# *The study of rainbow coloring of graphs and graph coloring in streaming*

*Anannya Upasana*

# The study of rainbow coloring of graphs and graph coloring in streaming

**DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF**

Master of Technology
in
Computer Science

by


**Anannya Upasana**
[ Roll No: CS-1801 ]


under the guidance of

**Dr. Sourav Chakraborty**
Advanced Computing and Microelectronics Unit


**Indian Statistical Institute**
**Kolkata-700108, India**

**July 2020**

# Acknowledgement

I would like to show my gratitude to my advisor, *Dr. Sourav Chakraborty*, Associate Professor, Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Kolkata, for his invaluable guidance and support. I am grateful to him for giving me the opportunity to work on and learn about a new topic, rainbow coloring, which in turn led me to imbibe various other concepts.

I would like to thank *Dr. Arijit Bishnu*, Associate Professor, Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Kolkata, for his insightful comments, suggestions and continuous encouragement. I am grateful to him for introducing me to the field of streaming and giving me an opportunity to work on a research project that has ensured a steep learning curve.

I would sincerely like to express my appreciation to *Gopinath Mishra*, SRF, Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Kolkata, for his time and effort in helping me understand various concepts that helped me with this work and writing technically rigorous proofs.

This work is part of a research work that has been submitted to a conference.

Finally, I am very much thankful to my parents and my sister for their everlasting support. Last but not the least, I would like to thank all of my friends for their help and support.

**Anannya Upasana**
Indian Statistical Institute
Kolkata - 700108 , India.

# Abstract

Graph coloring is a well known problem with wide-ranging applications. The vertex and edge coloring problems have been studied in various models of computation. Rainbow coloring is a type of edge coloring that also acts as a connectivity measure for graphs. A graph is said to be rainbow colored or rainbow connected if there exists an edge coloring such that every pair of vertices, if connected, is connected by a path having distinct colors for all edges contained in it. The verification of rainbow coloring is an NP-Complete problem whereas the problems of verifying vertex and edge coloring admit easy solutions in the RAM model. Verification of graph coloring in the streaming model of computation is a problem that has not been studied before. We focus on the vertex coloring problem in the streaming model and give algorithms that verify if a given vertex coloring is valid with a high probability. We also give lower bounds for verifying vertex coloring in a few streaming models.

**Keywords**: *Graph coloring, Streaming, Vertex coloring, Rainbow coloring*

# Contents

# Chapter 1

# Introduction

## 1.1 Introduction

Graph coloring is an ubiquitous problem in computer science and has widespread practical applications. The problem of graph coloring can be defined as the assignment of colors to different elements of the graph, provided certain constraints are satisfied. The various graph coloring problems that has been widely studied is vertex coloring, edge coloring and rainbow coloring.

Vertex coloring is the assignment of colors to vertices of the graph with the constraint that adjacent vertices do not get the same colors. Edge coloring is a graph coloring problem where you assign colors to the edges of the graph such that edges incident on the same vertex get assigned different colors.

Rainbow connectivity is a graph coloring problem that is also a connectivity measure for graphs. It was introduced by Chartrand et al. in 2008 [9]. Rainbow coloring is a special type of edge coloring where, for every pair of vertices in the graph, there should exist a path connecting the pair where every edge gets assigned a distinct color.

A lot of work has been done in this field in the RAM model of computation. While the problems of vertex coloring and edge coloring have been solved using different algorithmic paradigms like the greedy approach, rainbow coloring is a NP-Complete problem [7]. For a graph of maximum degree $\Delta$, a $\Delta + 1$ vertex coloring of the graph exists, and a $\Delta + 1$ edge coloring of the graph exists as well. The vertex coloring can be found easily using a greedy algorithm [16]. Misra and Gries gave a polynomial-time algorithm for the edge coloring of a graph [15]. Verifying if a given vertex coloring or edge coloring is proper also admits easy solution in the RAM model. Verifying if a graph is rainbow-colored, however, remains a NP-Complete problem [7].

In the streaming model of computation, while a lot of work has been done on finding a proper vertex and edge coloring for graphs [3–5], rainbow coloring is an untouched

field. Moreover, an unprecedented problem is the verification of any kind of coloring in the streaming model. In this work, we have tried to devise efficient algorithms for verification of graph coloring problems in streaming and have successfully found algorithms that solve the verification problem for vertex coloring in this model. The streaming models we consider are as follows:

- The vertex arrival model, where the vertices of a graph stream in an arbitrary order.
- The vertex arrival model with degree oracle, where the vertices of a graph stream in an arbitrary order and we also have access to a degree oracle using which we can query for the degree of a vertex in the stream.
- The vertex arrival model with random order, where the vertices of a graph stream in a random fashion.
- The adjacency list model, where the vertices stream in an arbitrary order and the neighborhood of each vertex is also exposed along with it in the stream.

In each of the models described above, the color assigned to the vertex also streams along with it. We also provide lower bounds for the verification problem for vertex coloring in two of the streaming models.

## 1.2 Notations

We denote the set $\{1, 2, ..., n\}$ by $[n]$ and $\mathbb{N}$ is the set of natural numbers. Let $G(V, E)$ be a connected graph where $V(G)$ and $E(G)$ denote the non-empty finite set of vertices and the finite set of edges of the graph respectively. We'll denote $n$ and $m$ as the number of vertices and edges of a graph, i.e., $|V(G)| = n$ and $|E(G)| = m$. For any vertex $v \in V(G)$, we denote $deg_G(v)$ as the degree of the vertex $v$ and $N_G(v)$ as its neighborhood in the graph $G$. We denote $deg^-_G(v)$ and $deg^+_G(v)$ as the number of neighbors of $v$ to the left and right of it in the stream. We let $d(u, v)$ denote the length of the path from $u$ to $v$, i.e., the number of edges in the path from $u$ to $v$. The diameter of a graph, which is the largest distance between any pair of vertices, will be denoted by $diam_G$.

Let $\mathbb{E}[X]$ denote the expectation of the random variable $X$. $Pr(E)$ denotes the probability of an event $E$. The statement "event $E$ occurs with high probability" is equivalent to $Pr(E) \geq 1 - \frac{1}{n^c}$, where $c$ is an absolute constant. By polylogarithmic, we mean $\mathcal{O}\left((\log n/\varepsilon)^{\mathcal{O}(1)}\right)$. The notation $\widetilde{\mathcal{O}}(\cdot)$ hides a polylogarithmic term in $\mathcal{O}(\cdot)$.

## 1.3 Our Contribution

We solve the problem of verifying vertex coloring of graphs in different models of streaming. In this problem, we consider the vertices of a graph to arrive in a stream and associate a color with each of them. The goal is to verify if the given coloring is a proper coloring of the graph or not. We formally state the problem in Chapter 5. The streaming models we consider are the vertex arrival model, the vertex arrival model with degree oracle, the vertex arrival model with random order stream and the adjacency list model. The main contribution is as follows:

- We give an algorithm that solves the verification problem in the vertex arrival streaming model with high probability in space $\widetilde{\mathcal{O}}\left(\min\{|V|, \frac{|V|^2}{\varepsilon|E|}\}\right)$, for an input parameter $\varepsilon > 0$.
- We give an algorithm that solves the verification problem in the vertex arrival streaming model with degree oracle with high probability in space $\tilde{\mathcal{O}}\left(\min\left(|V|, \frac{1}{\varepsilon}\right)\right)$, for an input parameter $\varepsilon > 0$.
- We give an algorithm that solves the verification problem in the vertex arrival random order streaming model with high probability in space $\widetilde{\mathcal{O}}\left(\frac{|V|}{\sqrt{\varepsilon|E|}}\right)$, for an input parameter $\varepsilon > 0$.
- We give an algorithm that solves the verification problem in the adjacency list streaming model with high probability in space $\tilde{\mathcal{O}}\left(\min\left(|V|, \frac{1}{\varepsilon}\right)\right)$, for an input parameter $\varepsilon > 0$.

We would like to observe here that verifying vertex coloring in the edge arrival model is easy. In the edge arrival model, the edges of the graph arrive in an arbitrary order in the stream. When an edge arrives in the stream, we have access to the colors of both its endpoints. This enables to check every edge for conflict, get the exact number of conflicting (or monochromatic) edges and check if the coloring is valid or not.

We also prove lower bounds for the verification problem in the vertex arrival model and the vertex arrival model with degree oracle.

- The space lower bound for the verification problem in the vertex arrival model is $\Omega\left(\frac{|V|}{\sqrt{|E|}}\right)$.
- The space lower bound for the verification problem in the vertex arrival streaming model with degree oracle is $\Omega\left(\frac{1}{\varepsilon}\right)$, for an $\varepsilon > 0$.

## 1.4 Thesis Outline

We start with defining the rainbow coloring problem and its variants in Chapter 2. In Chapter 3, we give already proven results on the rainbow coloring of certain graph classes to give a better understanding of the measure. Chapter 4 deals with results on the hardness of rainbow coloring as well as its parameterized variants.

In Chapter 5, we state and prove our results on the verification of vertex coloring in different streaming models.

We conclude by identifying some open problems in Chapter 6.

# Chapter 2

# Rainbow Coloring

## 2.1 Rainbow Coloring: Definition

Let $G$ be a non-trivial connected graph. An edge coloring on $G$ is defined as $c : E(G) \to [k]$ where $k \in \mathbb{N}$. We call a path in $G$ to be a `rainbow path` if every edge in the path gets a distinct color. An edge-colored graph $G$ is said to be `rainbow colored` (or `rainbow connected`) if for every pair of vertices $u, v \in V(G)$, there is a rainbow path connecting $u$ and $v$ and such a coloring is said to be a `rainbow coloring`. If $G$ is rainbow connected then it is also connected. The minimum number of colors required to make $G$ rainbow connected is called the `rainbow connection number` and is denoted by $rc(G)$. A coloring with only $rc(G)$ colors that makes $G$ rainbow connected is called a minimum rainbow coloring.

Every graph has a trivial edge coloring that makes it rainbow connected. In this trivial edge coloring, every edge gets a distinct edge color, and $rc(G) = m$, where $|E(G)| = m$. So, a loose general upper bound for the rainbow connection number $rc(G)$ is $m$. A lower bound for the rainbow connection number is the diameter of the graph.

**Remark 1** *For any graph $G$, $diam \leq rc(G) \leq m$ where $diam_G$ and $m$ are the diameter and the number of edges of $G$.*

Rainbow connectivity can also be defined for directed graphs. The study on directed graphs was initiated by Ananth et al. [2]. The directed graphs in consideration here are connected.

A directed graph $G$ is said to be rainbow connected if for every pair of vertices $u$ and $v$, there is a directed path either from $u$ to $v$, or from $v$ to $u$, and the directed path is a rainbow path. The rainbow connection number of a directed path is also defined similarly. The minimum number of colours needed to make a directed graph rainbow connected is the rainbow connection number of the directed graph [2].

**Remark 2 ( [2])** *The rainbow connection number of a directed graph is atleast the rainbow connection number of its underlying undirected graph.*

**Theorem 1 ( [2])** *For a directed graph $G$, it is NP-Complete to decide whether $rc(G) \leq 2$.*

The above result can be extended for any value of $k > 2$.

**Lemma 1 ( [2])** *For a directed graph $G$, it is NP-Complete to decide whether $rc(G) \leq k$.*

## 2.2 Variants

A geodesic in a graph $G$ is the shortest path between two vertices $u$ and $v$ in $G$. The `rainbow` $u - v$ `geodesic` is a rainbow path from $u$ to $v$ of length $d(u, v)$, where $d(u, v)$ is the length of the path from $u$ to $v$. A graph is said to be `strongly rainbow connected` if for all pairs of distinct vertices $u, v \in V(G)$, there exists a rainbow $u-v$ geodesic or a shortest path from $u$ to $v$ which is a rainbow path. The minimum number of colors required to make $G$ strongly rainbow connected is called the `strong rainbow connection number`, denoted by $src(G)$, and the coloring is called a `strong rainbow coloring` if $src(G)$ colors are used.

**Remark 3** *For any graph $G$, $rc(G) \leq src(G)$.*

Motivated by proving upper bounds on the $src(G)$ of a graph $G$, [8] came up with the concept of `very strong rainbow connectivity`. A graph is said to be `very strongly rainbow connected` if for all pairs of distinct vertices $u, v \in V(G)$, every shortest path between $u$ and $v$ is a rainbow path. The minimum number of colors required to make $G$ very strongly rainbow connected is called the `very strong rainbow connection number`, denoted by $vsrc(G)$, and the coloring is called a `very strong rainbow coloring` if $vsrc(G)$ colors are used.

**Remark 4** *For any graph $G$, $rc(G) \leq src(G) \leq vsrc(G)$.*

The `k-subset rainbow connectivity` is another variant of the rainbow connectivity problem, introduced by [7]. The input to the k-subset rainbow connectivity problem is a graph $G = (V, E)$ along with a set of pairs $P = \{(u, v) : (u, v) \subseteq V \times V\}$ and an integer $k$. The objective is to answer whether there exists an edge coloring of G with at most $k$ colors such that every pair $(u, v) \in P$ has a geodesic rainbow path.

The `k-steiner rainbow coloring`, introduced by [12], takes a graph $G$ and a set $S \subseteq V(G)$ and decides if for all $u, v \in S$, there is a rainbow path between $u$ and $v$ in S.

# Chapter 3

# Rainbow Connection numbers of certain graph classes

Chartrand et al. [9] were the first to study rainbow connection of graphs and determine the values for $rc(G)$ and $src(G)$ of various graph classes.

**Proposition 1 ( [9])** *Let $G$ be a non-trivial connected graph of size $m$. Then,*

1. *$src(G) = 1$ if and only if $G$ is a complete graph,*
2. *$rc(G) = 2$ if and only if $src(G) = 2$, and*
3. *$rc(G) = m$ if and only if $G$ is a tree.*

## 3.1 Cycles and Wheels

This proposition also implies that if a connected graph $G$ of size $m$ has $src(G) = m$, then $G$ is a tree. [9]

**Proposition 2 ( [9])** *For each integer $n \geq 4$, $rc(C_n) = src(C_n) = \lceil \frac{n}{2} \rceil$, where $C_n$ is a cycle of order $n$.*

The wheel $W_n$, for $n \geq 3$, is constructed by adding an edge from a new vertex to every vertex of $C_n$.

**Proposition 3 ( [9])** *For $n \geq 3$, the rainbow connection number of the wheel $W_n$ is*

$$rc(W_n) = \begin{cases} 1 & n = 3 \\ 2 & 4 \leq n \leq 6 \\ 3 & n \geq 7 \end{cases}$$

*The strong rainbow connection number of the wheel $W_n$ is $src(W_n) = \lceil \frac{n}{3} \rceil$*

## 3.2 Bipartite and Complete k-partite graphs

They further establish the strong rainbow connection numbers of the complete bipartite graph, and use this result to determine the strong rainbow connection numbers of the complete $k$-partite graph.

**Theorem 2 ( [9])** *For integers $s$ and $t$ with $1 \leq s \leq t$, $src(K_{s,t}) = \lceil \sqrt[s]{t} \rceil$ where $K_{s,t}$ is the complete bipartite graph.*

**Theorem 3 ( [9])** *For integers $s$ and $t$ with $2 \leq s \leq t$, $rc(K_{s,t}) = \min(\sqrt[s]{t}, 4)$*

**Theorem 4 ( [9])** *Let $G = K_{n_1,n_2,...,n_k}$ be a complete $k$-partite graph, where $k \geq 3$ and $n_1 \leq n_2 \leq ... \leq n_k$ such that $s = \sum_{i=1}^{k-1} n_i$ and $t = n_k$. Then,*

$$src(G) = \begin{cases} 1 & n_k = 1 \\ 2 & n_k \geq 2 \text{ and } s > t \\ \lceil \sqrt[s]{t} \rceil & s \leq t \end{cases}$$

**Theorem 5 ( [9])** *Let $G = K_{n_1,n_2,...,n_k}$ be a complete $k$-partite graph, where $k \geq 3$ and $n_1 \leq n_2 \leq ... \leq n_k$ such that $s = \sum_{i=1}^{k-1} n_i$ and $t = n_k$. Then,*

$$rc(G) = \begin{cases} 1 & n_k = 1 \\ 2 & n_k \geq 2 \text{ and } s > t \\ \min(\lceil \sqrt[s]{t} \rceil, 3) & s \leq t \end{cases}$$

**Theorem 6 ( [9])** *Let $a$ and $b$ be integers with $a \geq 4$ and $b \geq \frac{5a-6}{3}$. Then there exists a connected graph $G$ such that $rc(G) = a$ and $src(G) = b$*

# Chapter 4

# On the hardness of rainbow connectivity and its variants

## 4.1 Hardness of rainbow connectivity

Chakraborty et al. solved two of the conjectures posed by [6] and proved the following complexity results:

**Theorem 7 ( [7])** *Given a graph $G$, computing $rc(G)$ is NP-Hard.*

**Theorem 8 ( [7])** *Given a graph $G$, deciding if $rc(G) = 2$ is NP-Complete.*

They prove these results by reducing 3-SAT to the problem of extending a partial edge coloring with 2 colors in a graph to a complete edge coloring such that the resulting graph is 2-rainbow connected. This problem is polynomially reducible to the problem of 2-subset rainbow connectivity, which in turn is polynomially equivalent to the problem of 2-rainbow connectivity.

The above theorem led to a conjecture by the authors of [7] that deciding whether $rc(G) \leq k$ is NP-Complete for every fixed $k$. This conjecture was proved by [2]. They observed that the proof by [7] sufficed for every even $k > 1$. Furthermore, they cemented it by proving NP-Completeness for odd values of $k$.

**Remark 5 ( [7])** *For every even integer $k \geq 2$, deciding if $rc(G) \leq k$ is NP Complete.*

**Theorem 9 ( [2])** *For every odd integer $k \geq 3$, deciding if $rc(G) \leq k$ is NP Complete.*

This is proved by a reduction from k-subset rainbow connectivity problem, which itself is a hard problem.

**Lemma 2 ( [2])** *For $k \geq 3$, both the problems, k-subset strong rainbow connectivity and k-subset rainbow connectivity are NP-Hard, even when the input graph G is a star.*

Le and Tuza gave an alternate hardness proof for $k > 1$ [14].

**Conjecture 1 (Exponential Time Hypothesis [10])** *There exists a constant $c > 0$, such that there is no algorithm solving 3-SAT in time $\mathcal{O}^*(2^{cn})$.*

**Theorem 10 ( [12])** *For any $k \geq 2$, there is no algorithm for k-rainbow coloring running in time $2^{o(n^{3/2})}$, unless Exponential Time Hypothesis fails.*

They claim that the k-rainbow connectivity problem is the first NP-Complete graph problem for which the existence of a $2^{o(n^{1+\epsilon})}$ time algorithm is excluded, for an $\epsilon \geq 0$. They pose the following conjectures:

**Conjecture 2 ( [12])** *For any integer $k \geq 2$, there is no $2^{o(|E|)}n^{\mathbb{O}(1)}$-time algorithm for Rainbow k-Coloring, unless ETH fails.*

**Conjecture 3 ( [12])** *For any integer $k \geq 2$, there is no $2^{o(n^2)}n^{\mathbb{O}(1)}$-time algorithm for Rainbow k-Coloring, unless ETH fails.*

In a bid to solve the above conjectures, [1] came up with the following results:

**Theorem 11** *For $k \geq 3$, k-rainbow coloring does not admit an algorithm running in time $2^{o(|E(G)|)n^{\mathcal{O}(1)}}$, unless Exponential Time Hypothesis fails.*

The original conjecture of [12] was for $k \geq 2$, so the conjecture remains partially solved.

We would also like to remark that not only is computing the rainbow coloring of a graph NP-Hard, but also verifying if a given rainbow coloring is proper is NP-Complete. If we reduce the complexity of the problem by focusing on only a given pair of vertices, instead of every pair, then also the problem remains NP-Complete. Both of these results, stated below, were proved by [7].

**Theorem 12 ( [7])** *Given an edge-colored graph G and a pair of vertices s and t, deciding if s and t are connected by a rainbow path is NP-Complete.*

**Theorem 13 ( [7])** *Given an edge-colored graph G, checking if the edge coloring makes G rainbow connected is NP-Complete.*

They also note that when the number of colors is constant, the problem of checking if the given edge coloring makes the graph rainbow connected becomes easy.

## 4.2 Hardness of parameterized variants

Two algorithms for subset rainbow k-coloring, parameterized by $|S|$ where $S$ is the subset in question, were given by [12]. The running times were $2^{|S|}n^{\mathcal{O}(1)}$ for $k = 2$ and $|S|^{\mathcal{O}(|S|)}n^{\mathcal{O}(1)}$ for every fixed $k$. They also conjectured the existence of an algorithm for subset rainbow k-coloring running in time $2^{\mathcal{O}(|S|)}n^{\mathcal{O}(1)}$ for every fixed $k$. This was proved by [1] by coming up with a parameterized algorithm for the same.

Another of their conjectures for the steiner rainbow k-coloring was proved by [1] which is stated below.

**Theorem 14 ( [1])** *For $k \geq 3$, k-steiner rainbow coloring does not admit an algorithm running in time $2^{o(|S|^2)n^{\mathcal{O}(1)}}$, unless Exponential Time Hypothesis fails.*

However, since the original conjecture of was for $k \geq 2$, it remains partially solved.

# Chapter 5

# Results

In Section 4.1, we mentioned results about the hardness of rainbow coloring problem. In the RAM (Random Access Memory) model of computation, verifying if a given edge coloring of a graph makes it rainbow connected is NP-Complete. Given a graph and a pair of vertices, checking if the pair is connected by a rainbow path is NP-Complete as well, in the RAM model. A natural extension is to try to solve the above two problems in streaming. Verifying rainbow coloring in the streaming model of computation is an open problem. However, the problem of verifying a vertex coloring or an edge coloring in streaming also remains open. This chapter focuses on verifying if a given vertex coloring of a graph is valid in the streaming model.

## 5.1 Verifying Vertex Coloring in streaming

A coloring function $f : V(G) \to [C]$ is defined on the vertices of a graph. An edge $(u, v)$ is called monochromatic if $f(u) = f(v)$ where $u, v \in V$. We call a function $\varepsilon$-far from being a valid coloring function if at least $\varepsilon|E|$ edges in the graph are monochromatic, where $\varepsilon \in (0, 1)$.

### Problem definition:

Given any graph, its vertices along with a coloring function $f$ defined on them, arrive in a streaming fashion. We are given an assurance that there are either no conflicting edges or at least $\varepsilon|E|$ conflicting edges in the graph, for any $\varepsilon > 0$, if there are any. The objective is to decide if $f$ is $\varepsilon$-far from being a valid function or not. We call this the verification problem.

We solve the aforementioned problem for various streaming models such as vertex

arrival model with degree oracle, vertex arrival model, vertex arrival model with random order stream and the adjacency list model.

To check if an edge is conflicting, we need to be able to access the colors of both the end points of that edge. Say, $e = (u, v)$ is the edge we want to check for conflict and $u$ appears before $v$ in the stream. We are made aware of the existence of this edge only when $v$ appears in the stream. While we have access to $v$'s color $f(v)$ when we encounter it in the stream, we have lost the color of $u$, that has already streamed past. Unless, we explicitly store u and its color, we cannot verify if the edge is monochromatic or not. This is difficult, because, when $u$ appears in the stream, we are unaware of the edge $(u, v)$. Our approach to solve the verification problem, in the different streaming models, involves getting around this difficulty.

The following sections describe the streaming models and the respective algorithms to solve the verification problem.

### 5.1.1 Vertex Arrival model with degree oracle

This variant involves solving the verification problem for graphs in the vertex arrival streaming model. In this model, the vertices, along with their colors, stream in an arbitrary fashion. We assume access to a degree oracle, i.e., we can query for the degree $deg_G(u)$ of the vertex $u \in V(G)$ in the stream.

To sample and check an edge for conflict, we use the following approach. For a vertex $u$ that is currently being observed in the stream, $deg_G^-(u)$ and $deg_G^+(u)$ denote the number of neighbors of $u$ to its left (already exposed) and right (that are yet to be seen) in the stream, respectively. We identify each edge symbolically as $(u, j)$ where $u \in V$ and $j \in [deg_G^+(u)]$. So, $(u, j)$ represents the edge joining $u$ and its $j^{th}$ neighbor to its right. When $u$ arrives, we sample it, along with its color, independently. We also sample its $j^{th}$ neighbor to its right, where $j \in [deg_G^+(u)]$. Since we have access to a degree oracle, we can query for the degree $deg_G(u)$ of a vertex $u$ in the stream, and use this information to compute $deg_G^+(u)$. When the $j^{th}$ neighbor of $u$, say $v$, arrives, we check the edge $(u, v)$ for conflict.

If $\varepsilon \leq \frac{1}{|V|}$, then we store every vertex along with its color. We accurately determine the number of conflicting edges and easily verify if the coloring function $f$ is $\varepsilon$-far from being a valid function or not.

If $\varepsilon > \frac{1}{|V|}$, we use reservoir sampling to randomly sample a subset of $\widetilde{\mathcal{O}}\left(\frac{1}{\varepsilon}\right)$ vertices and their colors. For each vertex in the random sample, we sample one of its neighbor to its right in the stream, along with its color, uniformly at random. We effectively check $\widetilde{\mathcal{O}}\left(\frac{1}{\varepsilon}\right)$ edges for conflict and solve the verification problem. If none of the sampled edges are monochromatic, then $f$ is valid, else we declare $f$ to be $\varepsilon$-far to be valid.

**Theorem 15** *Given any input graph $G = (V, E)$, a coloring function $f : V(G) \to [C]$ and an input parameter $\varepsilon > 0$, there exists an algorithm that solves the verification problem in the vertex arrival streaming model with degree oracle with high probability in space $\tilde{\mathcal{O}}\left(\min\left(|V|, \frac{1}{\varepsilon}\right)\right)$.*

**Proof:**
If $\varepsilon \leq \frac{1}{|V|}$, then we store all the vertices along with their respective colors. This enables us to check every edge in the graph and determine the exact number of monochromatic edges and accurately verify if $f$ is $\varepsilon$-far from valid or not. If the number of monochromatic edges is zero, $f$ is valid. Otherwise, if the number of monochromatic edges exceeds $\varepsilon|E|$, then $f$ is $\varepsilon$-far from valid. Clearly, the space used is $\widetilde{\mathcal{O}}(|V|)$ in this case.

If $\varepsilon > \frac{1}{|V|}$, we sample $\widetilde{\mathcal{O}}\left(\frac{1}{\varepsilon}\right)$ edges, independently and uniformly at random, using reservoir sampling.
Every time we sample a vertex $u$, we also sample one of its (yet to be exposed) neighbors at random. We do this by sampling the $j^{th}$ neighbor of $u$, where $j \in [deg_G^+(u)]$ and check the colors of both the vertices for conflict. We charge every edge $e = (u, v)$ to the vertex $u$, thus ensuring that no edge is sampled more than once.

$$Pr(\texttt{an edge e is conflicting}) \geq \frac{\varepsilon|E|}{|E|} = \varepsilon$$

If we sample $t$ edges independently and uniformly at random, then,

$$Pr(\texttt{none of the t edges sampled are conflicting}) \leq (1-\varepsilon)^t \leq e^{-\varepsilon t}$$

So we sample $t = \tilde{\mathcal{O}}\left(\frac{1}{\varepsilon}\right)$ edges. If we get a monochromatic edge in one of our checks, which we get with high probability (i.e. atleast $(1 - n^{-c})$, for some constant $c > 0$) if there are atleast $\varepsilon m$ conflicting edges, then we can confirm that the coloring is $\varepsilon$-far from valid. Otherwise, we say that $f$ is valid. The space complexity of the algorithm is $\tilde{\mathcal{O}}\left(\frac{1}{\varepsilon}\right)$. □

We give a pseudocode for solving verification in vertex arrival with degree oracle model in the next page. We make use of a reservoir to store $\widetilde{\mathcal{O}}\left(\frac{1}{\varepsilon}\right)$ edges. Each entry of the reservoir $R$ stores a vertex $u$, it's color $f(u)$, a variable *count*, an integer $b$, the vertex $v$ (the $b^{th}$ neighbor of $u$) and it's color $f(v)$. Initially, the reservoir is empty. The variable *count* keeps a count of $u$'s successive neighbors seen so far in the stream. $b$ is an integer chosen uniformly at random from $[deg_G^+(u)]$.

**Input:** $G = (V, E)$, $|V| = n$ and $|E| = m$ and a coloring function $f$ on $V$ in the vertex arrival with degree oracle model
**Output:** The algorithm verifies if $f$ is $\varepsilon$-far from valid or not
If $\varepsilon \leq \frac{1}{|V|}$ , then store all the vertices and their colors and check every edge for conflict
Otherwise, initialize a reservoir $R$ of size $t$
**for** $i \leftarrow 1$ *to* $|V|$ **do**
  let $u$ be the $i^{th}$ vertex of the stream and $deg_G(u)$ , its degree
  **for** $j \leftarrow 1$ *to* $t$ **do**
    **if** $u$ *is adjacent to the* $j^{th}$ *vertex* $v$ *stored in* $R$ **then**
      increment the value of $v's$ count by $1$
      **if** *count* $b$ *(i.e.,* $u$ *is* $v's\ b^{th}$ *successive neighbor in the stream)* **then**
        store $u$ , along with its color $f(u)$ , as the $b^{th}$ neighbor of $v$ (equivalent to storing edge $(u, v)$ with the colors to check for conflict)
      **end**
    **end**
  **end**
  **if** $i \leq t$ **then**
    store $u$ and its color $f(u)$ and initialize its count to $0$
    $deg^+_G(u) \leftarrow deg_G(u) - deg^-_G(u)$
    choose $b$ uniformly at random in the range $[deg^+_G(u)]$
  **end**
  **else**
    with probability $\frac{t}{i}$ , replace an element of the reservoir $R$ , chosen uniformly at random, with $u$ and color $f(u)$
    initialize $u's$ count to $0$
    choose $b$ uniformly at random from $[deg^+_G(u)]$ (and replace the previous vertex's neighbor, if it has been stored)
  **end**
**end**
**for** $i \leftarrow 1$ *to* $t$ **do**
  let $u$ be the $i^{th}$ vertex stored in the reservoir $R$ and $v$ its stored neighbor (with their respective colors)
  check if the assigned colors to $u$ and $v$ make the edge $(u, v)$ monochromatic or not
**end**
Output $f$ is valid if none of the edges sampled are conflicting, else output that $f$ is $\varepsilon$-far from being valid.

**Algorithm 1:** Algorithm: verification in vertex arrival with degree oracle model

### 5.1.2 Vertex Arrival model

This variant solves the verification problem for graphs in the vertex arrival streaming model. In this model, the vertices, along with their colors, stream in an arbitrary fashion.

Unlike in the vertex arrival model with access to a degree oracle, where we know the degree of a vertex and can thus, exploit this information to sample edges, we don't have any advantage in the vertex arrival model. When a vertex $u$ appears in the stream, we have no knowledge of its future neighbors. Instead, we sample a subset of edges prior to the start of the algorithm and later when the vertices arrive in a stream, we sample the vertices corresponding to this subset and solve the verification problem.

If $\frac{1}{\varepsilon} \geq |V|$, then we store every vertex along with its color. We accurately determine the number of conflicting edges and easily verify if the coloring function $f$ is $\varepsilon$-far from being a valid function or not.

If $\frac{1}{\varepsilon} < |V|$, we randomly sample a subset $R$ of $\widetilde{\mathcal{O}}\left(\frac{n^2}{\varepsilon m}\right)$ pairs of vertices, before the start of the stream. Each pair of vertices is sampled independently with probability $\widetilde{\mathcal{O}}\left(\frac{1}{\varepsilon m}\right)$. We check pairs of vertices in $R$ for conflict. When the vertices start arriving in the stream, we store vertices and colors, corresponding to the first vertex $u$ in the pairs $(u, v) \in R$. When the second vertex $v$ of the pair $(u, v)$ arrives in the stream, we check its color against that of the stored vertex $u$ for conflict. If there are no monochromatic edges among the sampled edges, then $f$ is valid, else we declare $f$ to be $\varepsilon$-far from being a valid coloring function.

**Theorem 16** *Given any input graph $G = (V, E)$, a coloring function $f : V(G) \to [C]$ and an input parameter $\varepsilon > 0$, there exists an algorithm that solves the verification problem in the vertex arrival streaming model with high probability in space $\widetilde{\mathcal{O}}\left(\min\{|V|, \frac{|V|^2}{\varepsilon|E|}\}\right)$.*

**Proof:**
If $\frac{1}{\varepsilon} \geq |V|$, then we store all the vertices along with their respective colors. This enables us to check every edge in the graph and determine the exact number of monochromatic edges and accurately verify if $f$ is $\varepsilon$-far from valid or not. If the number of monochromatic edges is zero, $f$ is valid. Otherwise, if the number of monochromatic edges exceeds $\varepsilon|E|$, then $f$ is $\varepsilon$-far from valid. Clearly, the space used is $\widetilde{\mathcal{O}}(|V|)$ in this case.

If $\frac{1}{\varepsilon} < |V|$, we sample independently and uniformly at random, $\widetilde{\mathcal{O}}\left(\frac{n^2}{\varepsilon m}\right)$ vertex pairs. Let $S \subseteq V$ be the set of sampled vertices (corresponding to the first vertex of some

pair in R). When a vertex (corresponding to the second vertex of some pair in R) appears in a stream, we check if it forms a monochromatic edge. At the end of the stream, the algorithm declares the instance of the graph to be properly colored (valid) if it cannot find a monochromatic edge, else it declares the instance to be $\varepsilon$-far from being monochromatic (valid).

$$Pr\left(\texttt{an edge e is conflicting}\right) \geq \frac{\varepsilon m}{\binom{n}{2}} \geq \frac{\varepsilon m}{n^2}$$

If we sample $t$ edges overall, independently and uniformly at random, then,

$$Pr(\texttt{none of the edges sampled are conflicting}) \leq \left(1 - \frac{\varepsilon m}{n^2}\right)^t \leq e^{-\frac{\varepsilon m t}{n^2}}$$

Effectively, we sample $\widetilde{\mathcal{O}}\left(\frac{n^2}{\varepsilon m}\right)$ edges independently and uniformly at random with probability $p = \frac{10 \log n}{\varepsilon m}$. If at least one of the sampled edges we check is monochromatic, which we get with high probability (i.e. atleast $(1 - n^{-10})$) when there are atleast $\varepsilon m$ conflicting edges in the graph, then we can confirm that the coloring is $\varepsilon$-far from valid. Otherwise the coloring function is valid. □

We give a pseudocode for solving verification in vertex arrival streaming model in the next page.

**Input:** $G = (V, E)$, $|V| = n$ and $|E| = m$ and a coloring function $f$ on $V$ in the vertex arrival model
**Output:** The algorithm verifies if $f$ is $\varepsilon$-far from valid or not
If $\frac{1}{\varepsilon} \geq |V|$ , then store all the vertices and their colors and check every edge for conflict
Otherwise, sample a subset $R$ of vertex pairs $(u, v)$ with probability $\widetilde{\mathcal{O}}\left(\frac{1}{\varepsilon m}\right)$
$|R| = \widetilde{\mathcal{O}}\left(\frac{n^2}{\varepsilon m}\right)$
**for** $i \leftarrow 1$ *to* $|V|$ **do**
  let $u$ be the $i^{th}$ vertex of the stream
  **if** $u$ *is the first vertex of some pair in* $R$ **then**
    store $u$ and its color $f(u)$
  **end**
  **for** *every pair in* $R$ *where* $u$ *is the second vertex* **do**
    let $w$ be the first vertex of the concerned pair
    check the edge $(w, u)$ for conflict
  **end**
**end**
Output $f$ is valid if none of the edges sampled are conflicting, else output that $f$ is $\varepsilon$-far from being valid.

**Algorithm 2:** Algorithm: verification in vertex arrival streaming model

### 5.1.3 Vertex Arrival model with random order

This variant solves the verification problem for graphs in the vertex arrival random order streaming model. In this model, the vertices, along with their colors, stream in a random order.

We consider the subgraph $G'$ defined on only the $\varepsilon m$ monochromatic edges of the input graph $G$. We state the following lemma that guarantees the existence of either a matching or a high degree vertex in the subgraph $G'$.

**Lemma 3 ( [11])** *Let $G = (V, E)$ be a graph and $f : V(G) \to [C]$ be a coloring function such that at least $\varepsilon$ fraction of the edges $E(G)$ are known to be monochromatic. Then, there exists either a matching of size at least $\sqrt{\varepsilon m}$ or a vertex of degree at least $\sqrt{\varepsilon m}$ in the subgraph $G'$ defined on the monochromatic edges of $G$.*

We sample a subset of vertices arriving in the stream with probability $p = \frac{10 \log n}{\sqrt{m}}$ independently and uniformly at random. Let $S \subseteq V$ be the subset of sampled vertices. When these vertices arrive in the stream, we store them along with their colors. When a vertex appears in a stream, we check if it forms a monochromatic edge with one of the stored vertices in $S$. If a monochromatic edge is not found, $f$ is valid. Otherwise, $f$ is declared to be $\varepsilon$-far from being a valid coloring function.

**Theorem 17** *Given any input graph $G = (V, E)$, a coloring function $f : V(G) \to [C]$ and an input parameter $\varepsilon > 0$, there exists an algorithm that solves the verification problem in the vertex arrival random order streaming model with high probability in space $\widetilde{\mathcal{O}}\left(\frac{|V|}{\sqrt{\varepsilon |E|}}\right)$.*

**Proof:**
We use Lemma 3 to prove our results.

**Case 1: There exists a matching of size atleast $\sqrt{\varepsilon m}$**

There exists a matching of size at least $\sqrt{\varepsilon m}$, all the matched edges being monochromatic. Any matched edge $(u, v)$ will be detected as monochromatic only if the vertex $u$ has been sampled. We sample a subset $S \subseteq V$ of vertices and their colors with probability $p = \frac{10 \log n}{\sqrt{m}}$, where $|S| = \widetilde{\mathcal{O}}\left(\frac{n}{\sqrt{\varepsilon m}}\right)$.

$$\begin{aligned} &Pr(\texttt{atleast one of the matched edges is checked}) \\ &= 1 - Pr(\texttt{none of the matched edges are checked}) \\ &= 1 - (1-p)^{\sqrt{\varepsilon m}} \geq 1 - e^{-p\sqrt{\varepsilon m}} \end{aligned}$$

$$Pr(\texttt{a conflicting edge (u, v) is checked}) = Pr(\texttt{u is sampled}) = \frac{|S|}{n}$$

$$\begin{aligned} Pr(\texttt{atleast one of the matched edges is checked}) &= 1 - e^{-p\sqrt{\varepsilon m}} \\ &= 1 - e^{-\frac{|S|}{n}\sqrt{\varepsilon m}} \end{aligned}$$

We can verify if the coloring function is $\varepsilon$-far from being valid, if any of the $\sqrt{\varepsilon m}$ matched edges is detected. Otherwise, we declare it to be a valid coloring function.

**Case 2: There exists a high degree vertex of degree atleast $\sqrt{\varepsilon m}$**

There exists a vertex of degree at least $\sqrt{\varepsilon m}$. Most of the monochromatic edges, in this case, are incident on such high degree vertices. In order to detect these edges, we can store either the high degree vertices or one of its neighbours. But, if these high degree vertices appear at the beginning of the stream and we fail to sample them, then we may not be able to detect a monochromatic edge. This is the reason we assume a random ordering of vertices arriving in the stream. A random order stream ensures that the high degree vertex appears after a constant fraction of its neighbors. If we can sample some of its neighbors that appear before it in the stream, we can check the conflicting edges.

Let the random variable $X$ denote the number of neighbors that appear before the high degree vertex appears in the random order stream. Then, $X = \sum_{1 \le i \le \sqrt{\varepsilon m}} X_i$ and $\mathbb{E}[X] = \frac{\sqrt{\varepsilon m}}{2}$. Let $E$ be the event that the high degree vertex appears after a constant fraction $\frac{1}{10}$ of its neighbors.

$$Pr(E) = Pr(X > \frac{1}{10}\sqrt{\varepsilon m}) = 1 - Pr(X \le \frac{1}{10}\sqrt{\varepsilon m})$$

Using Chernoff Bound, we get,

$$Pr(X \le \frac{1}{10}\sqrt{\varepsilon m}) = Pr(X \le (1 - \frac{4}{5})\sqrt{\varepsilon m}) \le e^{-\frac{4}{25}\sqrt{\varepsilon m}}$$

Thus, assuming random order of vertices in the stream, at least $\frac{1}{10}\sqrt{\varepsilon m}$ neighbours of $v$ should appear before $v$ in the stream with probability at least $(1 - e^{-\frac{4}{25}\sqrt{\varepsilon m}})$.

$$\begin{aligned} Pr(\texttt{a conflicting edge is detected}) &\ge Pr(E) Pr(\texttt{a conflicting edge is detected}|E) \\ &= \frac{1}{c}\left(1 - (1-p)^{\frac{\sqrt{\varepsilon m}}{10}}\right) \ge \frac{1}{c}\left(1 - e^{-p\frac{\sqrt{\varepsilon m}}{10}}\right) \ge \frac{1}{c}\left(1 - n^{-c}\right) \end{aligned}$$

when the probability of sampling is $p = \frac{10 \log n}{\sqrt{\varepsilon m}}$.

Since we sample every vertex with probability $\frac{10 \log n}{\sqrt{\varepsilon m}}$, with high probability at least $(1 - 1/n^2)$ of its neighbors will be stored. Thus, we are able to check a conflicting edge with high probability. □

### 5.1.4 Adjacency List model

This variant solves the verification problem for graphs in the adjacency list streaming model. In this model, the vertices, along with their colors, stream in an arbitrary order. In addition, the neighborhood of the vertex $v$, $N_G(v)$, is also exposed along with it in the stream.

Since, the neighborhood of the vertex $v$, $N_G(v)$, is also exposed along with it in the stream, we see every edge twice in the stream. For example, if the stream consists of vertices in the order $v_1, v_2, v_3$ and $v_4$, where $N_G(v_1) = \{v_2, v_3\}$ and $N_G(v_4) = \{v_3\}$, then the stream can be visualized as edges coming in a stream: $(v_1, v_2), (v_1, v_3), (v_2, v_1)$, $(v_3, v_1), (v_3, v_4), (v_4, v_3)$. Each edge appears twice, once in the adjacency list of each of its endpoints.

The algorithm for this model is similar to that of the vertex arrival model with degree oracle.

If $\varepsilon \leq \frac{1}{|V|}$, then we store every vertex along with its color. We accurately determine the number of conflicting edges and easily verify if the coloring function $f$ is $\varepsilon$-far from being a valid function or not.

If $\varepsilon > \frac{1}{|V|}$, we use reservoir sampling to randomly sample a subset of $\widetilde{\mathcal{O}}\left(\frac{1}{\varepsilon}\right)$ vertices and their colors. For each vertex in the random sample, we sample one of its neighbors in its adjacency list, along with its color, uniformly at random. We effectively check $\widetilde{\mathcal{O}}\left(\frac{1}{\varepsilon}\right)$ edges for conflict and solve the verification problem. If none of the sampled edges are monochromatic, then $f$ is valid, else we declare $f$ to be $\varepsilon$-far to be valid.

**Theorem 18** *Given any input graph $G = (V, E)$, a coloring function $f : V(G) \to [C]$ and an input parameter $\varepsilon > 0$, there exists an algorithm that solves the verification problem in the adjacency list streaming model with high probability in space $\tilde{\mathcal{O}}\left(\min\left(|V|, \frac{1}{\varepsilon}\right)\right)$.*

**Proof:**
If $\varepsilon \leq \frac{1}{|V|}$, then we store all the vertices along with their respective colors. This enables us to check every edge in the graph and determine the exact number of monochromatic edges and accurately verify if $f$ is $\varepsilon$-far from valid or not. If the number of monochromatic edges is zero, $f$ is valid. Otherwise, if the number of monochromatic edges exceeds $\varepsilon|E|$, then $f$ is $\varepsilon$-far from valid. Clearly, the space used is $\widetilde{\mathcal{O}}\left(|V|\right)$ in this case.

If $\varepsilon > \frac{1}{|V|}$, we sample $\widetilde{\mathcal{O}}\left(\frac{1}{\varepsilon}\right)$ edges, independently and uniformly at random, using reservoir sampling.
Every time we sample a vertex $u$, we also sample one of its neighbors at random from

its adjacency list. We do this by sampling the $j^{th}$ neighbor of $u$, where $j \in [deg_G(u)]$ and check the colors of both the vertices for conflict. However, an edge $(u, v)$ may be sampled twice. Since an edge occurs only twice in the stream, when we sample $\tilde{\mathcal{O}}\left(\frac{1}{\varepsilon}\right)$ edges, at least half of the sampled edges will be distinct.

$$Pr(\texttt{an edge e is conflicting}) \geq \frac{2\varepsilon|E|}{2|E|} = \varepsilon$$

If we sample $t$ edges independently and uniformly at random, then,

$$Pr(\texttt{none of the t edges sampled are conflicting}) \leq (1-\varepsilon)^t \leq e^{-\varepsilon t}$$

So we sample $t = \tilde{\mathcal{O}}\left(\frac{1}{\varepsilon}\right)$ edges. If we get a monochromatic edge in one of our checks, which we get with high probability (i.e. atleast $(1 - n^{-c})$, for some constant $c > 0$) if there are atleast $\varepsilon m$ conflicting edges, then we can confirm that the coloring is $\varepsilon$-far from valid. Otherwise, we say that $f$ is valid. The space complexity of the algorithm is $\tilde{\mathcal{O}}\left(\frac{1}{\varepsilon}\right)$. □

## 5.2 Lower bounds for verification

We show a tight lower bound of $\Omega\left(\frac{1}{\varepsilon}\right)$ for the verification problem in the vertex arrival model with degree oracle. For the verification problem in the vertex arrival model we show a tight lower bound of $\Omega\left(\frac{|V|}{\sqrt{|E|}}\right)$. These bounds are proved using reductions from the *INDEX* problem in the one-way communication complexity model [13] to the verification problem in graphs in the vertex arrival model. In the INDEX problem, Alice gets an $M$-bit string $X \in \{0,1\}^M$ and Bob gets an integer $j \in [M]$. Both Alice and Bob are unaware of each other's input. Alice can send a message to Bob and Bob has to output $X_j$. It is well known that the lower bound on the message length for the INDEX problem in the one-way communication complexity model is $\Omega(M)$.

### 5.2.1 Lower bound for verification in vertex arrival model with degree oracle

**Theorem 19** *The lower bound for the verification problem in the vertex arrival streaming model with degree oracle is* $\Omega\left(\frac{1}{\varepsilon}\right)$.

**Proof:**
We show the lower bound using a reduction from the INDEX problem in one-way communication complexity to the verification problem in graphs in the vertex arrival model with access to degree oracle. The reduction works as follows: For each input bit $X_i$, Alice constructs a bipartite graph with bipartition $(L_i, R_i)$, where $L_i$ and $R_i$ are independent sets of size $k$. For any two sets $L_s$ and $R_t$, Alice does not add any edge between vertices in $L_s$ and $R_t$. If $X_i$ equals 1, then vertices in $L_i$ are colored with color $C_1$, else they are colored with color $C_0$. At the end of the stream, Alice sends this memory state to Bob. Bob colors all vertices in $R_i$ for $1 \leq i \neq j \leq M$ with color $C_2$. Vertices in $R_j$ are colored with $C_1$. Moreover, for any index $i \in [M]$, Bob adds all edges between $L_i$ and $R_i$. Access to a degree oracle does not provide any advantage to any protocol on this instance. The only difference is in the construction of the graph. Now, if $X_j = 0$, then there are no monochromatic edges in the graph. However, if $X_j = 1$, there are $k^2$ monochromatic edges. The number of vertices in the graph is $n = Mk$ and the number of edges is $m = Mk^2$. The number of conflicting edges in the graph is $k^2 = \varepsilon m = \varepsilon M k^2$. Therefore, for $M = \frac{1}{\varepsilon}$, distinguishing whether the graph is properly-colored (valid) or at least $\varepsilon$-fraction of edges are monochromatic ($\varepsilon$-far from being valid) requires $\Omega\left(\frac{1}{\varepsilon}\right)$ space. □

### 5.2.2 Lower bound for verification in vertex arrival model

**Theorem 20** *The space lower bound for the verification problem in the vertex arrival model is* $\Omega\left(\frac{|V|}{\sqrt{|E|}}\right)$.

**Proof:**
We show the lower bound using a reduction from the INDEX problem in one-way communication complexity to the verification problem in graphs in the vertex arrival model. For each input bit $X_i$, Alice constructs a bipartite graph with bipartition $(L_i, R_i)$, where $L_i$ and $R_i$ are independent sets of size $k$. For any two sets $L_s$ and $R_t$, Alice does not add any edge between vertices in $L_s$ and $R_t$. If $X_i$ equals 1, then vertices in $L_i$ are colored with color $C_1$, else they are colored with color $C_0$. At the end of the stream, Alice sends this memory state to Bob. Bob colors all vertices in $R_i$ for $1 \leq i \leq M$ with color $C_1$. Moreover, for index $j$, he adds all the edges between $L_j$ and $R_j$ to make it a complete bipartite graph. Now, if $X_j = 0$, then there are no monochromatic edges. However, if $X_j = 1$, there are $k^2$ monochromatic edges. We have the number of vertices in the graph $n = Mk$ and the number of edges $m = k^2$. Therefore, for $M = \frac{n}{\sqrt{m}}$ or $\frac{|V|}{\sqrt{|E|}}$, distinguishing whether the graph is properly-colored (valid) or all of its edges are monochromatic ($\varepsilon$-far from being valid) requires $\Omega\left(\frac{|V|}{\sqrt{|E|}}\right)$ space. □

# Chapter 6

# Future Work and Conclusion

Verification of most graph coloring problems, except rainbow coloring, is an easy problem in the RAM model of computation. However, in the streaming model, none of the graph coloring problems admit an easy solution. Given any edge colored graph, verifying if it is properly edge colored and/or rainbow colored still remains unsolved. In fact, for a given edge colored graph, checking if a given pair of vertices in it are connected by a rainbow path remains open as well.

The lower bound for solving the verification problem in vertex arrival with random order stream is also an open problem.

# Bibliography


[1] Agrawal, A.: Fine-grained complexity of rainbow coloring and its variants. In: MFCS (2017)

[2] Ananth, P., Nasre, M., Sarpatwar, K.K.: Rainbow connectivity: Hardness and tractability (2011)

[3] Assadi, S., Chen, Y., Khanna, S.: Sublinear algorithms for ($\Delta$ + 1) vertex coloring. In: Chan, T.M. (ed.) Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019. pp. 767–786. SIAM (2019), https://doi.org/10.1137/1.9781611975482.48

[4] Behnezhad, S., Derakhshan, M., Hajiaghayi, M., Knittel, M., Saleh, H.: Streaming and massively parallel algorithms for edge coloring. In: Bender, M.A., Svensson, O., Herman, G. (eds.) 27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany. LIPIcs, vol. 144, pp. 15:1–15:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019), https://doi.org/10.4230/LIPIcs.ESA.2019.15

[5] Bera, S.K., Chakrabarti, A., Ghosh, P.: Graph coloring via degeneracy in streaming and other space-conscious models. CoRR abs/1905.00566 (2019), http://arxiv.org/abs/1905.00566

[6] Caro, Y., Lev, A., Roditty, Y., Tuza, Z., Yuster, R.: On rainbow connection. Electr. J. Comb. 15(1) (2008), http://www.combinatorics.org/Volume_15/Abstracts/v15i1r57.html

[7] Chakraborty, S., Fischer, E., Matsliah, A., Yuster, R.: Hardness and algorithms for rainbow connection. Journal of Combinatorial Optimization 21(3), 330–347 (Apr 2011)

[8] Chandran, L.S., Das, A., Issac, D., van Leeuwen, E.J.: Algorithms and bounds for very strong rainbow coloring (2017)

[9] Chartrand, G., Johns, G., McKeon, K., Zhang, P.: Rainbow connection in graphs. Mathematica Bohemica 133 (01 2008)

[10] Impagliazzo, R., Paturi, R.: The complexity of k-sat. In: Proceedings of the Fourteenth Annual IEEE Conference on Computational Complexity. p. 237. COCO '99, IEEE Computer Society, USA (1999)

[11] Jukna, S.: Extremal Combinatorics - With Applications in Computer Science. Texts in Theoretical Computer Science. An EATCS Series, Springer (2011), https://doi.org/10.1007/978-3-642-17364-6

[12] Łukasz Kowalik, Lauri, J., Socała, A.: On the fine-grained complexity of rainbow coloring (2016)

[13] Kushilevitz, E., Nisan, N.: Communication complexity. Cambridge University Press (1997)

[14] Le, V., Tuza, Z.: Finding Optimal Rainbow Connection is Hard. Preprints aus dem Institut für Informatik / CS, Inst. für Informatik (2009), https://books.google.co.in/books?id=0ErVPgAACAAJ

[15] Misra, J., Gries, D.: A constructive proof of vizing's theorem. Inf. Process. Lett. 41(3), 131–133 (1992), https://doi.org/10.1016/0020-0190(92)90041-S

[16] Vizing, V.G.: On an estimate of the chromatic class of a p-graph (1964)