The vertex-coloring problem seeks to assign a label (aka color) to each vertex of a graph such that no edge links any two vertices of the same color.

- Trivial solution: assign each vertex a different color.
- However, goal is usually to use as few colors as possible.

Applications of the vertex-coloring problem:

- Apart from working at National Geographic, when might you encounter a vertex-coloring problem?
- Vertex-coloring problems arise in scheduling problems, where access to shared resources must be coordinated.
- Example: register allocation by compilers
  - Variables are used for fixed timespan (after initialization, before final use).
  - Two variables with intersecting lifespans can’t be put in the same register.
  - We can build a graph with variables assigned to vertices and edges drawn between vertices if the variables’ lifespan intersects.
  - Color the graph, and assign variables to the same register if their vertices have the same color.
Vertex-coloring problem special case: two colors

- A bipartite graph is an undirected graph whose vertices can be divided into disjoint sets $U$ and $V$ such that every edge connects a vertex in $U$ to one in $V$.
- Bipartite graphs arise in matching problems: matching workers to jobs, matching kidney donors with recipients, finding heterosexual mates.
- If we can color a graph’s vertices using just two colors, then we have a bipartite graph.
- Problem: given a graph, find its two-coloring or report that a two-coloring is not possible.

Bipartite graphs

Lemma. Let $G$ be a connected graph, and let $L_0,\ldots,L_q$ be the layers produced by BFS starting at node $x$. Exactly one of the following holds.

(i) No edge of $G$ joins two nodes of the same layer, and $G$ is bipartite.
(ii) An edge of $G$ joins two nodes of the same layer, and $G$ contains an odd-length cycle (and hence is not bipartite).

Testing bipartiteness

Many graph problems become:

- Easier if the underlying graph is bipartite (matching).
- Tractable if the underlying graph is bipartite (independent set).

Before attempting to design an algorithm, we need to understand structure of bipartite graphs.
**Breadth-first search**

**Property.** Let $T$ be a BFS tree of $G = (V, E)$, and let $(x, y)$ be an edge of $G$. Then, the level of $x$ and $y$ differ by at most 1.

**Bipartite graphs**

**Lemma.** Let $G$ be a connected graph, and let $L_0, ..., L_k$ be the layers produced by BFS starting at node $s$. Exactly one of the following holds.

(i) No edge of $G$ joins two nodes of the same layer, and $G$ is bipartite.

(ii) An edge of $G$ joins two nodes of the same layer, and $G$ contains an odd-length cycle (and hence is not bipartite).

**Pf.** (i)
- Suppose no edge joins nodes in adjacent layers.
- By previous lemma, this implies all edges join nodes on same level.
- Bipartition: red = nodes on odd levels, blue = nodes on even levels.

**Corollary.** A graph $G$ is bipartite iff it contain no odd length cycle.
Two-coloring algorithm

- Suppose there are two colors: blue and red.
- Color the first vertex blue.
- Do a breadth-first traversal. For each newly-discovered node, color it the opposite of the parent (i.e., red if parent is blue).
- If the child node has already been discovered, check if the colors are the same as the parent. If so, then the graph isn’t bipartite.
- If the traversal completes without any conflicting colors, then the graph is bipartite.

Two-coloring algorithm example 1

Two-coloring algorithm example 2

Exercise: Check for bipartness
Global Optimization

- Consider the following the following IL:

```plaintext
x := 3
if b > 0 goto L1
y := z + w
goto L2
L1:
y := x
L2:
a := 2 * x
if a < b goto L1
ret y
```

Register Interference Graph

- Use liveness analysis to compute a register interference graph (RIG)

  - Each variable is a node in the RIG
  - An edge exists between two nodes (variables) if:
    - at ANY point in program, both variables are live
  - Directly connected nodes are variables that cannot share a register

RIG Example

Graph Coloring

- A coloring of a graph is a assignment of colors to nodes:
  - such that node that share an edge have different colors

- A k-coloring is a coloring that uses at most k different colors

- A k-colorable graph is a graph for which there exists at least one k-coloring
Register Allocation via Coloring

- A \( k \)-coloring of a RIG is a valid register allocation for \( k \) registers:
  - Each color is a register
  - Variables with the same color are never live at the same time

- Graph coloring is a hard problem (NP-hard)
  - Have to use heuristics

Our Heuristic

- Start with full RIG
- While graph is not empty:
  - Select a node with minimum number of edges
  - Remove node from graph, place on stack
- While stack is not empty:
  - Pop node from stack, put back in graph
  - Add back any edges to other nodes in graph
  - Pick a color for the node that doesn’t match any neighbor
    - Pick a new color if necessary

Example Coloring

Graph:
- a
- b
- w
- x
- y
- z

Stack:
- b
- w
- x
- z
- a
- y
Example Coloring

Graph:

Stack:

Example Coloring

Graph:

Stack:

Example Coloring

Graph:

Stack:

Example Coloring

Graph:

Stack: