**CS120: Intro. to Algorithms and their Limitations** Prof. Salil Vadhan

# Lecture 13: Graph Coloring

*Harvard SEAS - Fall 2021* *Oct. 19, 2021*

## 1 Announcements

Recommended Reading:

- Lewis–Zax Ch. 18
- Roughgarden III Sec. 13.1
- PS5 due tomorrow
- Salil OH Thursday after class

## 2 Graph Coloring

**Motivating Problem:** Register allocation.

Goal: more efficiently simulate RAM programs on a CPU with a fixed number of registers by storing "temporary variables" directly in registers (rather to memory as in Active Learning Exercise 3)

Difficulty: need to make sure each register is only handling one temporary variable at a time

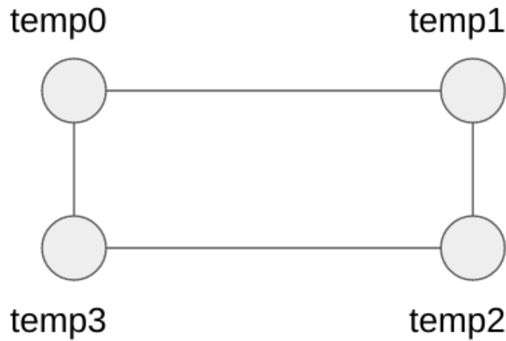
Example: Suppose

- $\texttt{temp}_0$ is only relevant in code blocks 0 and 1
- $\texttt{temp}_1$ is only relevant in code blocks 1 and 2
- $\texttt{temp}_2$ is only relevant in code blocks 2 and 3,
- $\texttt{temp}_3$ is only relevant in code blocks 3 and 0

How can we use just two registers to store $\texttt{temp}_0$, $\texttt{temp}_1$, $\texttt{temp}_2$, and $\texttt{temp}_3$? Use register $A$ for $\texttt{temp}_0$ in code blocks 0 and 1 and $\texttt{temp}_1$ in code blocks 2 and 3. Use register $B$ for $\texttt{temp}_1$ in code blocks 1 and 2 and $\texttt{temp}_3$ in code blocks 3 and 0.

**Q:** How can we model this problem graph-theoretically?

Define a conflict graph (aka the "register interference graph"):




**Definition 2.1.** For an undirected graph $G = (V, E)$, a (proper) *$k$-coloring* of $G$ is a mapping $f : V \to [k]$ such that for all edges $\{u, v\} \in E$, we have $f(u) \neq f(v)$

An improper coloring allows us to assign the same color to vertices that share an edge, but we will work with proper colorings unless explicitly stated.
**Example:**

| | |
|---|---|
| **Input** | **:** A graph $G = (V, E)$ and a number $k$ |
| **Output** | **:** A $k$-coloring of $G$, or $\perp$ if none exists |

**Computational Problem** Graph Coloring

Alternatively, we are given a graph $G$ and we wish to find a proper coloring using as *few* colors as possible. What problem is this the opposite of?

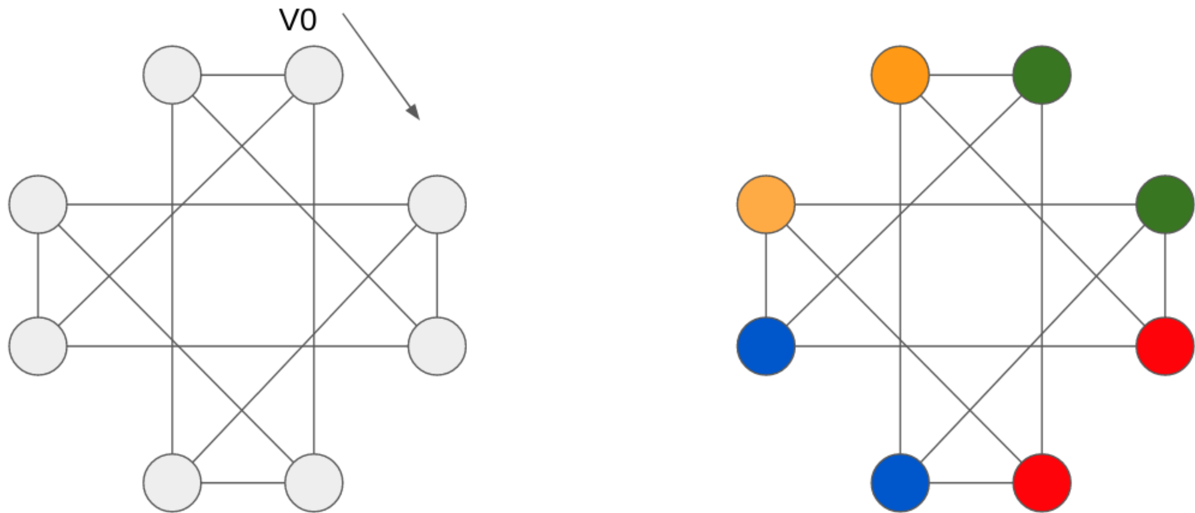
Here we want to partition the graph into *disconnected* components - partition the graph into as few subsets as possible so that there are no edges inside each subset. The connected components problems asks as to partition the graph into as many subsets as possible such that there are no edges between subsets.

# 3 Greedy Coloring

A natural first attempt at graph coloring is to use a *greedy* strategy:

```
1 GreedyColoring(G)
  Input   : A graph G = (V, E)
  Output  : A coloring f of G using "few" colors
2 Fix an arbitrary ordering v_0, v_1, v_2, ..., v_{n-1} of V;
3 foreach i = 0 to n - 1 do
4   | f(v_i) = min {c ∈ ℕ : c ≠ f(v_j)  ∀j < i s.t. {v_i, v_j} ∈ E}.
5 return f
```

**Example:**




In general, a *greedy* algorithm is one that makes a sequence of myopic decisions (above, the color of a vertex $v$), without regard to what choices will need to be made in the future.

By inspection, `GreedyColoring`$(G)$ can be implemented in time $O(n+m)$ and always outputs a proper coloring of $G$. What can we prove how many colors it uses?

**Theorem 3.1.** *When run on a graph $G = (V, E)$, `GreedyColoring`$(G)$ will use at most $d_{max} + 1$ colors, where $d_{max} = \max\{d(v) : v \in V\}$.*

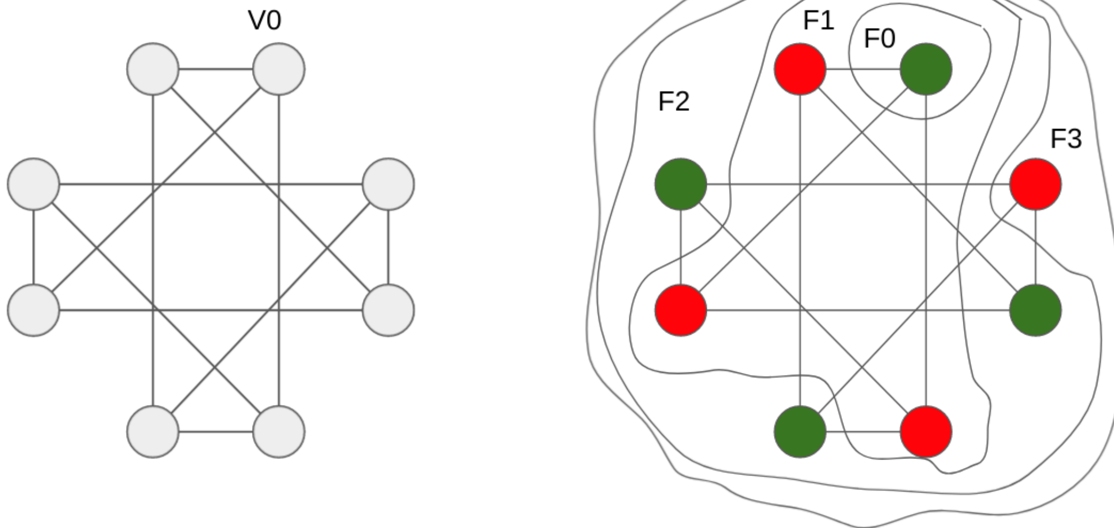
*Proof.* The set $\{c \in \mathbb{N} : c \neq f(v_j) \quad \forall j < i \text{ s.t. } \{v_i, v_j\} \in E\}$ of size at most $d(v_j) \leq d_{max}$, so cannot include all of the colors $0, 1, 2, \ldots, d_{max}$. Thus when we assign $f(v_i)$ to be the minimum element of the set, we will have $f(v_i) \in [d_{max} + 1]$. □

Note that this is an algorithmic proof of a pure graph theory fact: every graph is $(d_{max} + 1)$-colorable. However, this bound of $d_{max} + 1$ can be much larger than the number of colors actually needed to color $G$, but this turns out to be tight for greedy coloring in an arbitrary vertex order, even on 2-colorable graphs.

However, the performance of greedy algorithms is very sensitive to the order in which decisions are made, and often we can achieve much better performance by picking a careful ordering. For example, we can process the vertices in *BFS order*:

1 `BFSColoring`$(G)$
  **Input** : A connected graph $G = (V, E)$
  **Output** : A coloring $f$ of $G$ using "few" colors
2 Fix an arbitrary start vertex $v_0 \in V$;
3 Start breadth-first search from $v_0$ to obtain a vertex order $v_1, v_2, \ldots, v_{n-1}$;
4 **foreach** $i = 0$ *to* $n - 1$ **do**
5   $f(v_i) = \min\{c \in \mathbb{N} : c \neq f(v_j) \quad \forall j < i \text{ s.t. } \{v_i, v_j\} \in E\}$.
6 **return** $f$

**Example:**




**Theorem 3.2.** *If $G$ is a connected 2-colorable graph, then* `BFSColoring` *$(G)$ will color $G$ using 2 colors.*

*Proof.* Let$f^*$ be a 2-coloring of $G$. We may assume that $f^*(v_0) = 0$ without loss of generality (why?). Let $f$ be the coloring of $G$ found by `BFSColoring(`$G$`)`. We argue by (strong) induction on $i$ that $f(v_i) = f^*(v_i)$ for $i = 0, \ldots, n-1$.

For $i = 0$, we observe that `BFSColoring(`$G$`)` sets $f(v_0) = 0$. Now for $i > 0$, we will argue that $f^*$ satisfies the same rule used to construct $f$, namely:

$$f^*(v_i) = \min \{c \in \mathbb{N} : c \neq f^*(v_j) \quad \forall j < i \text{ s.t. } \{v_i, v_j\} \in E\}. \tag{1}$$

In other words, the value of $f^*$ at $v_i$ is "forced" by its values at the previously assigned vertices $v_j$. Since $f^*$ is a valid 2-coloring, the value $c = f^*(v_i)$ satisfies the condition $c \neq f^*(v_j)$ for all $j < i$ such that $\{v_i, v_j\} \in E$ automatically holds. If $f^*(v_i) = 0$, then it is certainly the minimum value of $c$ satisfying this condition. If $f^*(v_i) = 1$, we note that that by the definition of BFS, there is a previous vertex $v_j$ (with $j < i$) with an edge to $v_i$. Since $f^*$ is a valid 2-coloring, we must have $f^*(v_j) = 0$. So $c = 0$, does not satisfy the condition in Equation (1), and hence $c = 1$ must be the minimum value satisfying the condition.

By the definition of `BFSColoring(`$G$`)`, we have

$$f(v_i) = \min \{c \in \mathbb{N} : c \neq f(v_j) \quad \forall j < i \text{ s.t. } \{v_i, v_j\} \in E\} \tag{2}$$

By our (strong) induction hypothesis, the right-hand sides of (1) and (2) are equal, and thus $f(v_i) = f^*(v_i)$. □

**Corollary 3.3.** *Graph 2-Coloring can be solved in time $O(n + m)$.*

*Proof.* We can partition $G$ into connected components in time $O(n+m)$. Then, for each connected component we can use BFSColoring on each component, which takes total time $O(n + m)$. □