and sink. Furthermore, since the auxiliary graph has $3n$ vertices and $m + 6$ edges, computing the depth using BFS on the auxiliary graph is a $\mathcal{O}(3n + m + 6) = \mathcal{O}(m + n)$ operation. Thus, the final runtime for Walk-American is,

$$\mathcal{O}(3n) + \mathcal{O}(m + 6) + \mathcal{O}(m + n) = \mathcal{O}(m + n)$$

□

6. **Suppose G is again a patriotic directed graph where all the edges are colored either red, white, or blue. An *un-American* walk is a walk that never cycles through the American dream; that is, a walk where no three consecutive edges in the walk have colors that form any of the following three sequences: (red, white, blue),(white, blue, red), or (blue, red, white). Compute the length of the shortest un-American walk from s to t.**

Algorithm 5: Walk-Un-American$(G = (V, E), s, t)$: Given a directed graph G with vertices V and edges E, some $s, t \in V$, and, return the length of the shortest un-American walk from s to t (or $+\infty$ if no such walk exists)

```
Walk-Un-American$(G = (V, E), s, t)$
    Let $G' = (V', E')$ be a directed unweighted graph
    // We label the vertices of $V$ as $v_1, v_2, ..., v_n$
    // WLOG, say $s = v_1$ and $t = v_2$
    Let $L = \{R, W, B, \text{RR}, \text{RW}, \text{RB}, \text{WR}, \text{WW}, \text{WB}, \text{BR}, \text{BW}, \text{BB}\}$
    for $i \in [n]$
        for $j \in L$
            Add to $V'$ a vertex $v_{i,j}$
            // $j$ represents the color of up to two previous edges
        end for
    end for
    for $e = (v_\alpha, v_\beta) \in E$ do
        let $c$ be the color of $e$
        if $c$ is Red then do
            Add edges $(v_{\alpha,R}, v_{\beta,\text{ RR}}), (v_{\alpha,R}, v_{\beta,\text{ RW}}), (v_{\alpha,R}, v_{\beta,\text{ RB}})$ to $E'$
            Add edges $(v_{\alpha,\text{ RR}}, v_{\beta,\text{ RR}}), (v_{\alpha,\text{ RR}}, v_{\beta,\text{ RW}}), (v_{\alpha,\text{ RR}}, v_{\beta,\text{ RB}})$ to $E'$
            Add edges $(v_{\alpha,\text{ WR}}, v_{\beta,\text{ RR}}), (v_{\alpha,\text{ WR}}, v_{\beta,\text{ RW}}), (v_{\alpha,\text{ WR}}, v_{\beta,\text{ RB}})$to $E'$
            Add edges $(v_{\alpha,\text{ BR}}, v_{\beta,\text{ RR}}), (v_{\alpha,\text{ BR}}, v_{\beta,\text{ RB}})$ to $E'$
        else if $c$ is White then do
            Add edges $(v_{\alpha,W}, v_{\beta,\text{ WR}}), (v_{\alpha,W}, v_{\beta,\text{ WW}}), (v_{\alpha,W}, v_{\beta,\text{ WB}})$ to $E'$
            Add edges $(v_{\alpha,\text{ RW}}, v_{\beta,\text{ WR}}), (v_{\alpha,\text{ RW}}, v_{\beta,\text{ WW}})$ to $E'$
            Add edges $(v_{\alpha,\text{ WW}}, v_{\beta,\text{ WR}}), (v_{\alpha,\text{ WW}}, v_{\beta,\text{ WW}}), (v_{\alpha,\text{ WW}}, v_{\beta,\text{ WB}})$to $E'$
            Add edges $(v_{\alpha,\text{ BW}}, v_{\beta,\text{ WR}}), (v_{\alpha,\text{ BW}}, v_{\beta,\text{ WW}}), (v_{\alpha,\text{ BW}}, v_{\beta,\text{ WB}})$ to $E'$
        else if $c$ is Blue then do
            Add edges $(v_{\alpha,B}, v_{\beta,\text{ BR}}), (v_{\alpha,B}, v_{\beta,\text{ BW}}), (v_{\alpha,B}, v_{\beta,\text{ BB}})$ to $E'$
            Add edges $(v_{\alpha,\text{ RB}}, v_{\beta,\text{ BR}}), (v_{\alpha,\text{ RB}}, v_{\beta,\text{ BW}}), (v_{\alpha,\text{ RB}}, v_{\beta,\text{ BB}})$ to $E'$
            Add edges $(v_{\alpha,\text{ WB}}, v_{\beta,\text{ BW}}), (v_{\alpha,\text{ WB}}, v_{\beta,\text{ BB}})$to $E'$
            Add edges $(v_{\alpha,\text{ BB}}, v_{\beta,\text{ BR}}), (v_{\alpha,\text{ BB}}, v_{\beta,\text{ BW}}), (v_{\alpha,\text{ BB}}, v_{\beta,\text{ BB}})$ to $E'$
        end for
    Add vertices $s^*, t^*$ to $V'$
    for $j \in \{R, W, B\}$ do
        Add edges $(s^*, v_{1,j})$ to $E'$
    end for
    for $j \in L$ do
        Add edges $(v_{2,j}, t^*)$ to $E'$
    end for
    let $d$ be the depth of $t^*$ when performing BFS on $G'$ starting at $s^*$
    if $d = +\infty$ then return $+\infty$
    else return $d - 2$
end algorithm
```

Observation 2.4: Algorithm 5 constructs a new auxiliary graph $G' = (V', E')$ from $G = (V, E)$ which contains 12 layers, with three of the layers representing the colors red, white, and blue, and the other 9 representing all possible combinations of size 2 of the 3 colors. Each layer contains all n vertices from G, with each vertex $v_{i,j}$ representing the ith vertex from G in the jth layer of G'. The three layers with just one color, represent the color of the starting edge of the shortest path, while for the layers with two colors, the first color represents the color of the edge leading into any vertex in the layer, and the second color represents the color of the edge leaving the layer. Edges are added to G' such that they satisfy the above conditions, while ensuring that no American walk is possible. Furthermore, a super source, s^*, is added, and edges from s^* to each of the three starting nodes at each layer are added as well. Similarly, a super sink, t^* is added along with edges from all three destination nodes to t^*. The addition of the super source and sink adds 2 to every path from the source to the destination, thus the final result substracts 2 from the final shortest path.

Proposition 2.4 *(Runtime of Algorithm 5)*: The computation of the shortest un-American walk between any two vertices in $G = (V, E)$ using Walk-Un-American has asymptotic time complexity $\mathcal{O}(m + n)$, where $m := |E|$ and $n := |V|$.

Proof: Note that creating the auxiliary graph is a $\mathcal{O}(m + n)$ operation. This is because adding all the vertices to V' is a $\mathcal{O}(12n)$ operation, while adding all the edges to E' is a $\mathcal{O}(m)$ operation, since there are exactly 12 layers of vertices, and some constant number of edges are added to the auxiliary graph per edge in G. Furthermore, since the auxiliary graph has $\mathcal{O}(n)$ vertices and $\mathcal{O}(m)$ edges, computing the depth using BFS on the auxiliary graph is a $\mathcal{O}(m + n)$ operation. Thus, the final runtime for Walk-American is,

$$\mathcal{O}(12n) + \mathcal{O}(m) + \mathcal{O}(m + n) = \mathcal{O}(m + n)$$

□

7. **Let $s, t \in V$ be two vertices. Consider the following game with two people Alice and Bob. Initially, Alice is on s, and Bob is on t. Each round, Alice and Bob are on two vertices, and they simultaneously take an outgoing edge to another vertex. The goal is to guide Alice and Bob to the same vertex with the minimum number of rounds or declare that it is impossible to have them meet. Compute the minimum number of rounds needed to have Alice and Bob meet at the same vertex.**

Algorithm 6: Find-Meeting-Spot$(G = (V, E), s, t)$: Given a unweighted directed graph G with vertices V and edges E, and some vertices $s, t \in V$, return the minimum number of rounds needed to have Alice and Bob meet at the same Vertex (or $+\infty$ if Alice and Bob can never meet)

```
Find-Meeting-Spot($G = (V, E), s, t$)
    if $s = t$ then
        return 0
    end if
    let $G' \leftarrow (V', E')$ be an empty graph
    let $V' \leftarrow \{(u, v) : u, v \in V\}$
    let $E' \leftarrow \{((u, v), (u', v')) : (u, u'), (v, v') \in E\}$
    Compute the depths of all nodes when using BFS on $G'$ starting at $(s, t)$
    return $\min\Big\{\text{depth}_{(s,t)}((v, v)) : v \in V\Big\}$.
end algorithm
```

Observation 2.5: Algorithm 6 constructs a new auxiliary graph $G' = (V', E')$ from $G = (V, E)$. Each vertex in G' represents a pair of vertices in G. Thus, there are exactly n^2 vertices in G'. Furthermore, an edge $((u, v), (u', v'))$ is only added to E' when both $(u, u'), (v, v') \in E$. Thus, the graph G' represents all the possible moves that a given source and destination can take at a time. Thus, a node that both the source and destination can meet at, is a node where the pair of vertices contains the same vertex.

Proposition 2.5: The runtime of computing the minimum number of rounds in which Alice and Bob might meet in a directed unweighted graph $G = (V, E)$ using Find-Meeting-Spot has asymptotic time complexity $\mathcal{O}(m^2 + n^2)$ where $n := |V|$ and $m := |E|$.

Proof: Notice that we construct a new graph G' using G; notice that G' has n^2 vertices which we compute in $\mathcal{O}(n^2)$ time by iterating over pairs of vertices in G; G' has m^2 edges which we construct in $\mathcal{O}(m^2)$ time by iterating over the edges in G. We then perform breadth-first-search on G' which takes time $\mathcal{O}(m^2 + n^2)$. Finally, we iterate ove a subset of the vertices in G' and compute the minimum of the depths from the breadth-first-search for those vertices, taking time $\mathcal{O}(n)$. We also perform some extra $\mathcal{O}(1)$ computation. Hence, letting $T(m, n)$ be the time taken to execute this algorithm on a graph with m edges and n vertices,

$$T(m, n) \in \mathcal{O}(n^2) + \mathcal{O}(m^2) + \mathcal{O}(m^2 + n^2) + \mathcal{O}(n) + \mathcal{O}(1) = \mathcal{O}(m^2 + n^2),$$

whence the desired result. $\square$