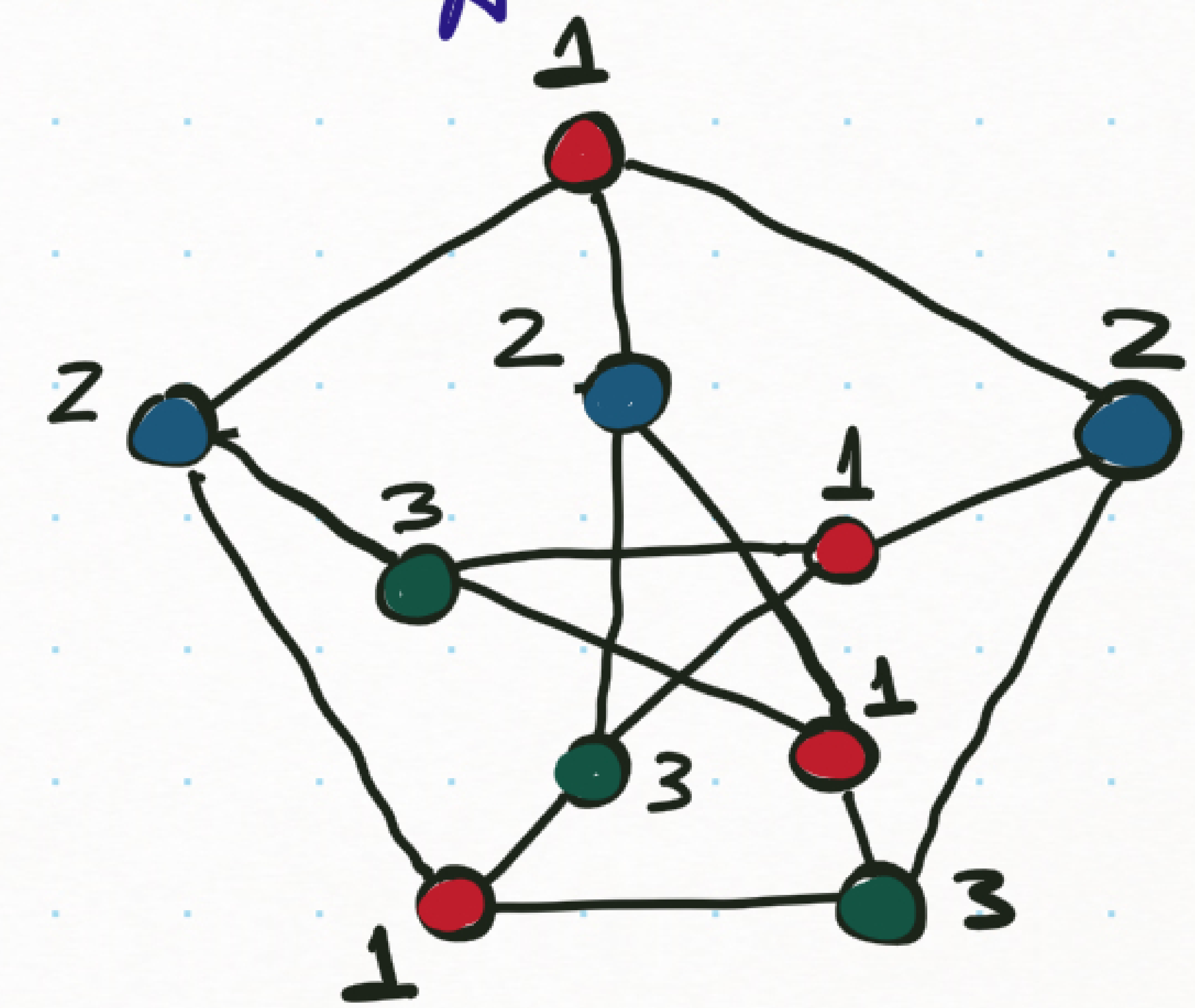
MATH 380

Hemanshu Kaul

kaul@iit.edu

Proper coloring

We color vertices in such a way that any pair of vertices with an edge between them must receive different colors.

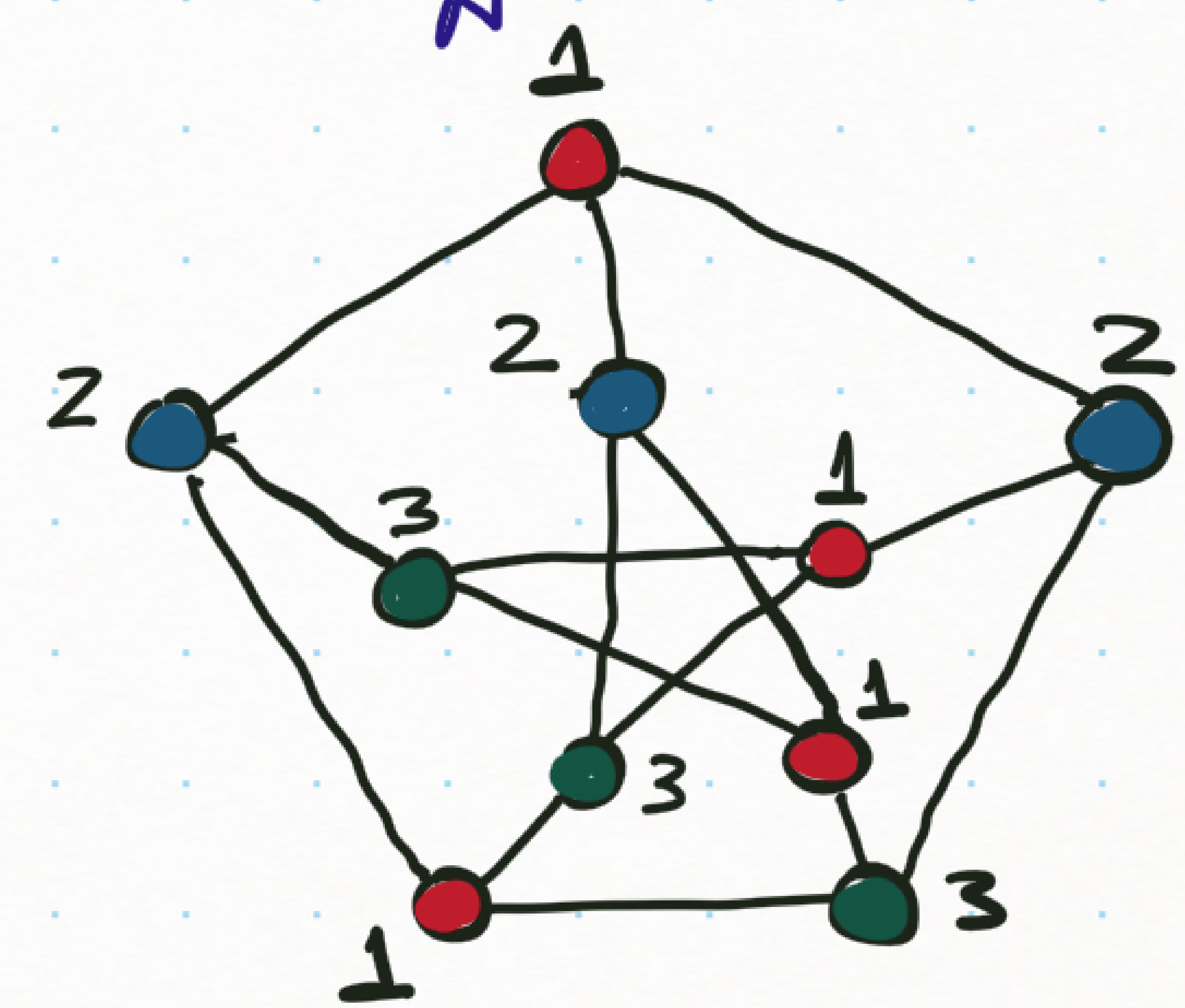


Vertices receiving the same color have no edges in between them (conflict-free allocation of resource)

We are partitioning the vertices into color classes, each are independent sets (no edges within).

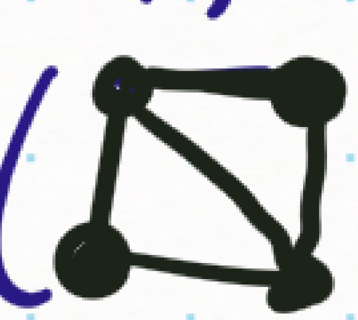
Proper coloring

We color vertices in such a way that any pair of vertices with an edge between them must receive different colors.



Vertices receiving the same color have no edges in between them (conflict-free allocation of resource)

We are partitioning the vertices into color classes, each are independent sets (no edges within).

The least number of colors need for a graph G, is called the chromatic number of G, $\chi(G)$. e.g. $\chi($  $) = 3$

Chromatic number as an optimization problem:

Input G with $V(G) = \{v_1, \dots, v_n\}$, $E(G)$.

Available colors $C = \{1, 2, \dots, n\}$

Chromatic number as an optimization problem:

Input G with $V(G) = \{v_1, \dots, v_n\}$, $E(G)$.

Available colors $C = \{1, 2, \dots, n\}$

Let $x_{ic} = \begin{cases} 1 & \text{if vertex } v_i \text{ is colored with } c \in C \\ 0 & \text{otherwise} \end{cases}$ for each $v_i \in V(G)$ and $c \in C$.

Let $y_c = \begin{cases} 1 & \text{if color } c \text{ is used} \\ 0 & \text{otherwise} \end{cases}$ for each $c \in C$.

Chromatic number as an optimization problem:

Input G with $V(G) = \{v_1, \ldots, v_n\}$, $E(G)$.

Available colors $C = \{1, 2, \ldots, n\}$

Let $x_{ic} = \begin{cases} 1 & \text{if vertex } v_i \text{ is colored with } c \in C \\ 0 & \text{otherwise} \end{cases}$ for each $v_i \in V(G)$ and $c \in C$.

Let $y_c = \begin{cases} 1 & \text{if color } c \text{ is used} \\ 0 & \text{otherwise} \end{cases}$ for each $c \in C$.

min $\sum_{c=1}^{n} y_c$ $\longleftarrow$?

s.t.

Chromatic number as an optimization problem:

Input G with $V(G) = \{v_1, \dots, v_n\}$, $E(G)$.

Available colors $C = \{1, 2, \dots, n\}$

Let $x_{ic} = \begin{cases} 1 & \text{if vertex } v_i \text{ is colored with } c \in C \\ 0 & \text{otherwise} \end{cases}$ for each $v_i \in V(G)$ and $c \in C$

Let $y_c = \begin{cases} 1 & \text{if color } c \text{ is used} \\ 0 & \text{otherwise} \end{cases}$ for each $c \in C$.

$$\min \sum_{c=1}^{n} y_c$$ [minimize total # colors used]

$$\text{s.t.} \quad \sum_{c=1}^{n} x_{ic} = 1 \quad \forall v_i \in V(G)$$ $\leftarrow$?

Chromatic number as an optimization problem:

Input G with $V(G) = \{v_1, \dots, v_n\}$, $E(G)$.

Available colors $C = \{1, 2, \dots, n\}$

Let $x_{ic} = \begin{cases} 1 & \text{if vertex } v_i \text{ is colored with } c \in C \\ 0 & \text{otherwise} \end{cases}$ for each $v_i \in V(G)$ and $c \in C$

Let $y_c = \begin{cases} 1 & \text{if color } c \text{ is used} \\ 0 & \text{otherwise} \end{cases}$ for each $c \in C$.

$$\min \sum_{c=1}^{n} y_c$$ [minimize total # colors used]

s.t. $$\sum_{c=1}^{n} x_{ic} = 1 \quad \forall v_i \in V(G)$$ [each vertex is assigned exactly one color]

$$x_{ic} + x_{jc} \leq y_c \quad \forall v_iv_j \in E(G) \text{ and } \forall c \in C$$ ← ?

Chromatic number as an optimization problem:

Input G with $V(G) = \{v_1, \ldots, v_n\}$, $E(G)$.

Available colors $C = \{1, 2, \ldots, n\}$

Let $x_{ic} = \begin{cases} 1 & \text{if vertex } v_i \text{ is colored with } c \in C \\ 0 & \text{otherwise} \end{cases}$ for each $v_i \in V(G)$ and $c \in C$

Let $y_c = \begin{cases} 1 & \text{if color } c \text{ is used} \\ 0 & \text{otherwise} \end{cases}$ for each $c \in C$.

$$\min \sum_{c=1}^{n} y_c$$

[minimize total # colors used]

s.t. $\sum_{c=1}^{n} x_{ic} = 1 \quad \forall v_i \in V(G)$ [each vertex is assigned exactly one color]

$x_{ic} + x_{jc} \leq y_c \quad \forall v_i v_j \in E(G) \text{ and } \forall c \in C$

[Any used color is assigned to exactly one vertex out of two with an edge between them]

$x_{ic} \in \{0,1\} \quad \forall i \, \forall c$

$y_c \in \{0,1\} \quad \forall c$

Finding $\chi(G)$ or even finding a good coloring is a very hard computational problem.

A simple algorithm is often the starting point, and often very important for its versatility (for parallel computing / for fault tolerant computing / ...).

Greedy Algorithm

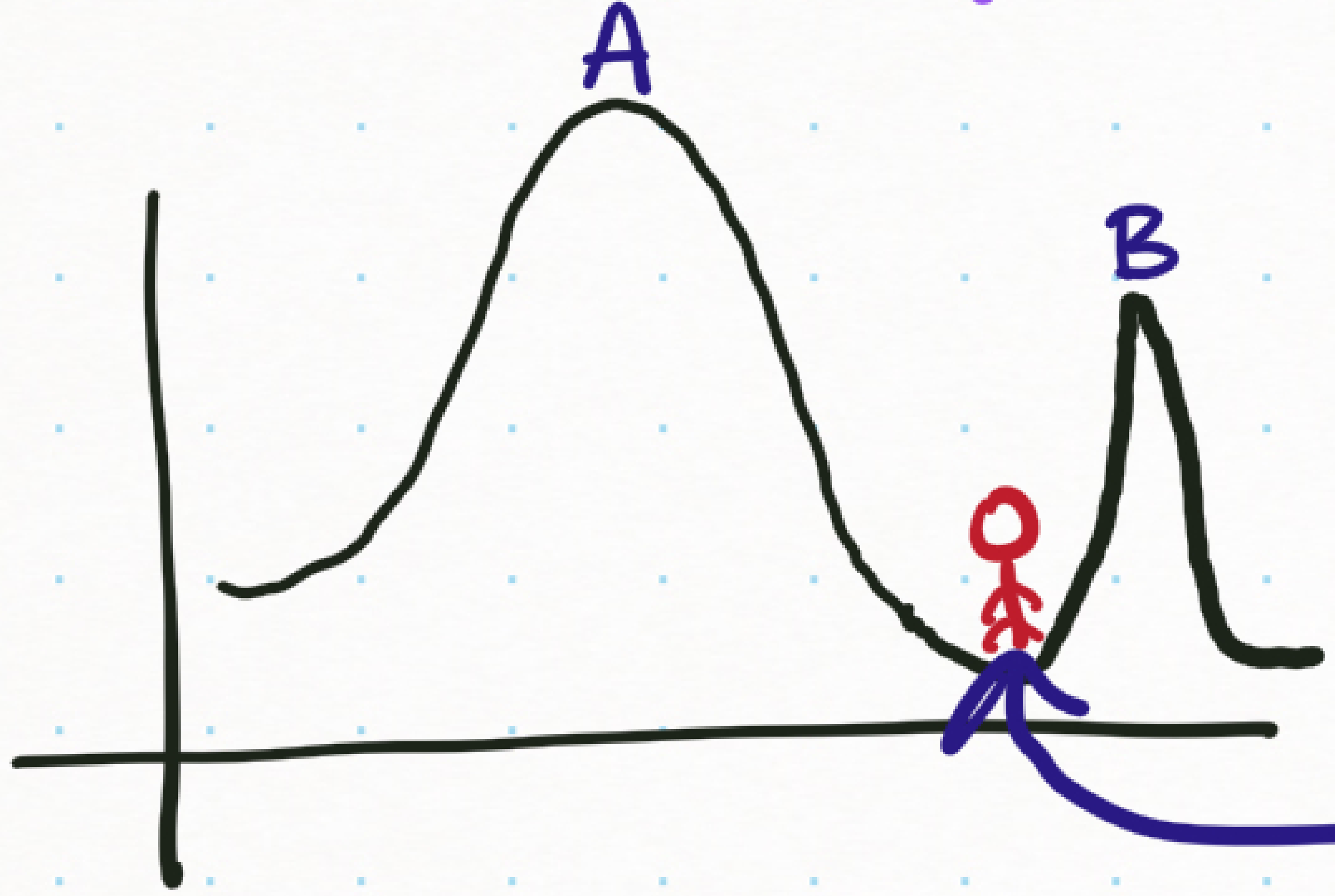
Build a partial solution, one step at a time, and at each step make a choice that is best (based on the objective) at that step (and for your existing partial solution).

Finding $\chi(G)$ or even finding a good coloring is a very hard computational problem.

A simple algorithm is often the starting point, and often very important for its versatility (for parallel computing / for fault tolerant computing / ...).

Greedy Algorithm

Build a partial solution, one step at a time, and at each step make a choice that is best (based on the objective) at that step (and for your existing partial solution).



Trying to find global maximum

Algo: always move in the direction of largest derivative

at the current step, we will move to the right until we reach B (local max) & miss A, the global max.

Finding $\chi(G)$ or even finding a good coloring is a very hard computational problem.

A simple algorithm is often the starting point, and often very important for its versatility (for parallel computing / for fault tolerant computing / ...).

Greedy Algorithm

Build a partial solution, one step at a time, and at each step make a choice that is best (based on the objective) at that step (and for your existing partial solution).

This is often combined with a local search algorithm to find better solution for next step quickly.

Only a local optimum is guaranteed in general.

Greedy Coloring

Order the vertices of the input graph into a sequence:

$v_1, v_2, \ldots, v_n$

For $i = 1$ to n

Assign the smallest available feasible color to v_i

smallest color that has not already been used on any of the neighbors of v_i

Greedy Coloring

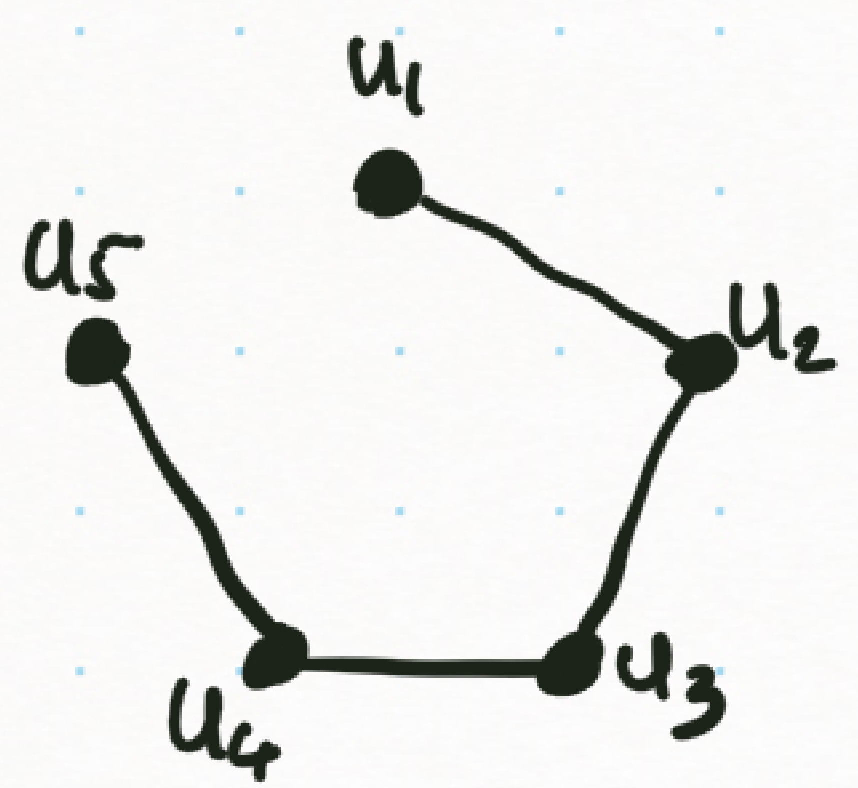
Order the vertices of the input graph into a sequence:

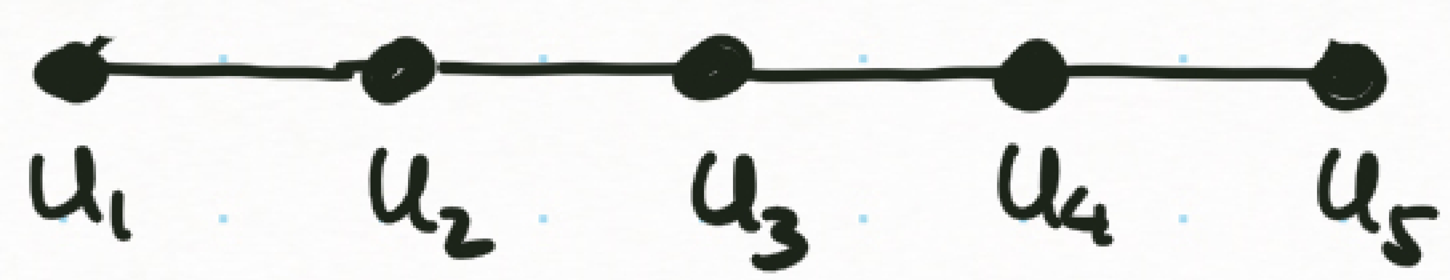
$v_1, v_2, \ldots, v_n$

For $i = 1$ to n

Assign the smallest available feasible color to v_i

smallest color that has not already been used on any of the neighbors of v_i





Path, P_5

Greedy Coloring

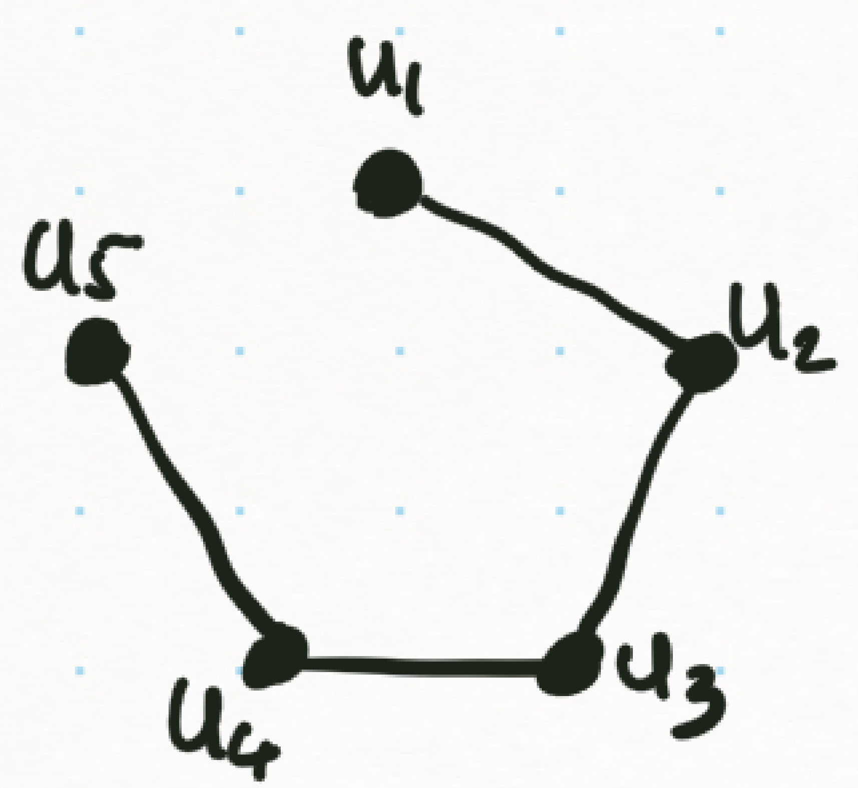
Order the vertices of the input graph into a sequence:

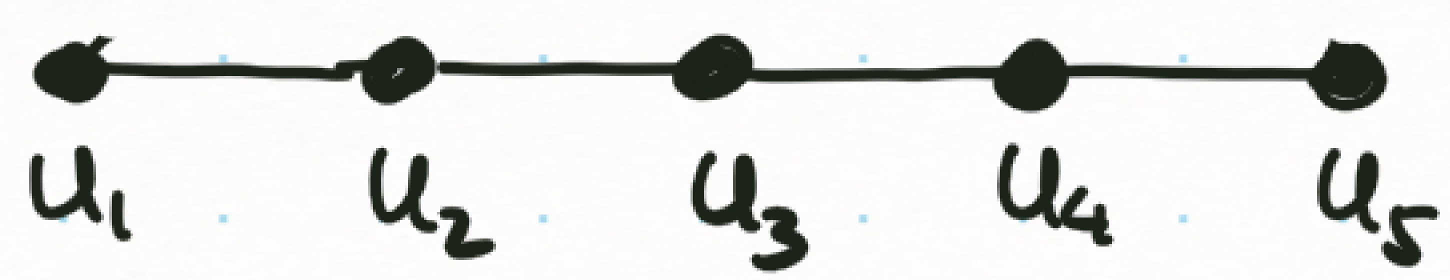
$v_1, v_2, \ldots, v_n$

For $i = 1$ to n

Assign the smallest available feasible color to v_i

smallest color that has not already been used on any of the neighbors of v_i





color: 1

Greedy Coloring

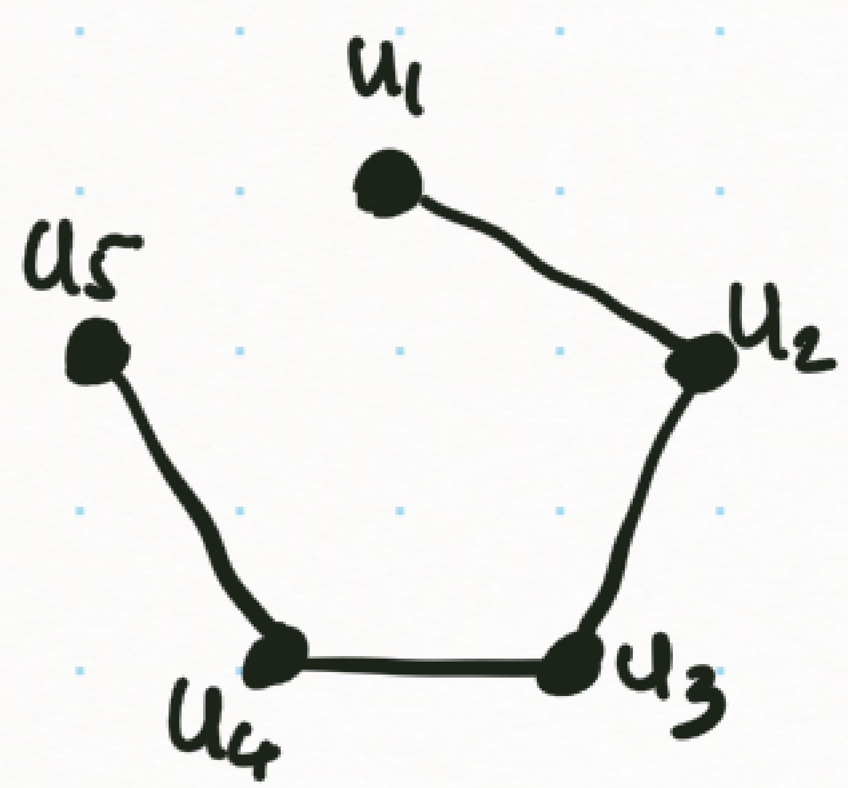
Order the vertices of the input graph into a sequence:

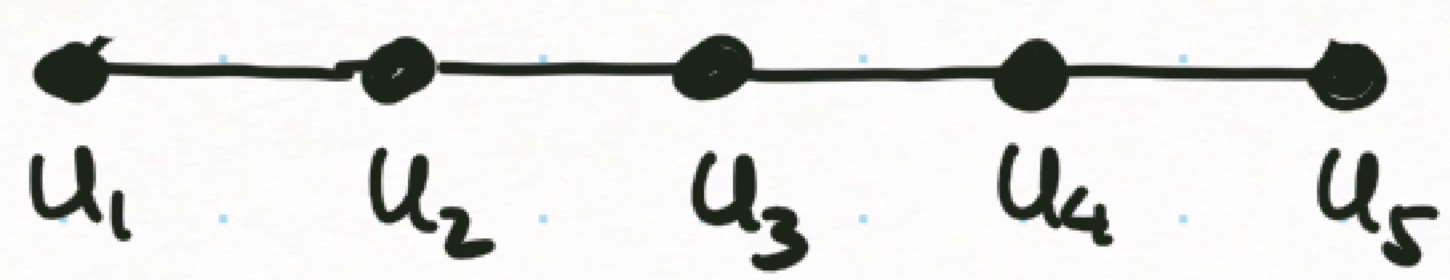
$v_1, v_2, \ldots, v_n$

For $i = 1$ to n

Assign the smallest available feasible color to v_i

smallest color that has not already been used on any of the neighbors of v_i





color: 1 ~~1~~ 2

Greedy Coloring

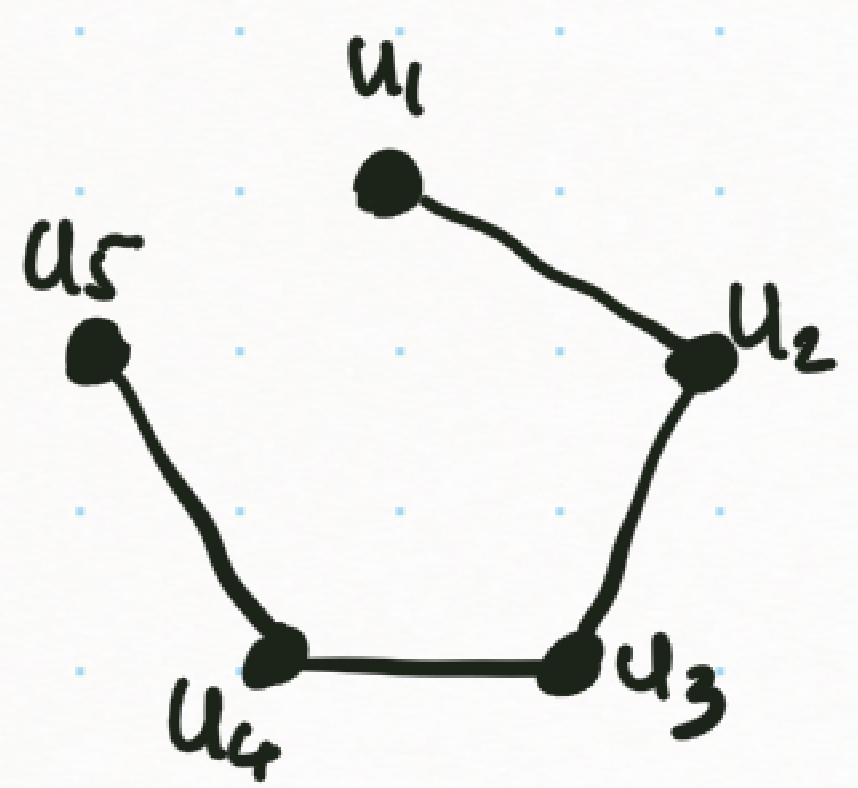
Order the vertices of the input graph into a sequence:

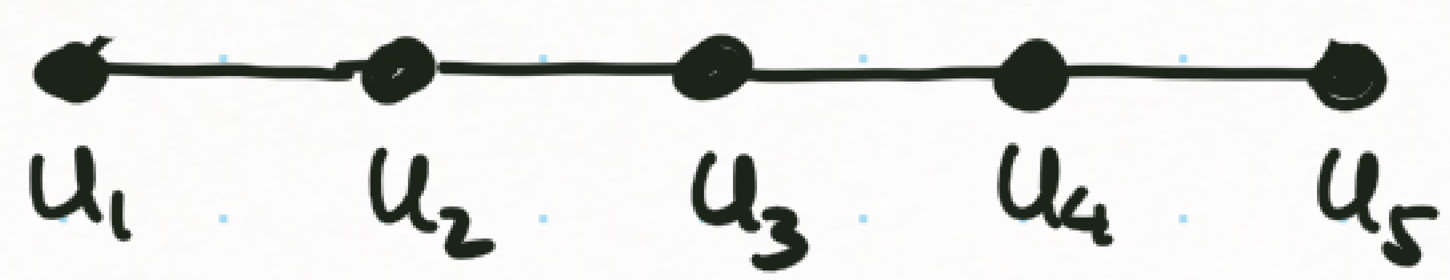
$v_1, v_2, \ldots, v_n$

For $i=1$ to n

Assign the smallest available feasible color to v_i

smallest color that has not already been used on any of the neighbors of v_i





color: 1 ~~1~~ 2 1

Greedy Coloring

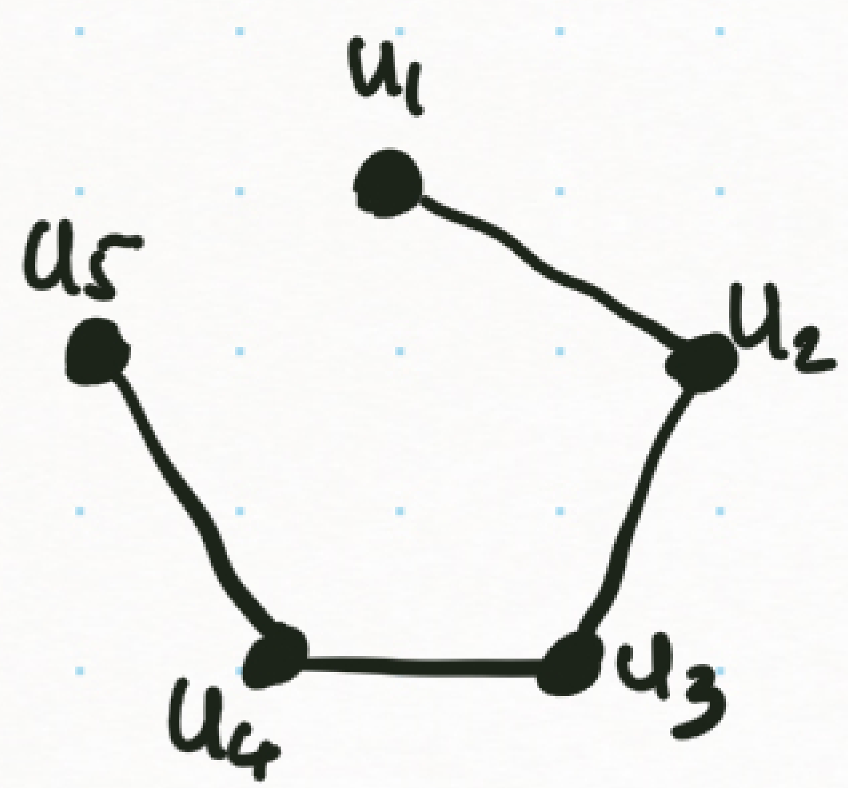
Order the vertices of the input graph into a sequence:

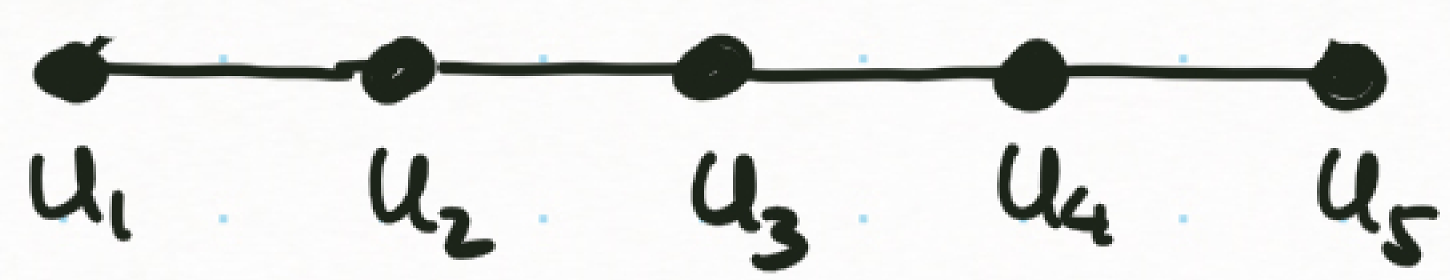
$v_1, v_2, \ldots, v_n$

For $i = 1$ to n

Assign the smallest available feasible color to v_i

smallest color that has not already been used on any of the neighbors of v_i





color: 1 $\cancel{1}$ 2 1 $\cancel{1}$ 2

Greedy Coloring

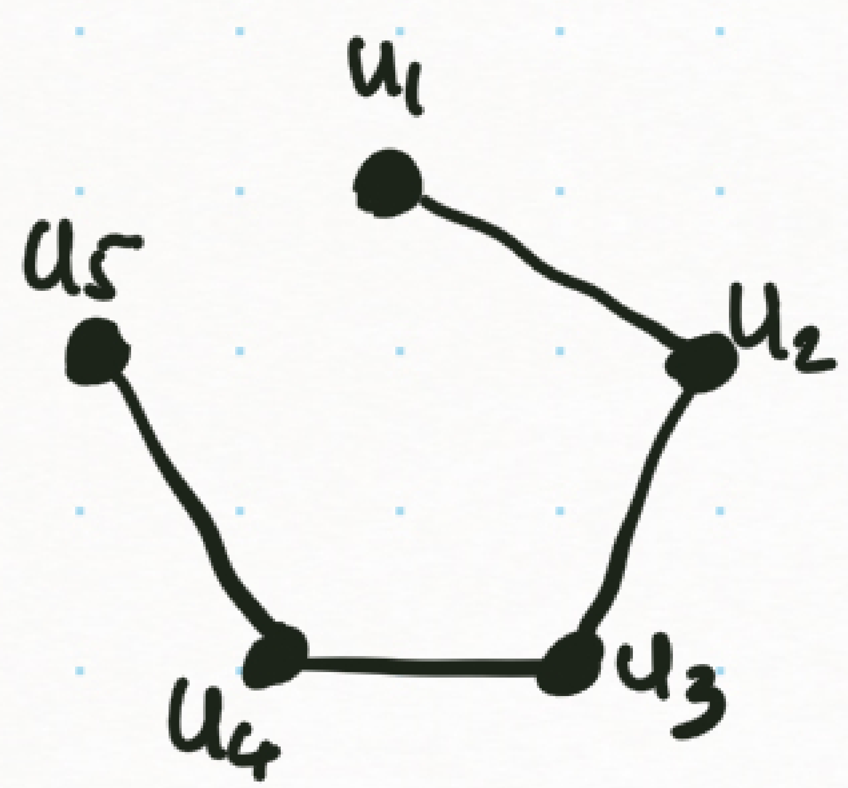
Order the vertices of the input graph into a sequence:

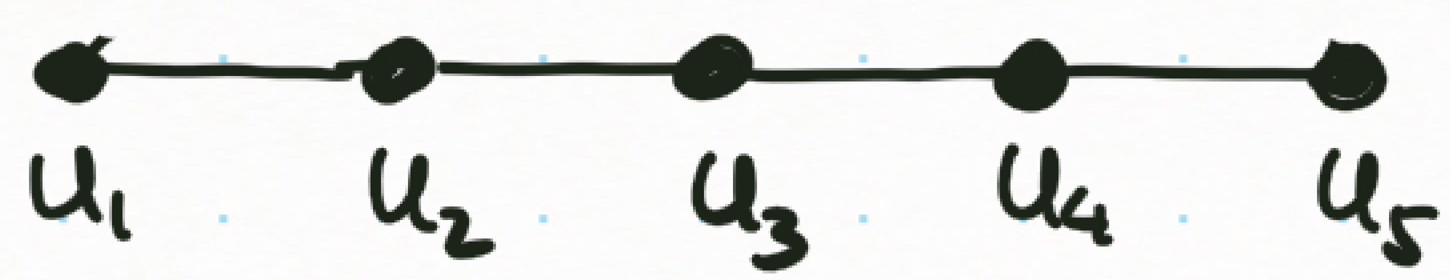
$v_1, v_2, \ldots, v_n$

For $i = 1$ to n

Assign the smallest available feasible color to v_i

smallest color that has not already been used on any of the neighbors of v_i





color: 1 ✓ ~~1~~ 2 ✓ 1 ✓ ~~1~~ 2 ✓ ~~1~~ 2 ✓

$\therefore \chi(G) \leq 2$

for $G = P_n$, path on n vertices

Greedy Coloring

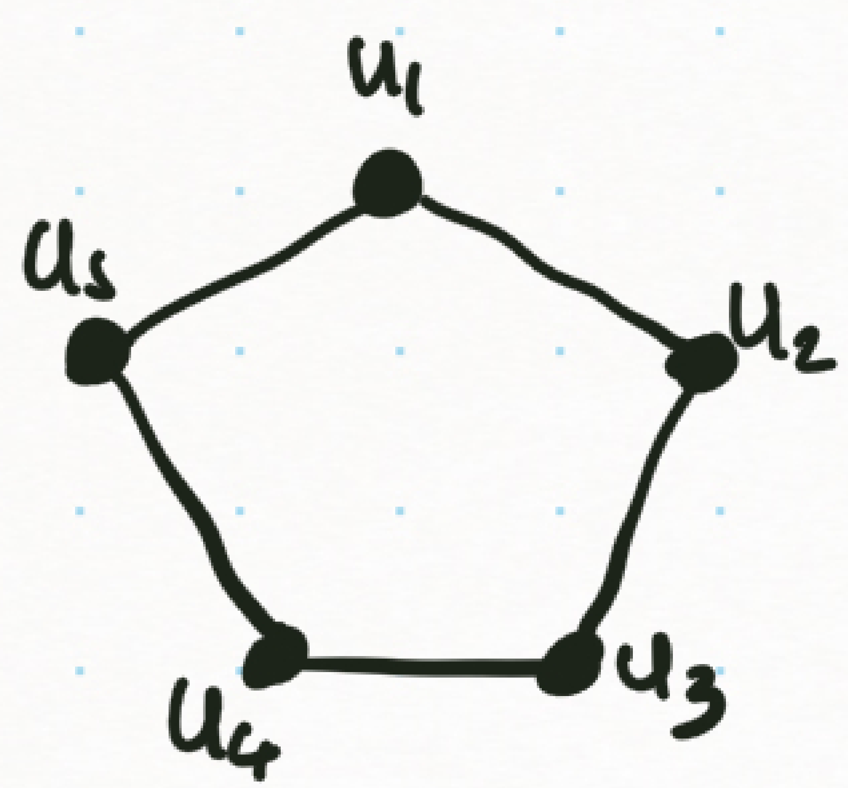
Order the vertices of the input graph into a sequence:

$v_1, v_2, \ldots, v_n$

For $i=1$ to n

Assign the smallest available feasible color to v_i

smallest color that has not already been used on any of the neighbors of v_i



Cycle, C_5

Greedy Coloring

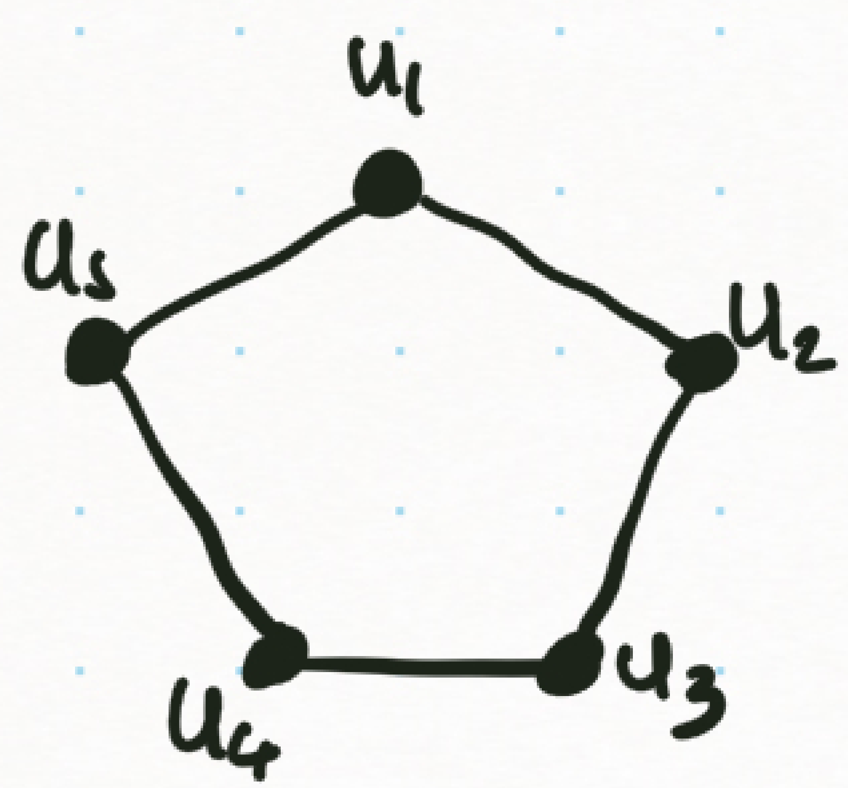
Order the vertices of the input graph into a sequence:

$v_1, v_2, \ldots, v_n$

For $i = 1$ to n

Assign the smallest available feasible color to v_i

smallest color that has not already been used on any of the neighbors of v_i



color: 1

Greedy Coloring

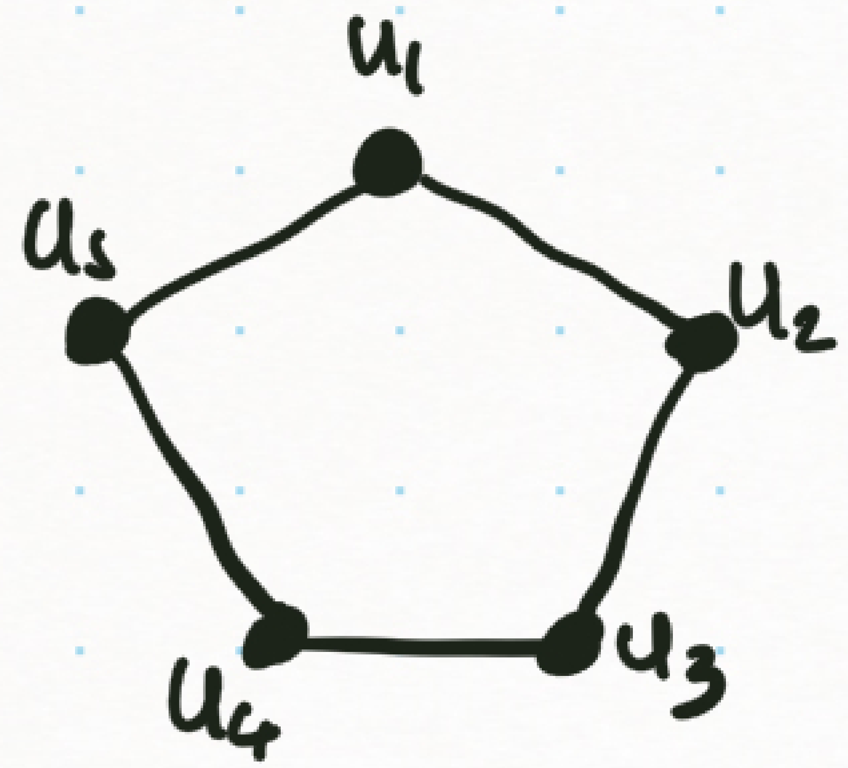
Order the vertices of the input graph into a sequence:

$v_1, v_2, \dots, v_n$

For $i = 1$ to n

Assign the smallest available feasible color to v_i

smallest color that has not already been used on any of the neighbors of v_i



color: 1 ~~1~~ 2

Greedy Coloring

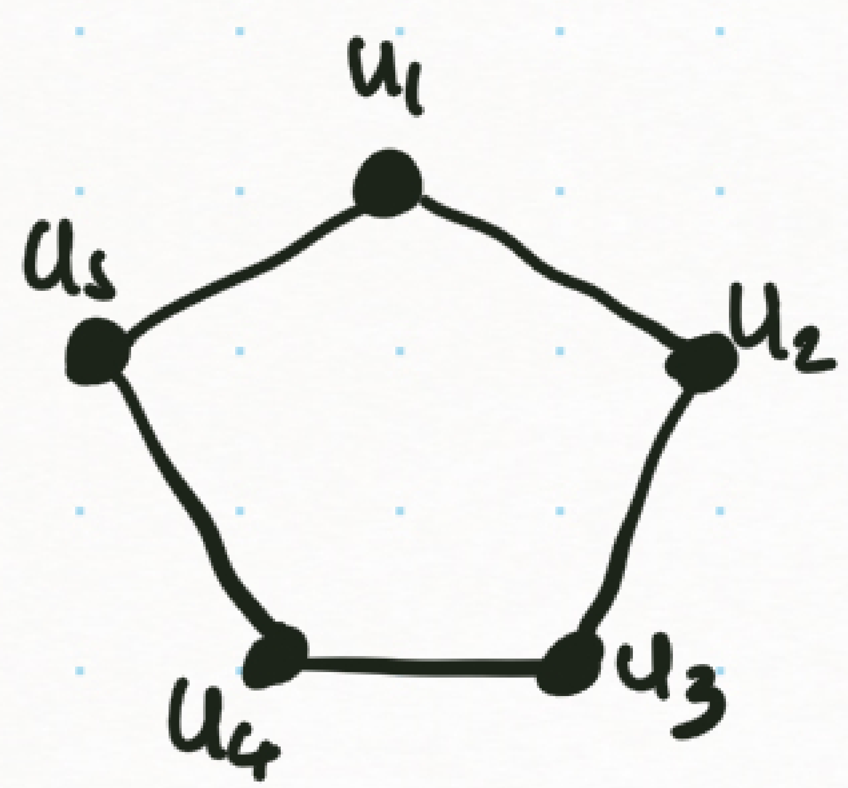
Order the vertices of the input graph into a sequence:

$v_1, v_2, \ldots, v_n$

For $i = 1$ to n

Assign the smallest available feasible color to v_i

smallest color that has not already been used on any of the neighbors of v_i



color: 1 ~~1~~ 2 1 ~~1~~ 2

Greedy Coloring

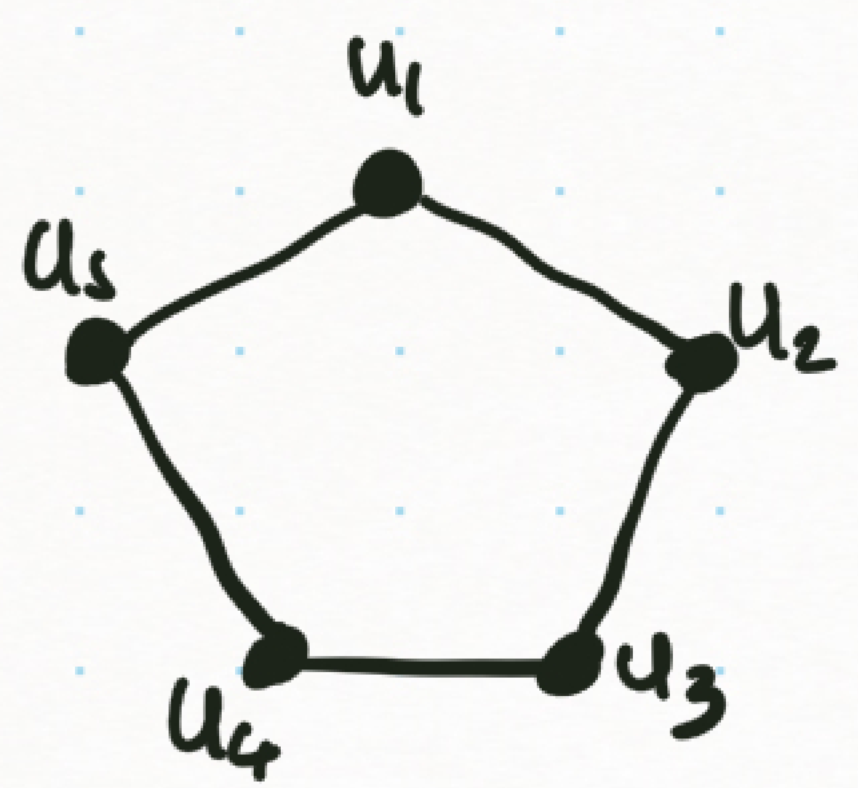
Order the vertices of the input graph into a sequence:

$v_1, v_2, \ldots, v_n$

For $i = 1$ to n

Assign the smallest available feasible color to v_i

smallest color that has not already been used on any of the neighbors of v_i



color: 1 ~~1~~ 2 1 ~~1~~ 2 ~~1~~ ~~2~~ 3

$\therefore \chi(G) \leq 3$ where $G = C_n$, cycle on n vertices.

Greedy Coloring

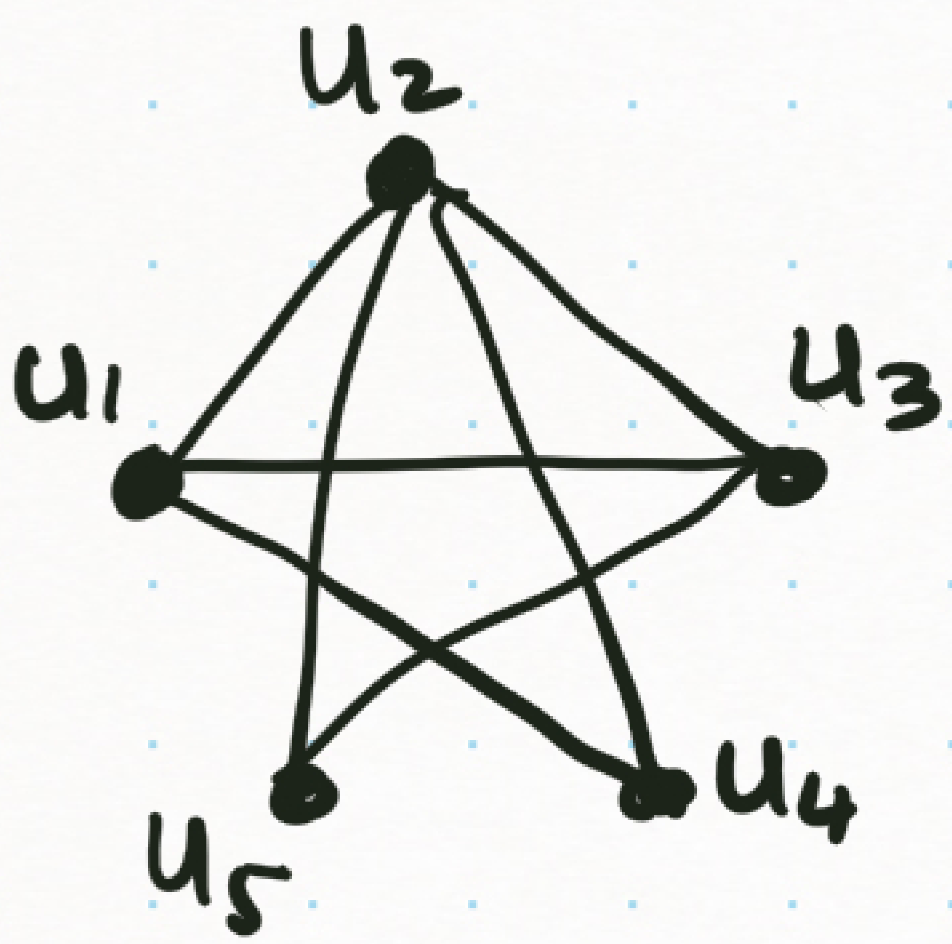
Order the vertices of the input graph into a sequence:

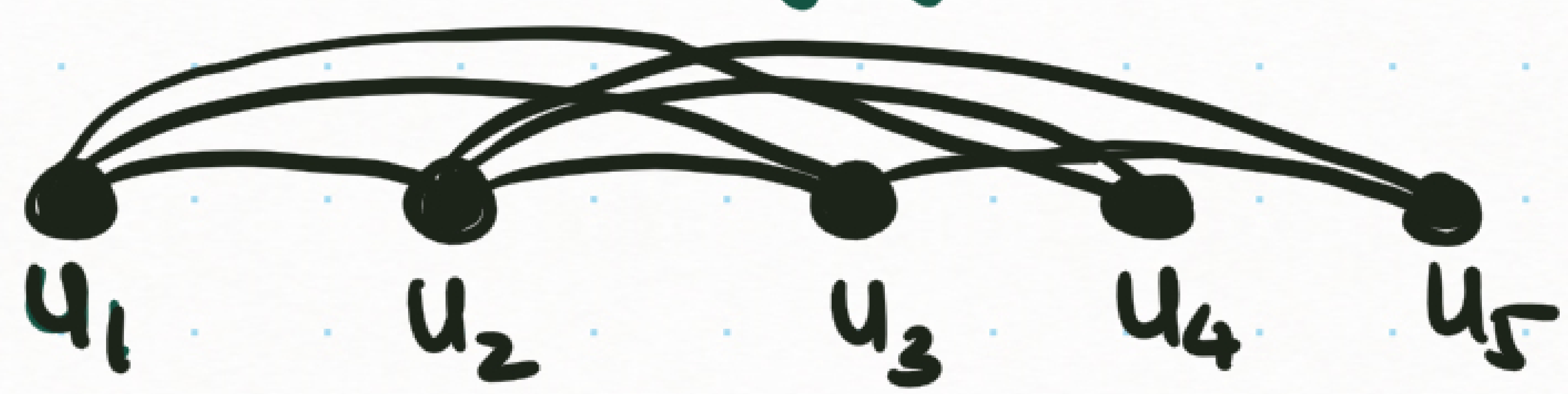
$v_1, v_2, \ldots, v_n$

For $i = 1$ to n

Assign the <u>smallest available feasible color</u> to v_i

smallest color that has not already been used on any of the neighbors of v_i





Greedy Coloring

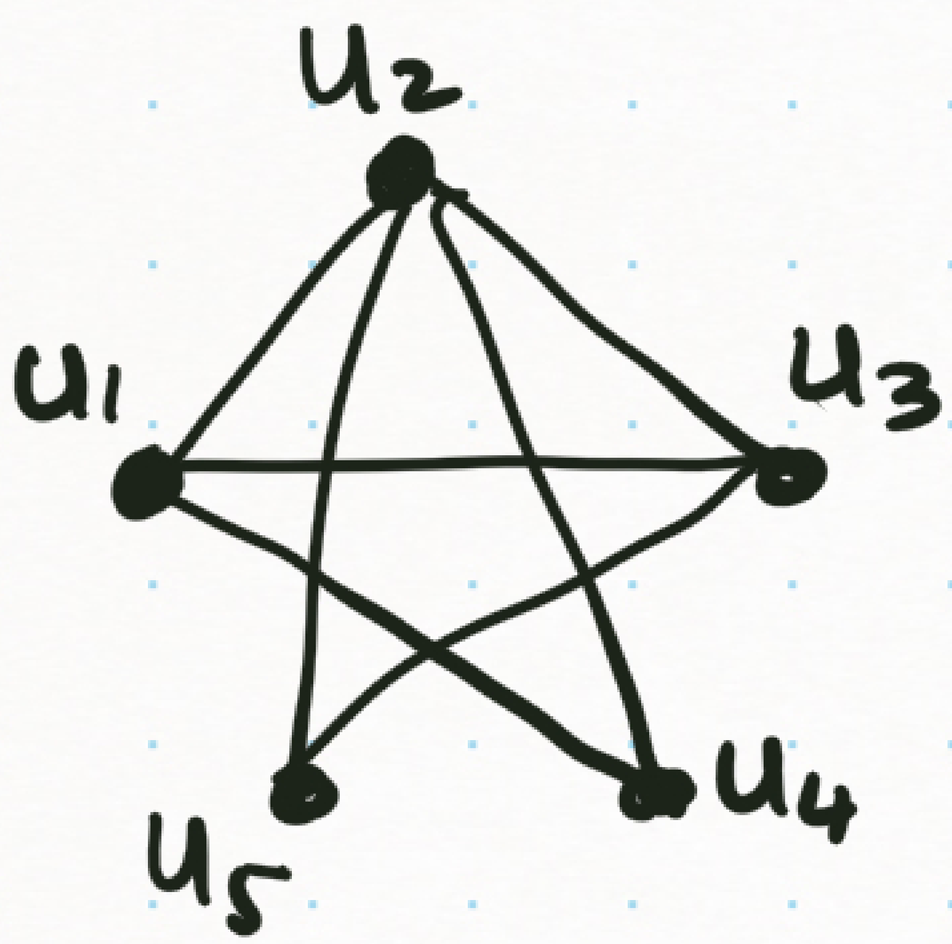
Order the vertices of the input graph into a sequence:

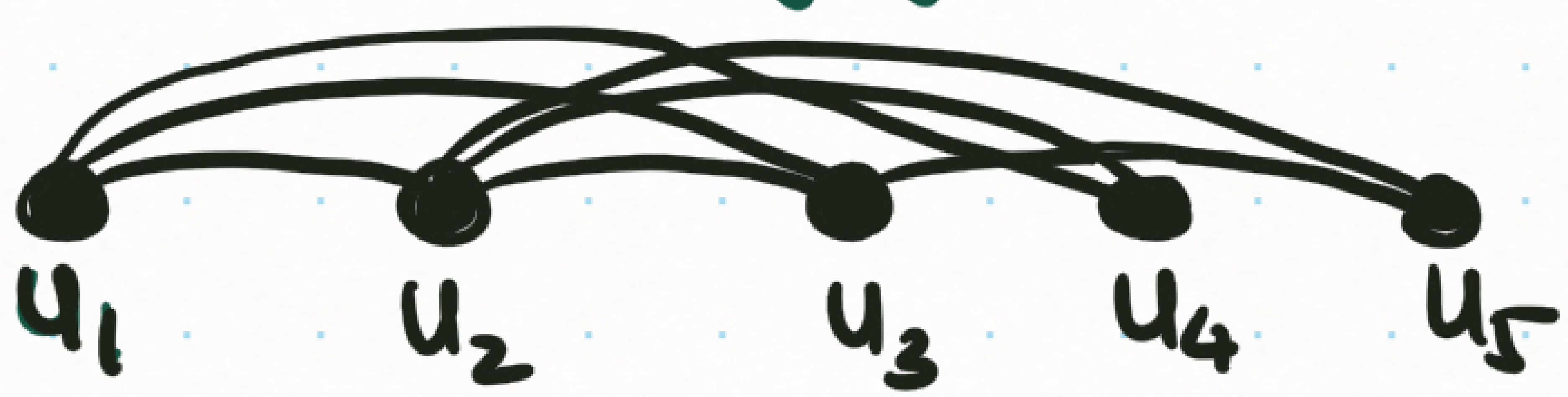
$v_1, v_2, \ldots, v_n$

For $i = 1$ to n

Assign the smallest available feasible color to v_i

smallest color that has not already been used on any of the neighbors of v_i





color: 1 ✓ 2 ✓ ~~1~~

Greedy Coloring

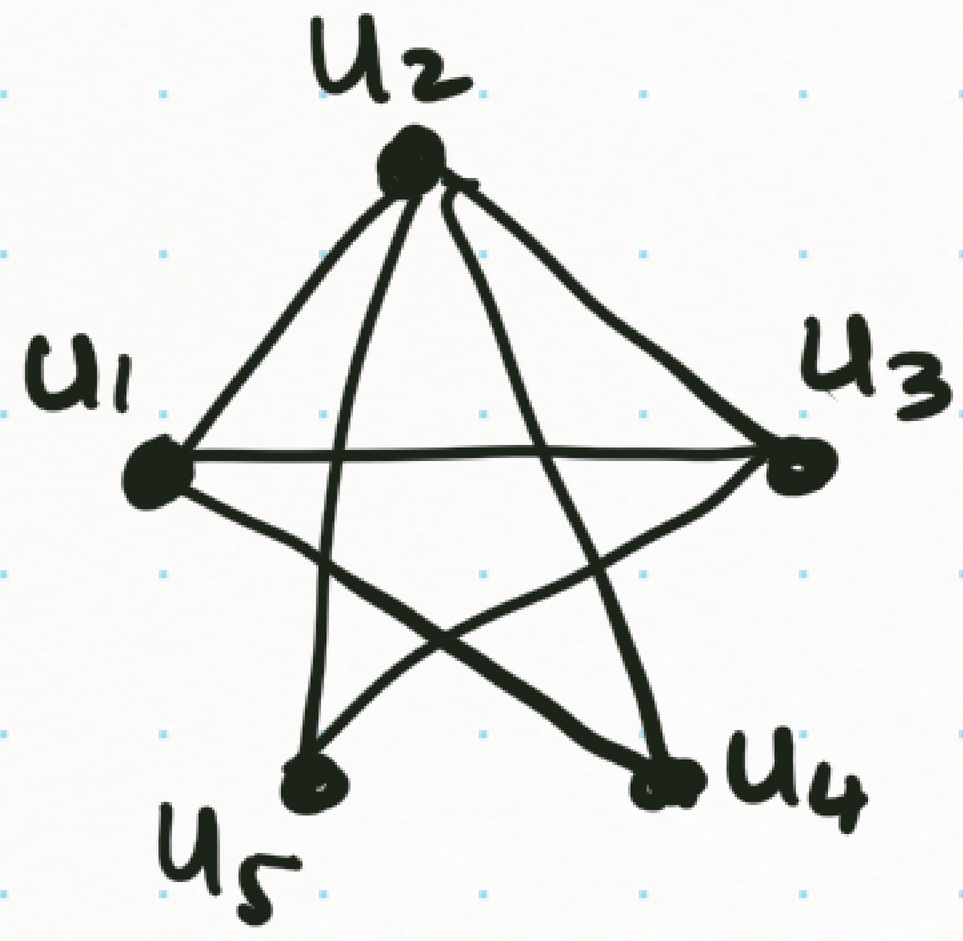
Order the vertices of the input graph into a sequence:

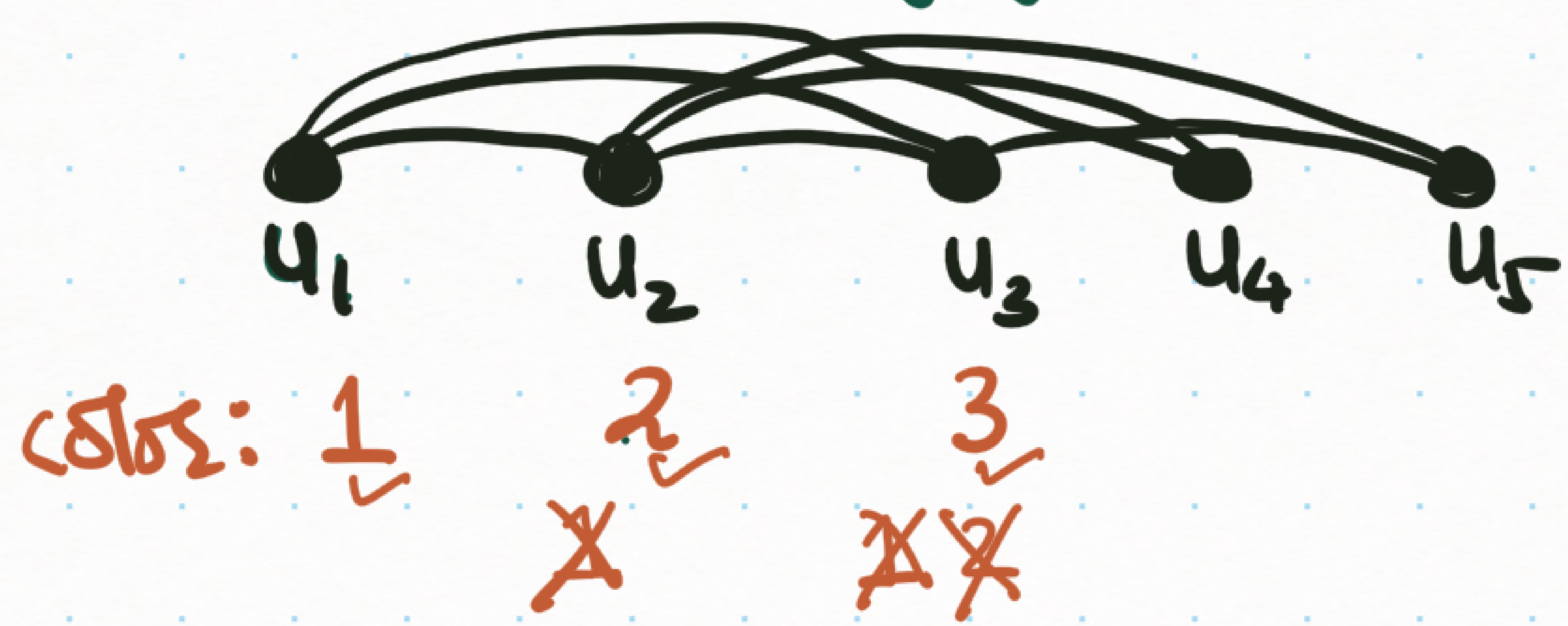
$v_1, v_2, \ldots, v_n$

For $i = 1$ to n

Assign the <u>smallest available feasible color</u> to v_i

smallest color that has not already been used on any of the neighbors of v_i





Greedy Coloring

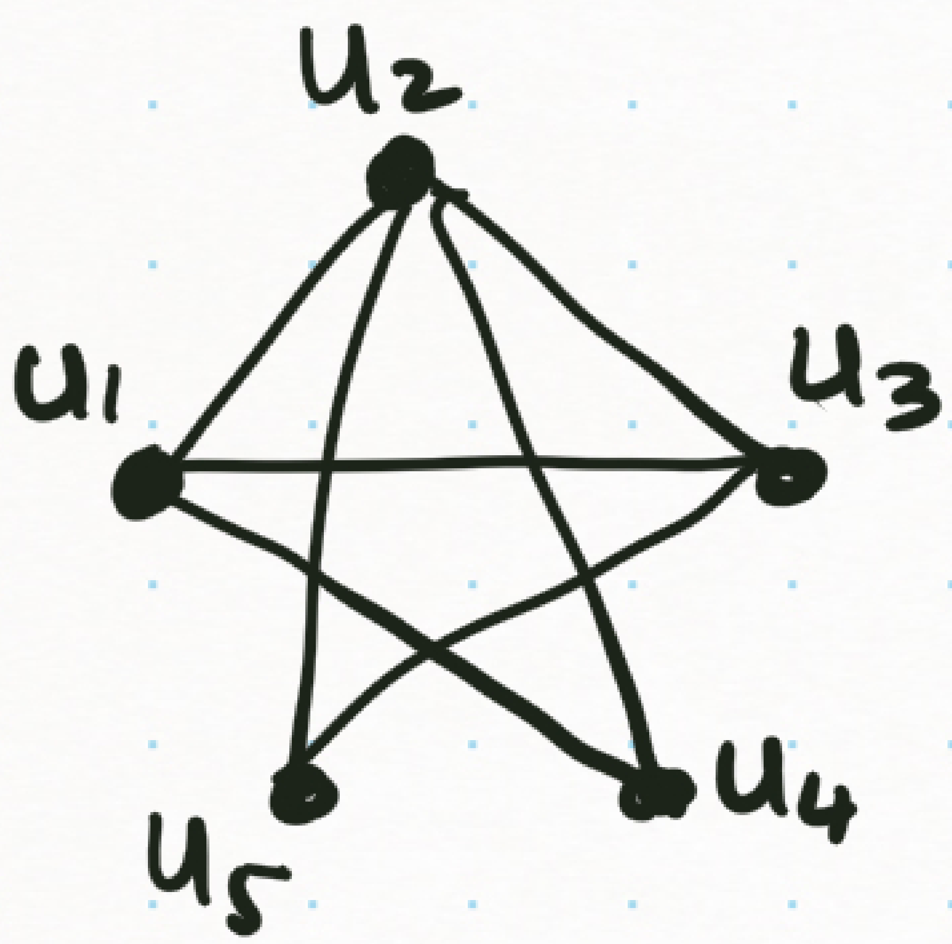
Order the vertices of the input graph into a sequence:

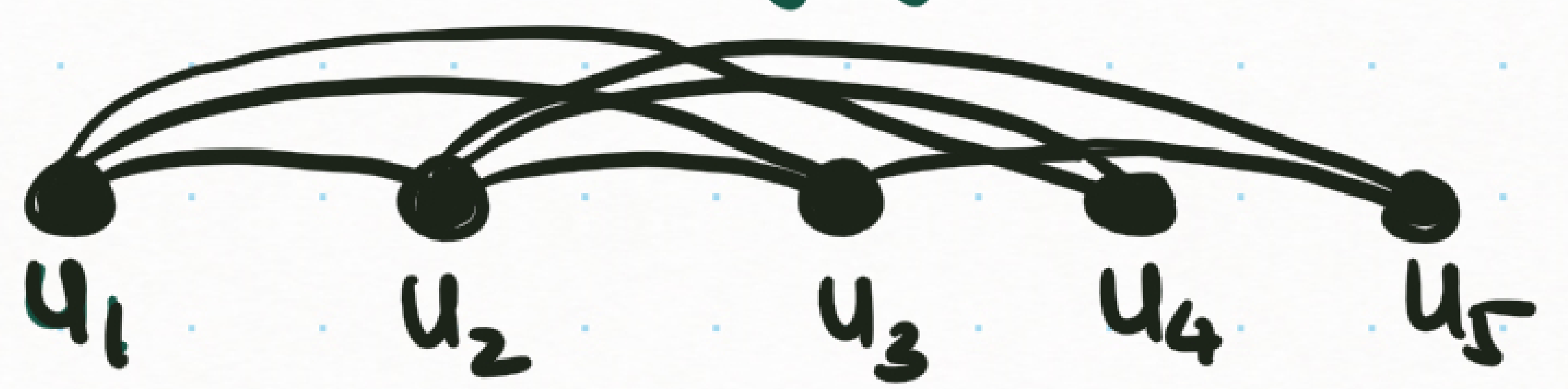
$v_1, v_2, \dots, v_n$

For $i = 1$ to n

Assign the smallest available feasible color to v_i

smallest color that has not already been used on any of the neighbors of v_i





color: 1 ✓   2 ✓   3 ✓   3 ✓   1 ✓

$\cancel{1}$ $\cancel{1}\ \cancel{2}$ $\cancel{1}\ \cancel{2}$ $\cancel{2}\ \cancel{3}$

$\therefore \chi(G) \le 3$

Greedy Coloring

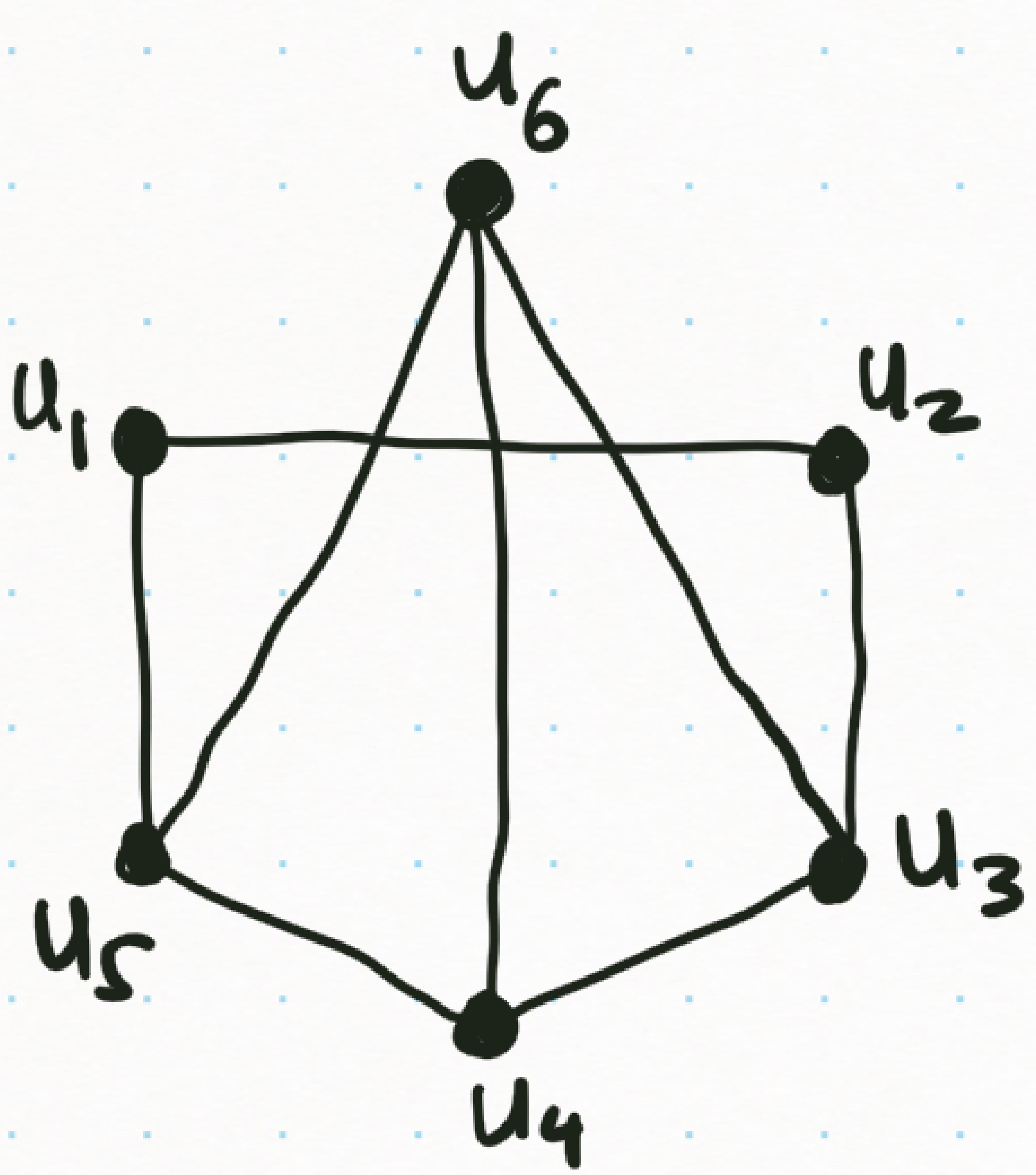
Order the vertices of the input graph into a sequence:

$v_1, v_2, \ldots, v_n$

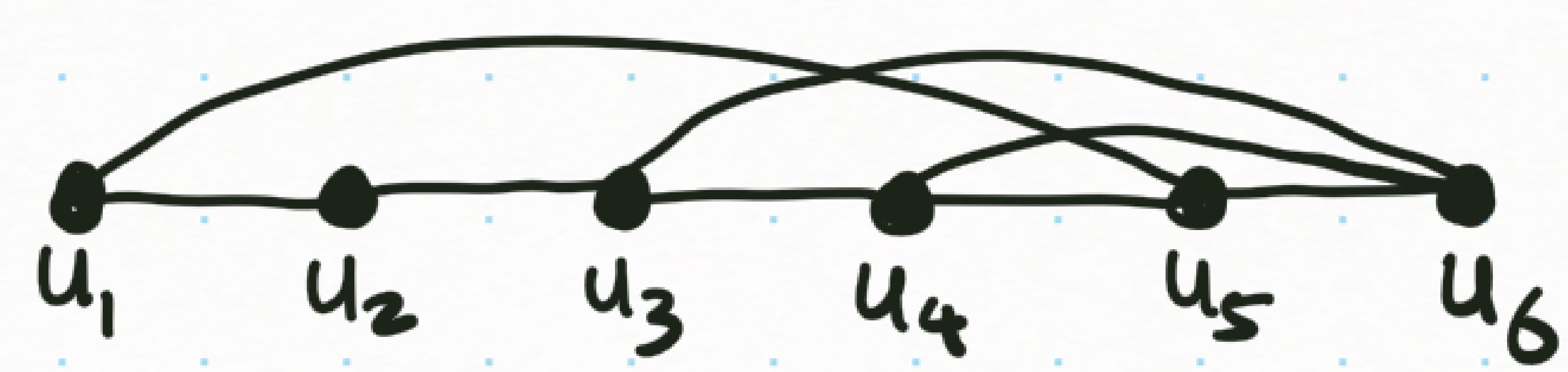
For $i = 1$ to n

Assign the smallest available feasible color to v_i

smallest color that has not already been used on any of the neighbors of v_i



G



Greedy Coloring

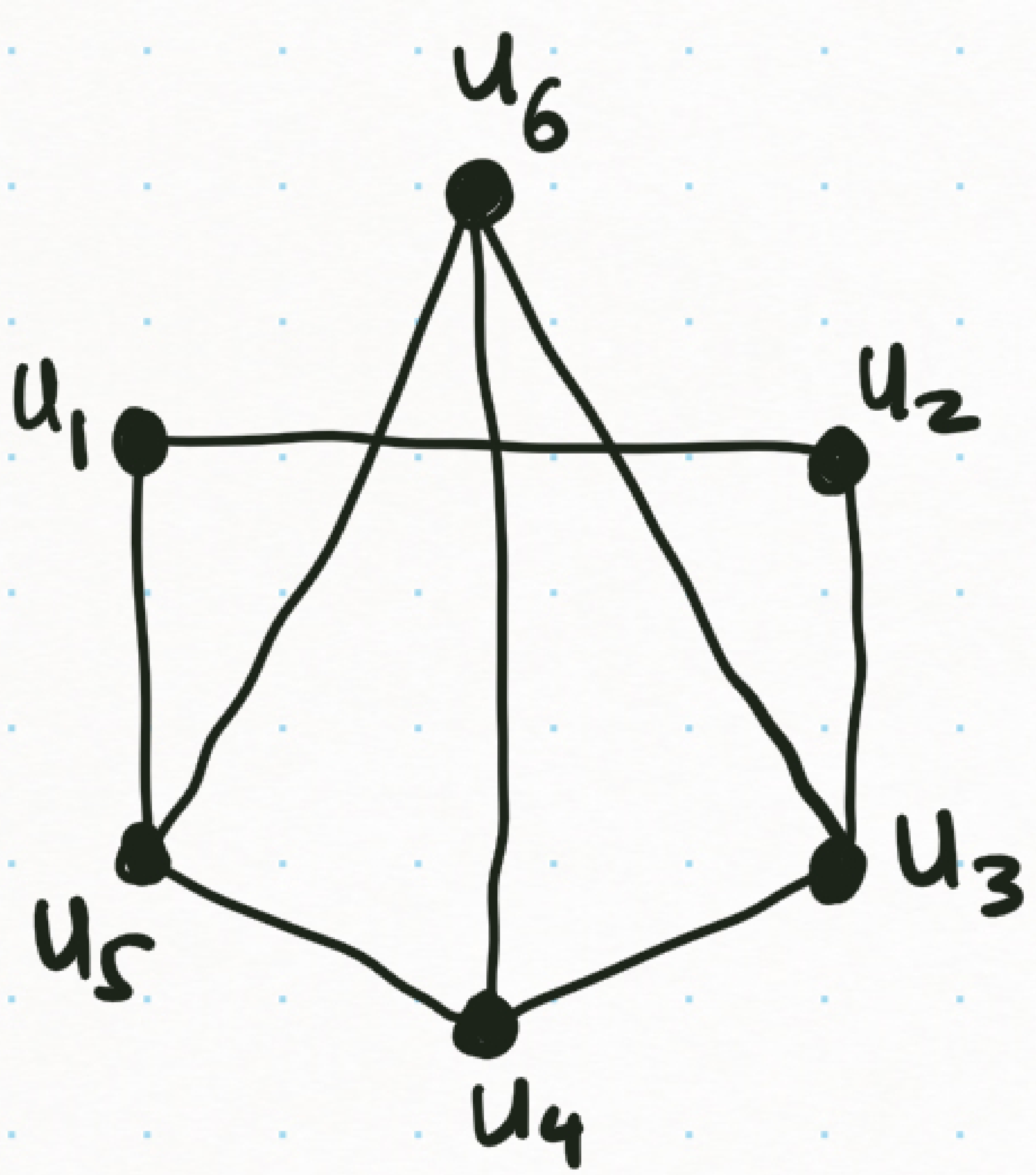
Order the vertices of the input graph into a sequence:

$v_1, v_2, \ldots, v_n$

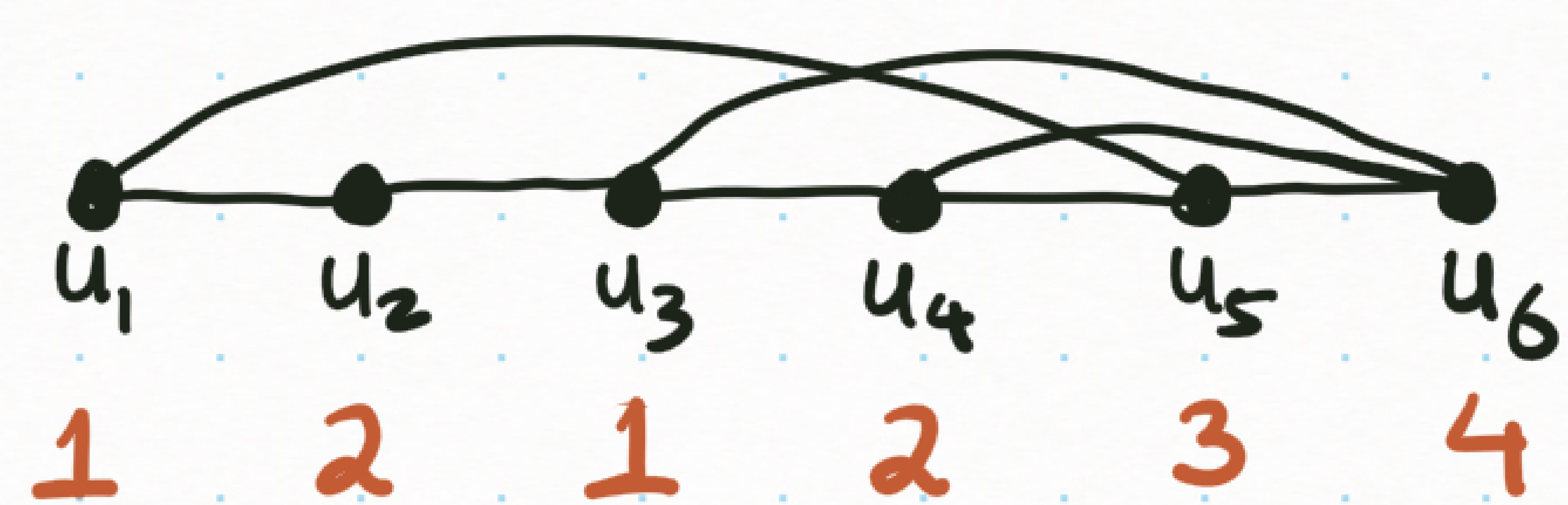
For $i = 1$ to n

Assign the smallest available feasible color to v_i

smallest color that has not already been used on any of the neighbors of v_i



G



$\Rightarrow \chi(G) \leq 4$

Is this the best ?

Greedy Coloring

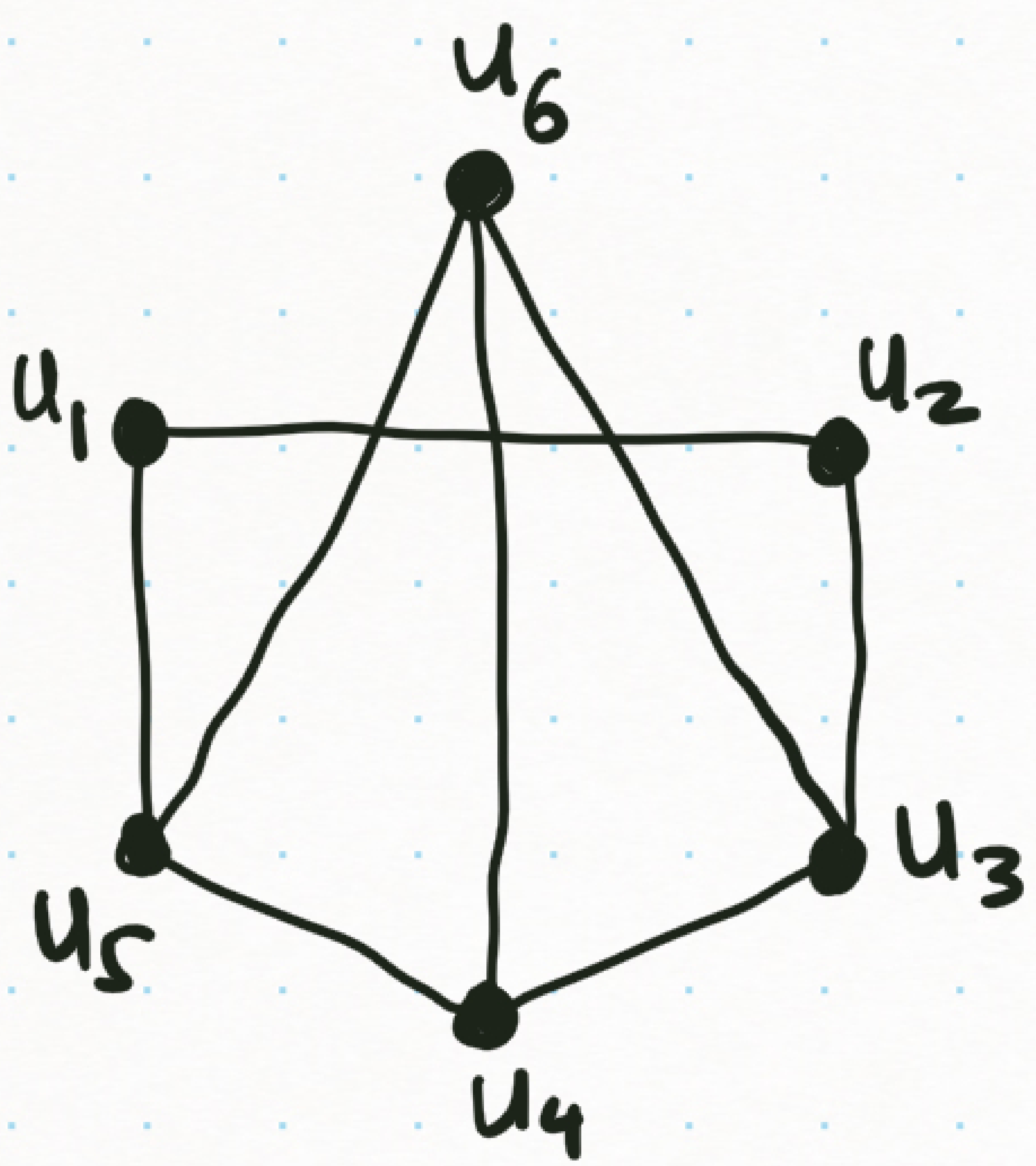
Order the vertices of the input graph into a sequence:

$v_1, v_2, \ldots, v_n$

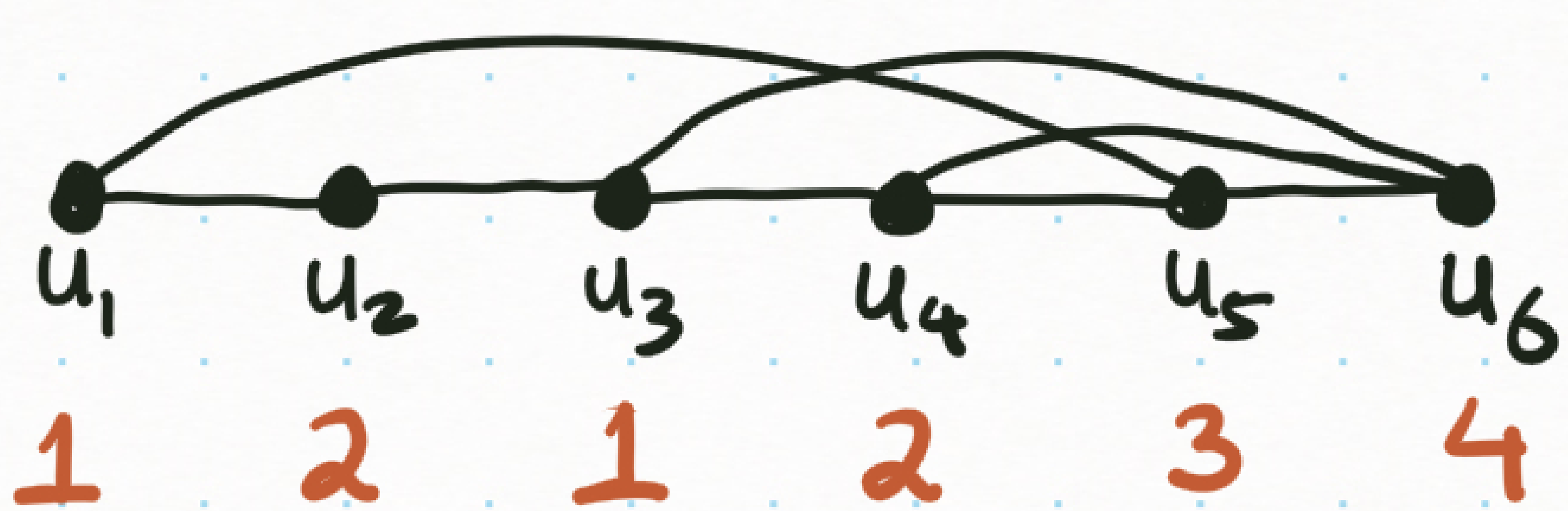
For $i = 1$ to n

Assign the smallest available feasible color to v_i

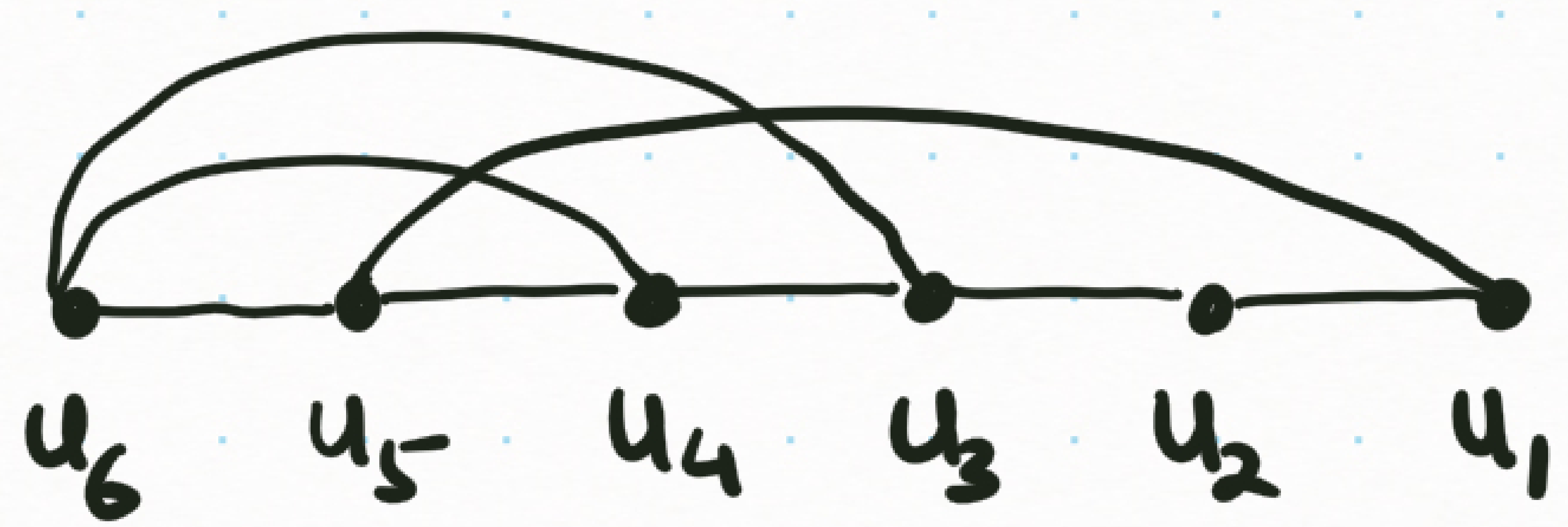
smallest color that has not already been used on any of the neighbors of v_i



G



$\Rightarrow \chi(G) \leq 4$



Greedy Coloring

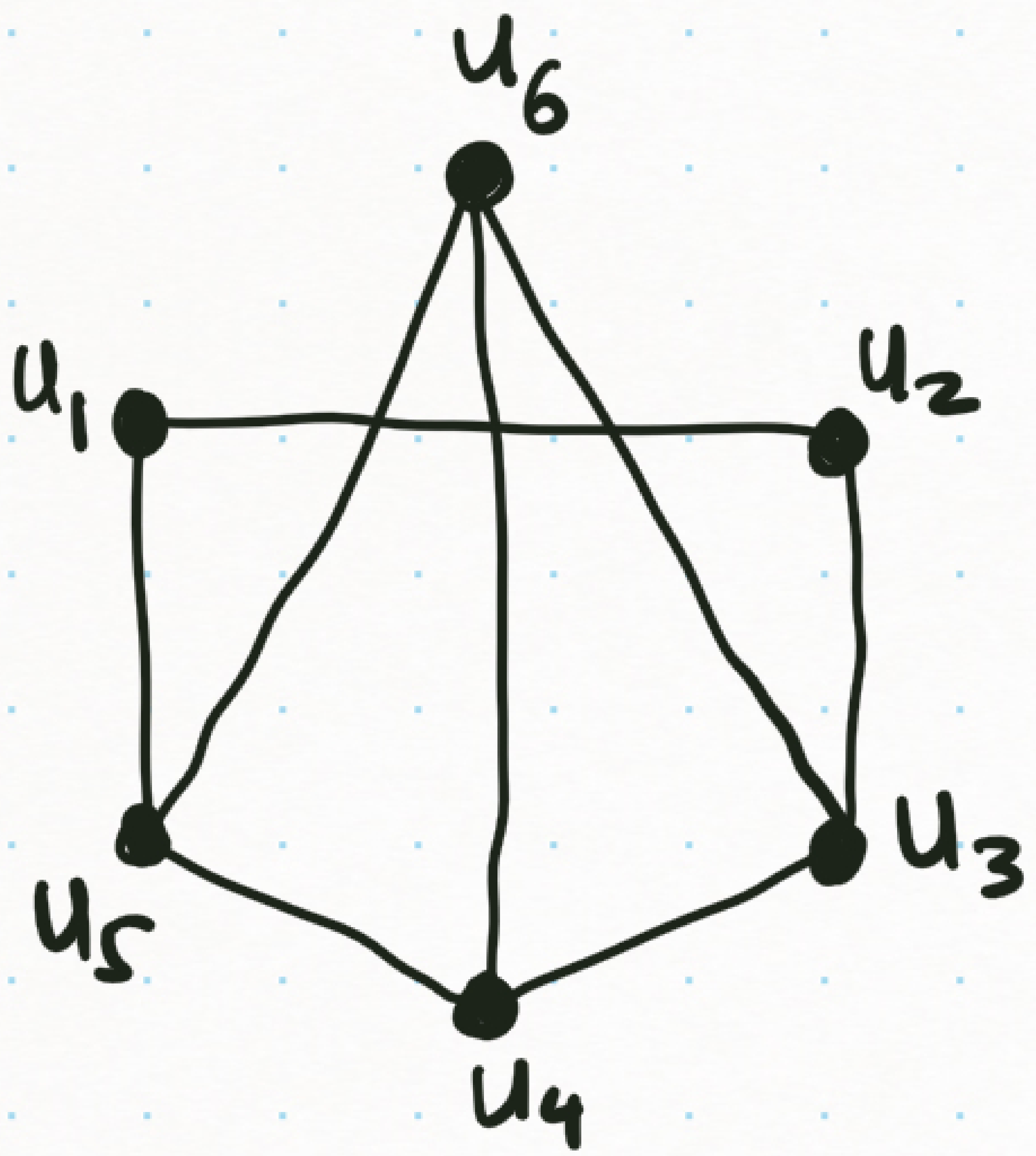
Order the vertices of the input graph into a sequence:

$v_1, v_2, \dots, v_n$

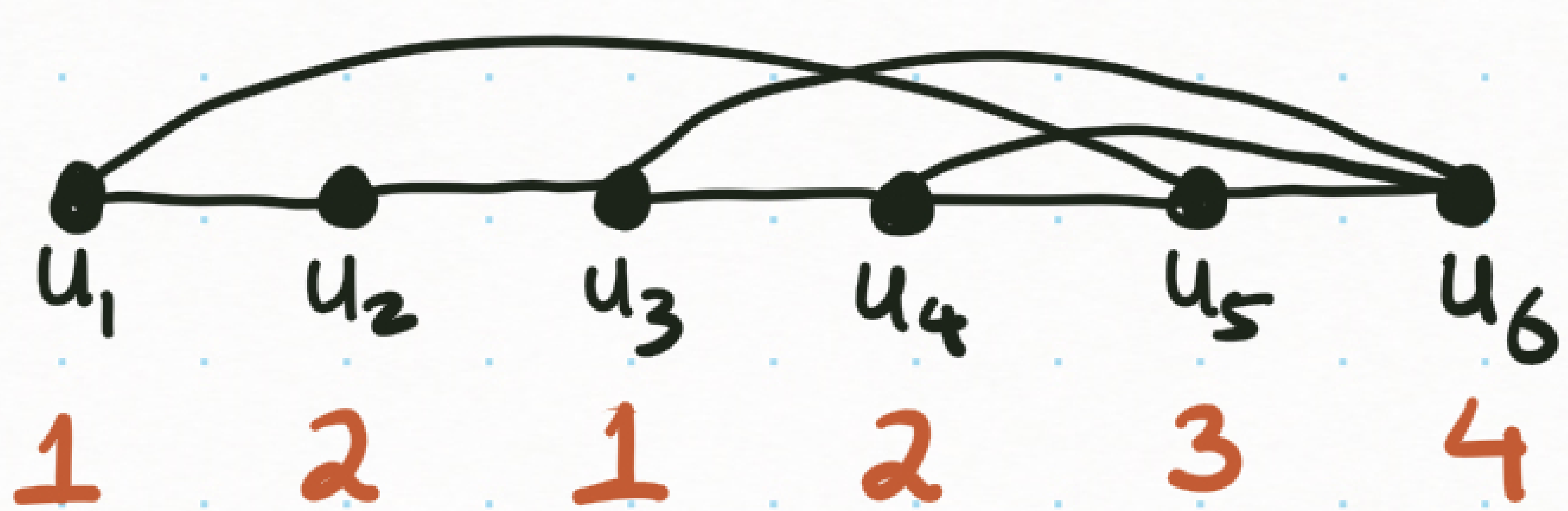
For $i = 1$ to n

Assign the smallest available feasible color to v_i

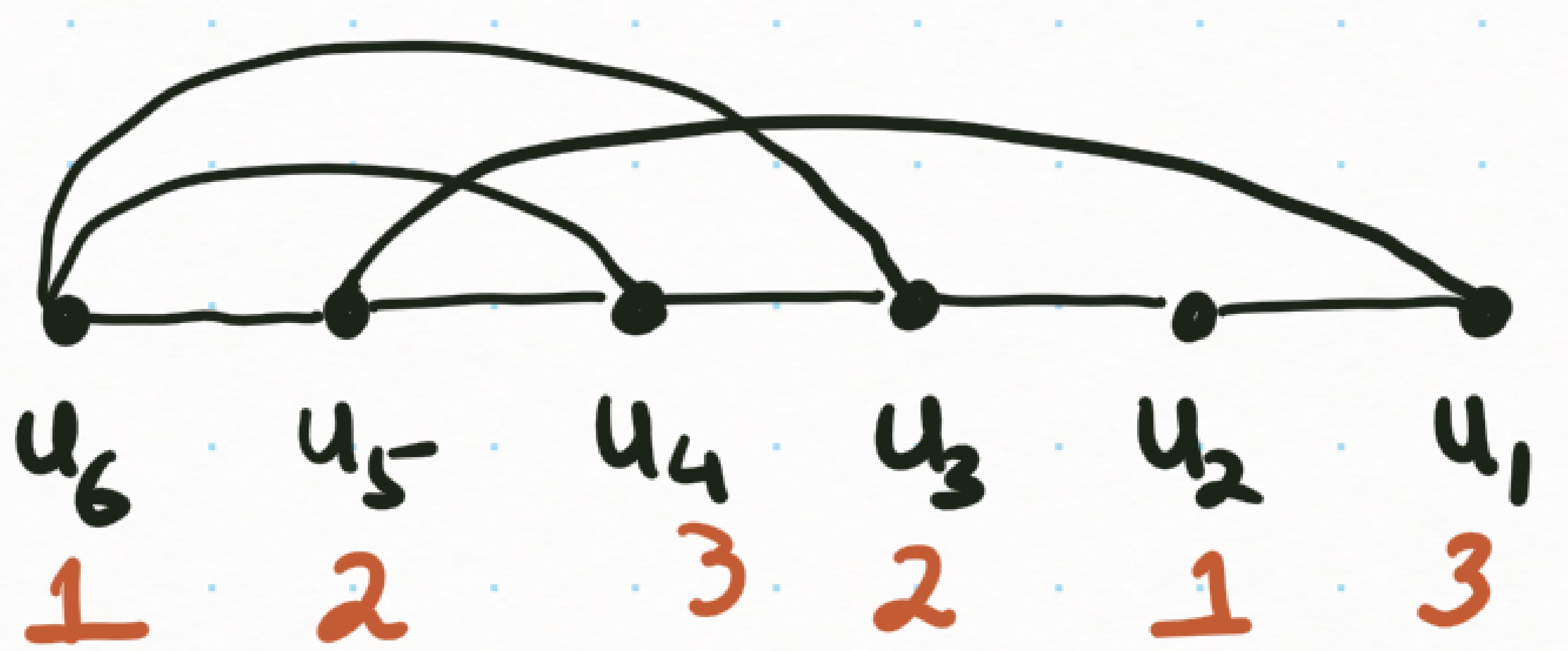
smallest color that has not already been used on any of the neighbors of v_i



G



$\Rightarrow \chi(G) \leq 4$



$\Rightarrow \chi(G) \leq 3$

Can we do better?

Greedy Coloring

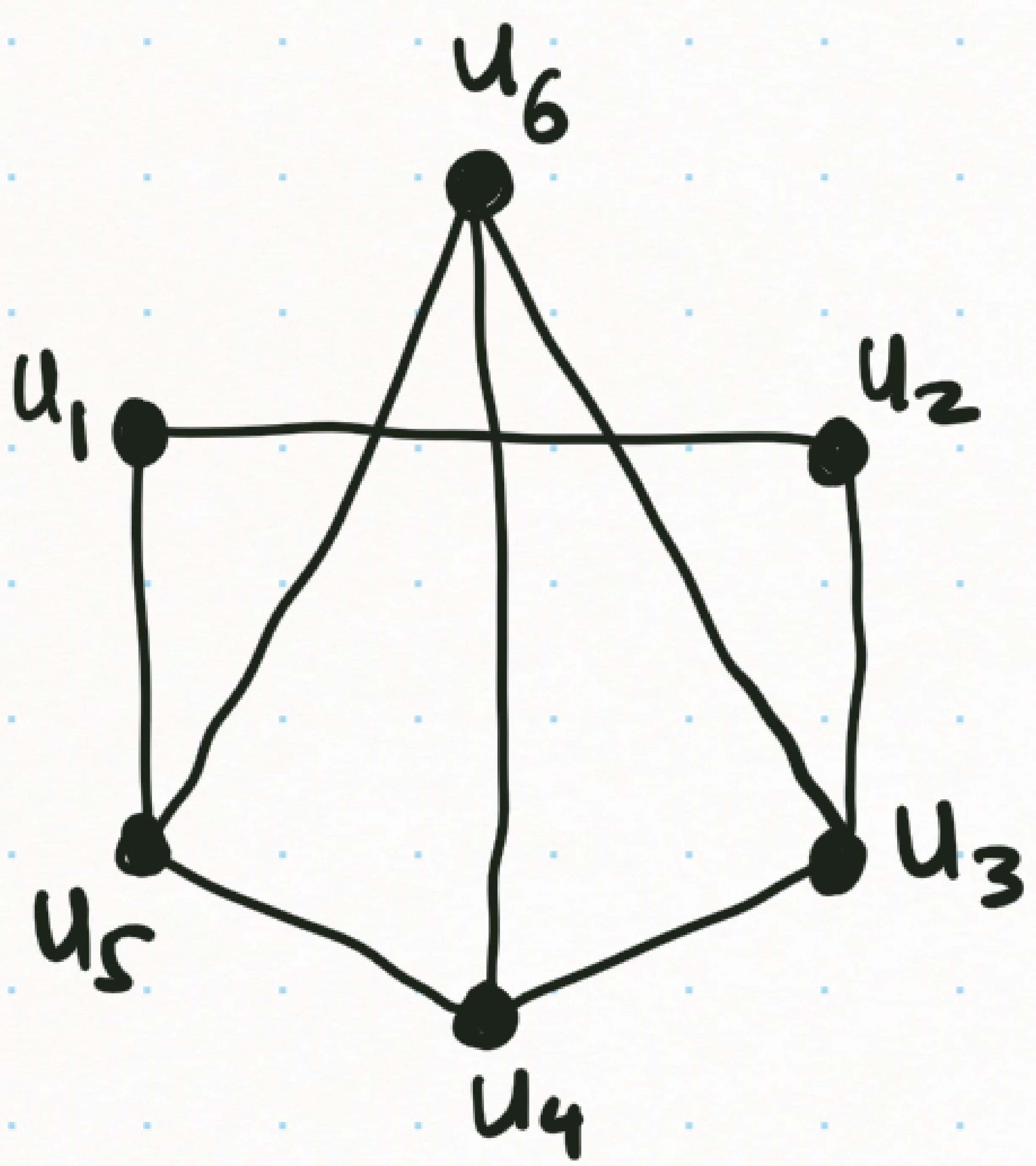
Order the vertices of the input graph into a sequence:

$v_1, v_2, \ldots, v_n$

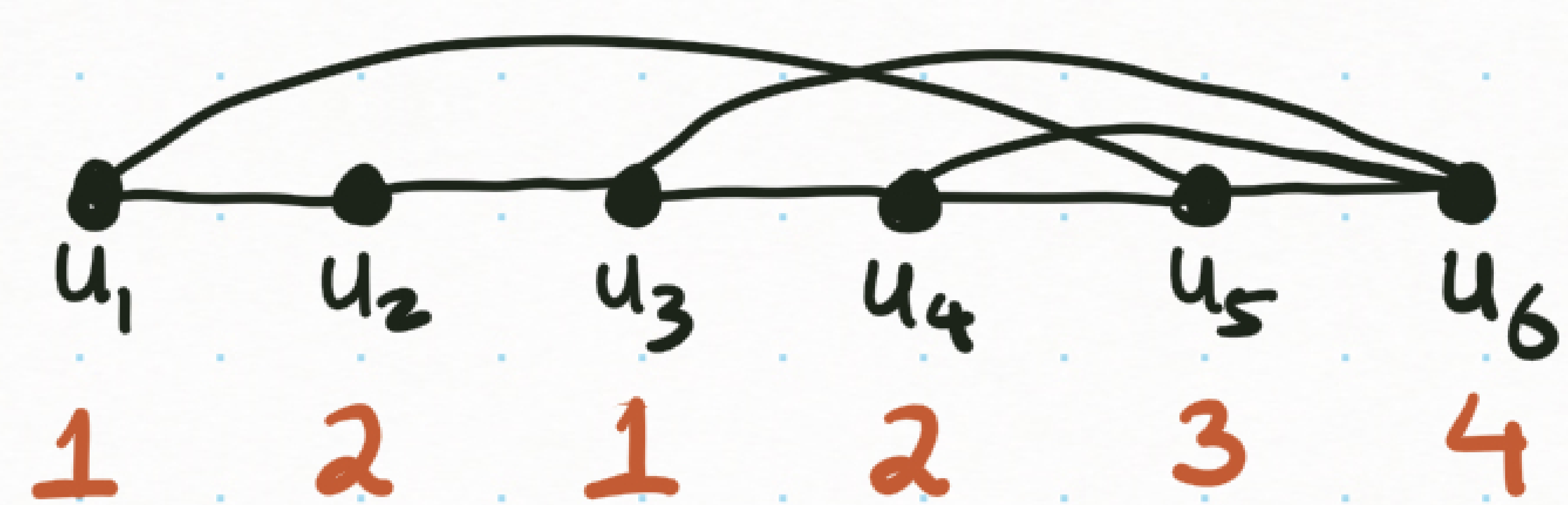
For $i = 1$ to n

Assign the smallest available feasible color to v_i

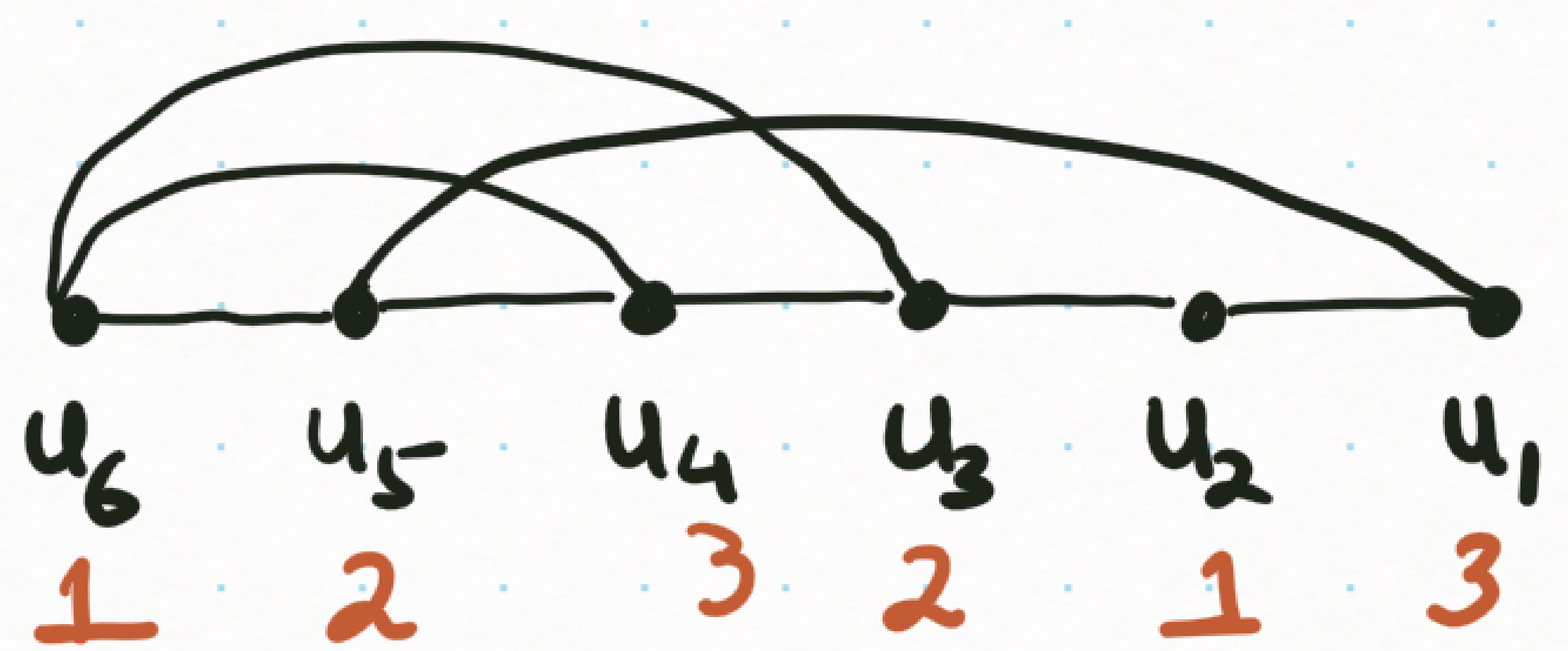
smallest color that has not already been used on any of the neighbors of v_i



G



$\Rightarrow \chi(G) \leq 4$



$\Rightarrow \chi(G) \leq 3$

which is the best since $K_3 \subseteq G \Rightarrow 3 \leq \chi(G)$

Monitoring a network

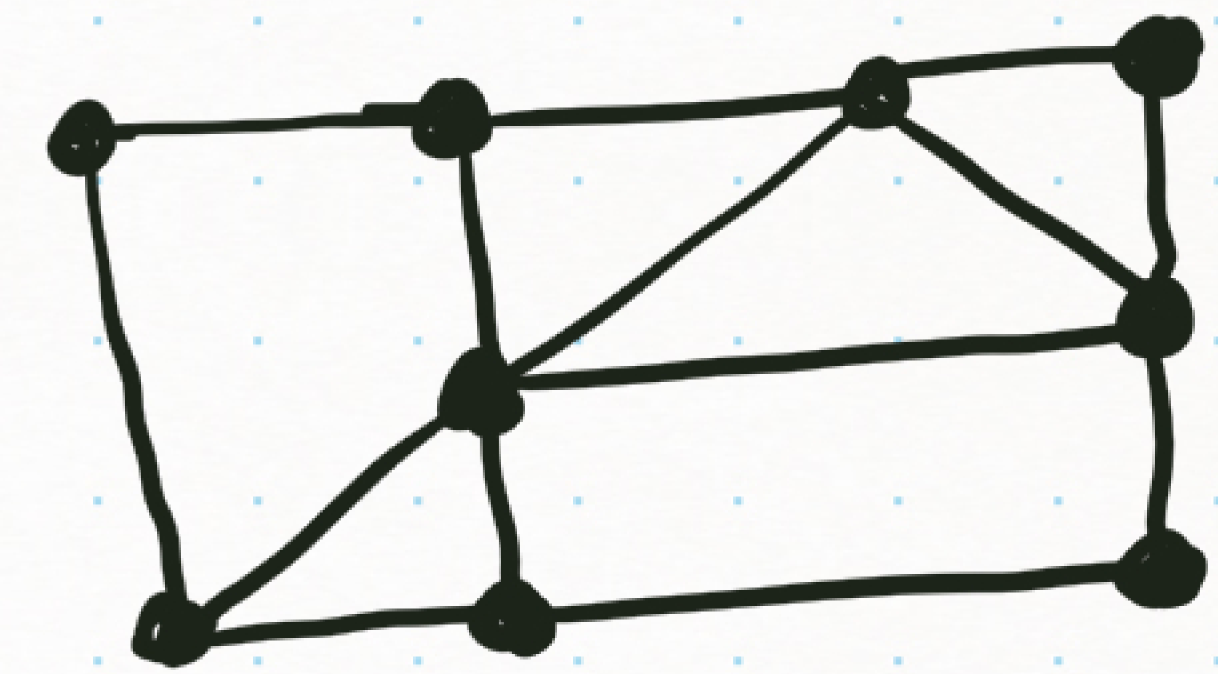
CPD wants to position police officer at road intersections so that every road segment is visible to at least one police officer.

One officer at each intersection!! Expensive! Can we do better?

Let us model the road network in the city as:

Vertices are road intersections

Edges are road segments joining the intersections.



Monitoring a network

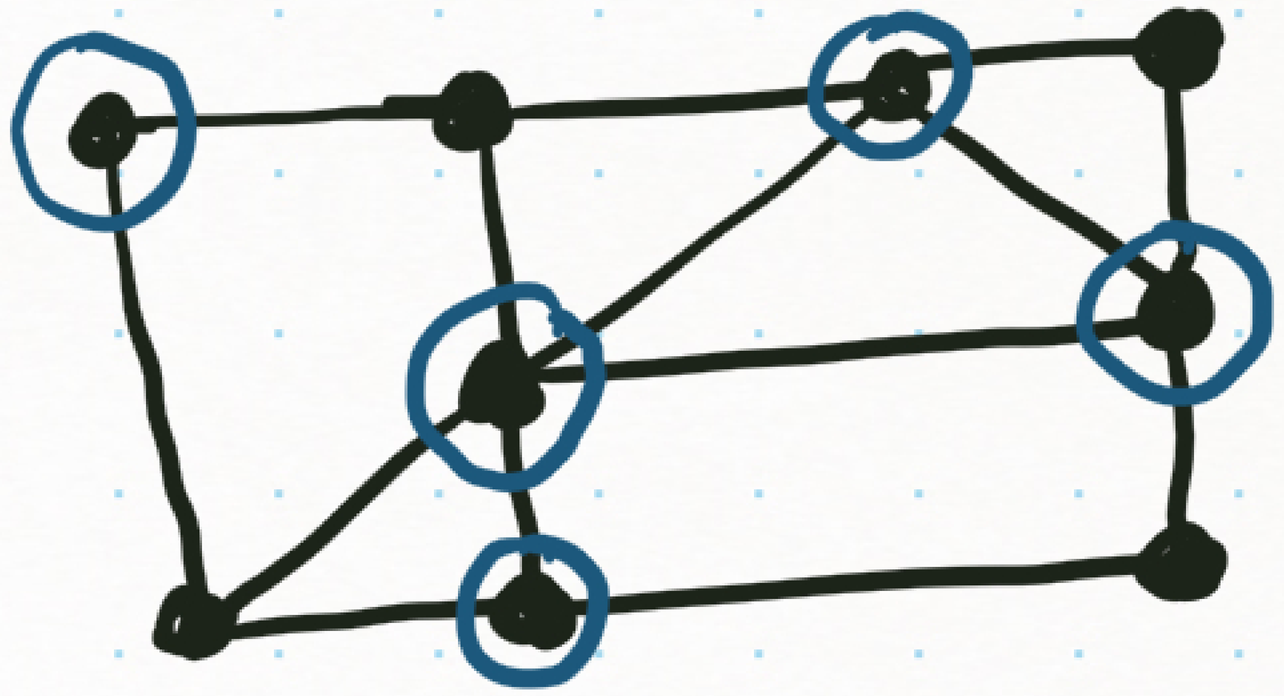
CPD wants to position police officer at road intersections so that every road segment is visible to at least one police officer.

One officer at each intersection!! Expensive! Can we do better?

Let us model the road network in the city as:

Vertices are road intersections

Edges are road segments joining the intersections.



Vertex Cover Problem Given a graph $G=(V,E)$ we seek a subset $S \subseteq V(G)$ such that every edge in G is incident with at least one vertex in S.

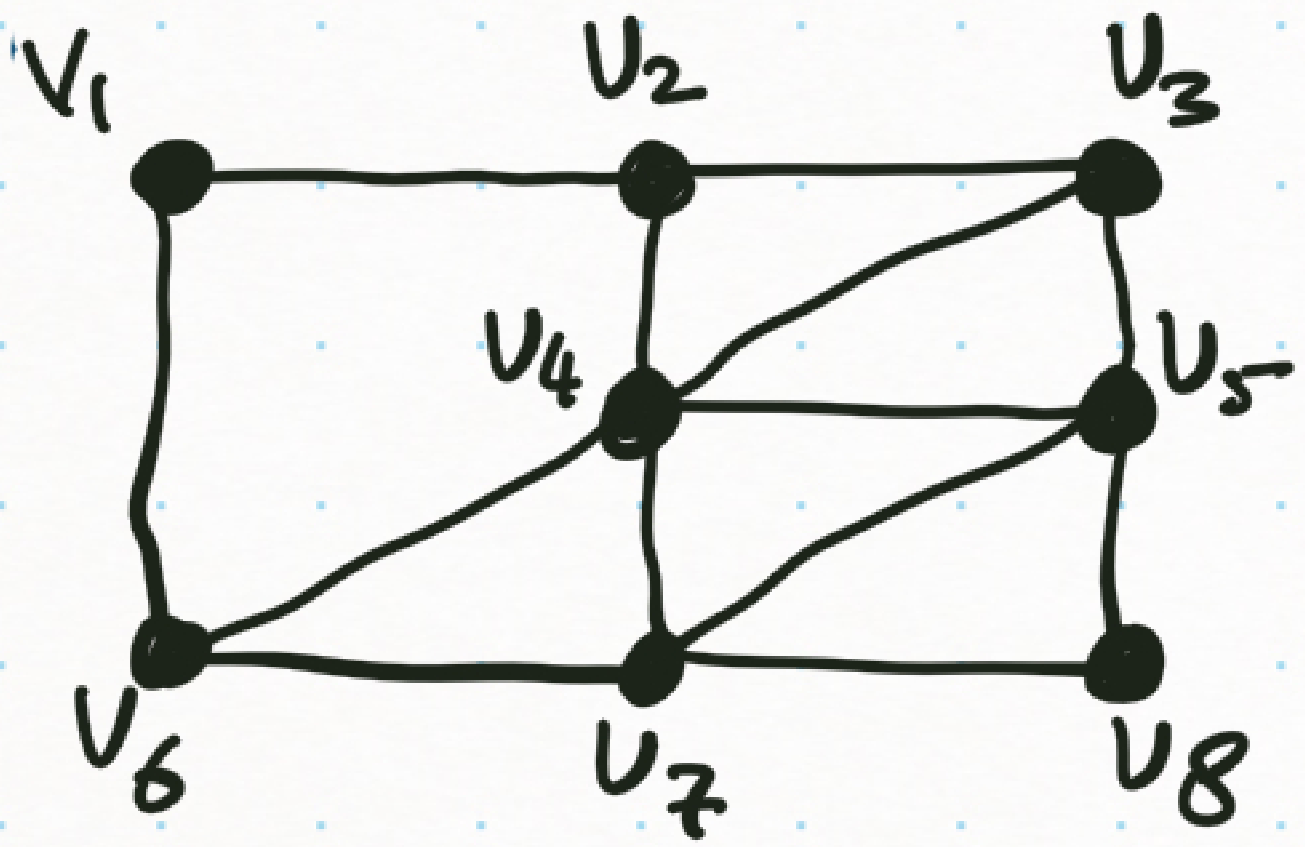
Such an S is called a vertex cover of G

& we want find S with minimal $|S|$: $\beta(G) = \min\{|S| : S \text{ is a vertex cover of } G\}$

How would you design a greedy algorithm for finding a vertex cover of G?

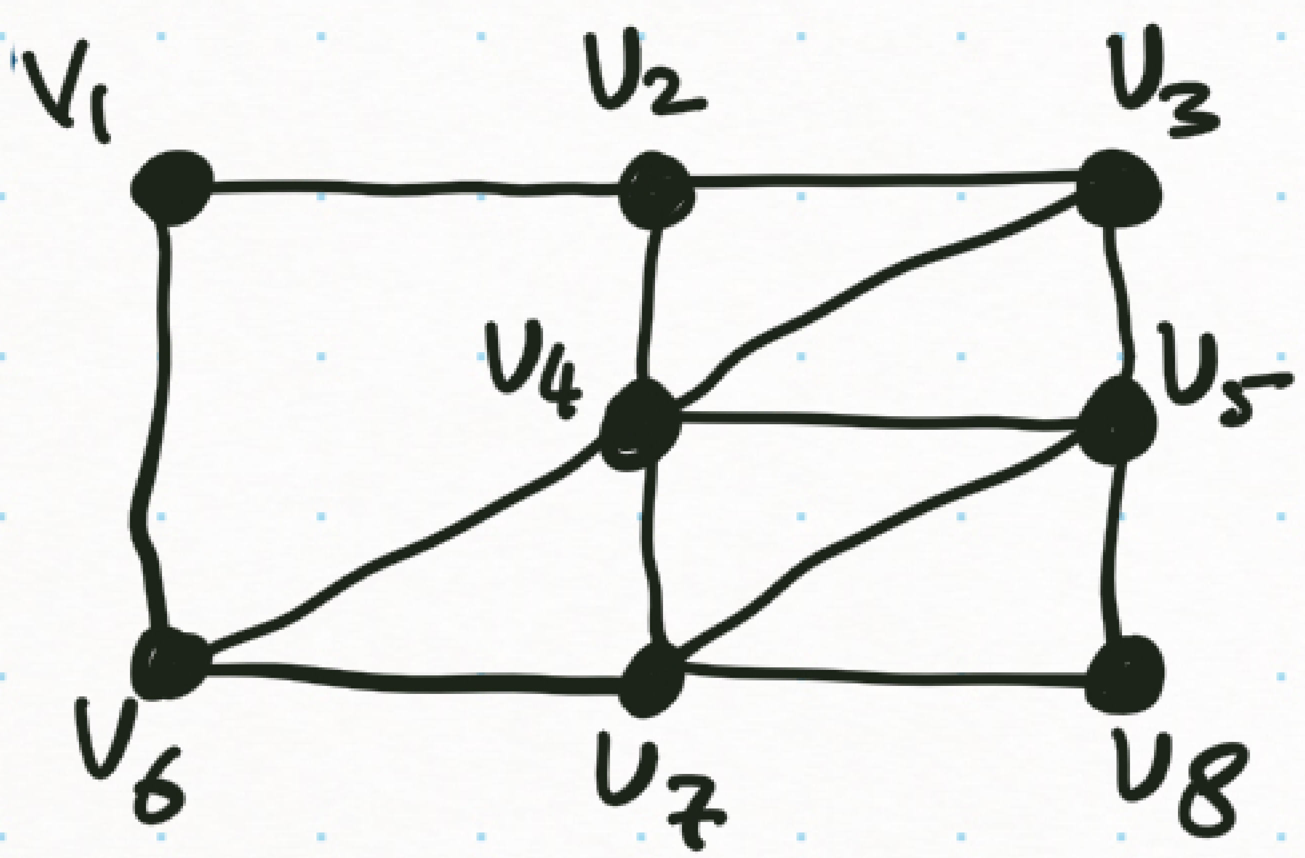
Process V(G), one vertex at a time ← But in what order? What greedy criterion?

Pick a vertex which "takes care" of most edges in that step.



How would you design a greedy algorithm for finding a vertex cover of G?

Process V(G), one vertex at a time ← But in what order? what greedy criterion?

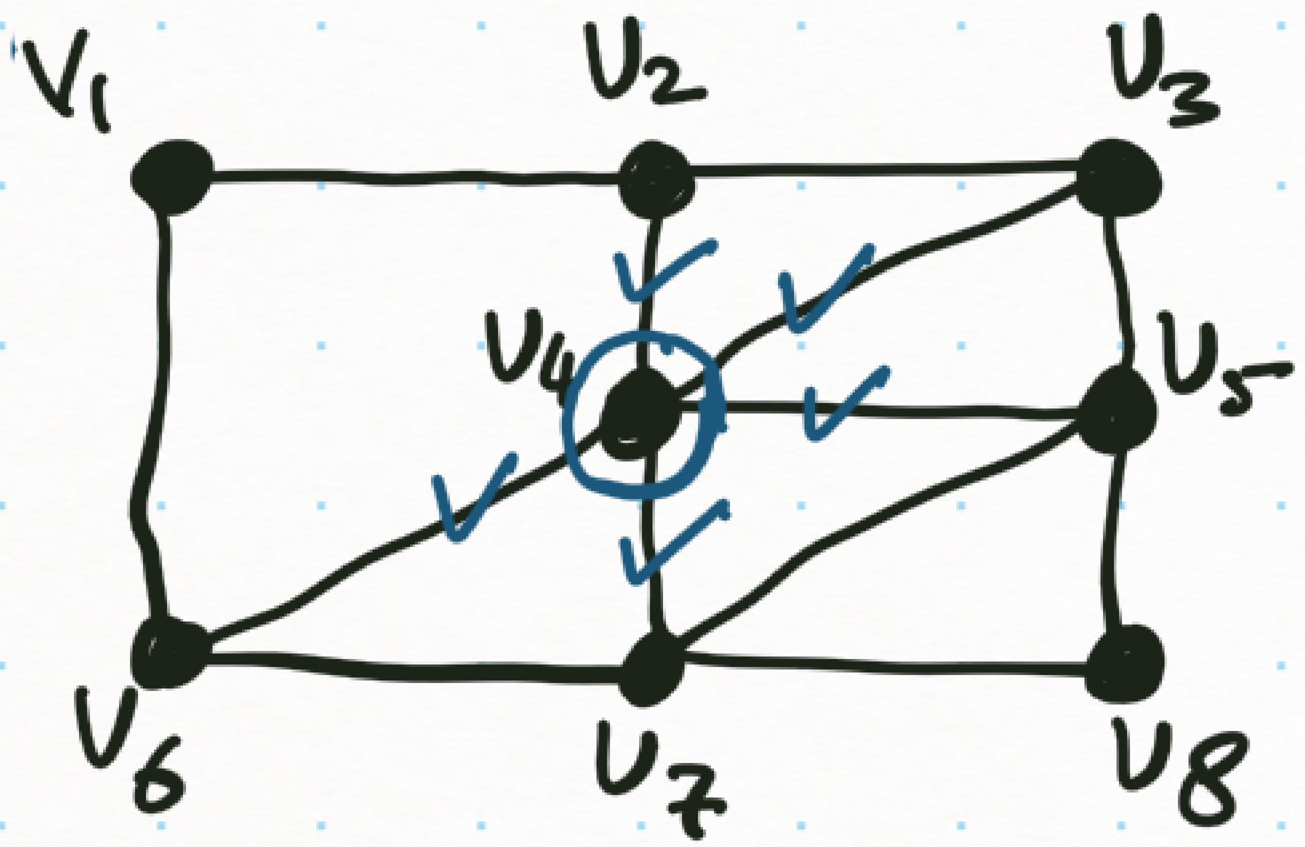


highest degree vertex in the remaining graph ← Pick a vertex which "takes care" of most edges in that step.

How would you design a greedy algorithm for finding a vertex cover of G?

Process V(G), one vertex at a time ← But in what order? what greedy criterion?

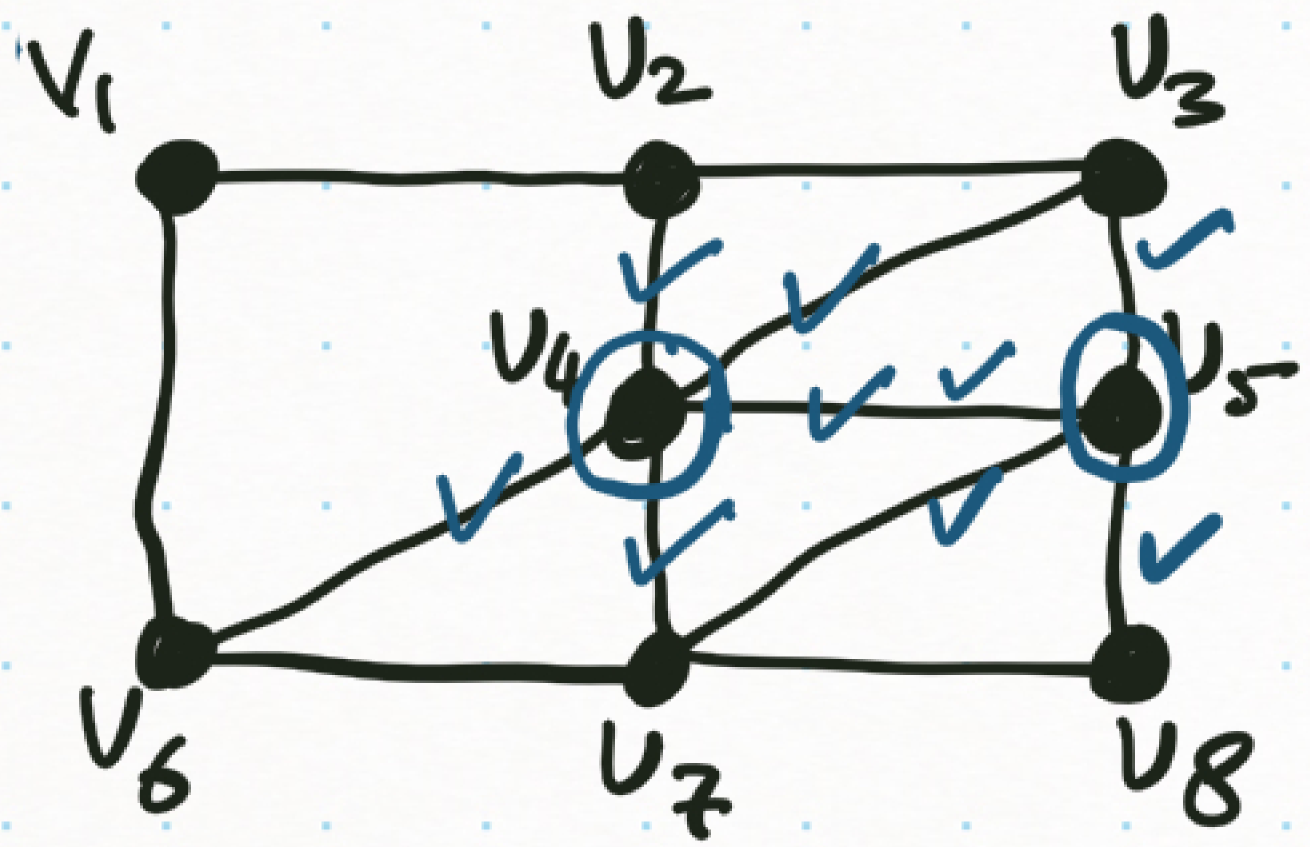
Pick a vertex which "takes care" of most edges in that step. → highest degree vertex in the remaining graph



Pick v_4

How would you design a greedy algorithm for finding a vertex cover of G?

Process V(G), one vertex at a time ← But in what order? what greedy criterion?



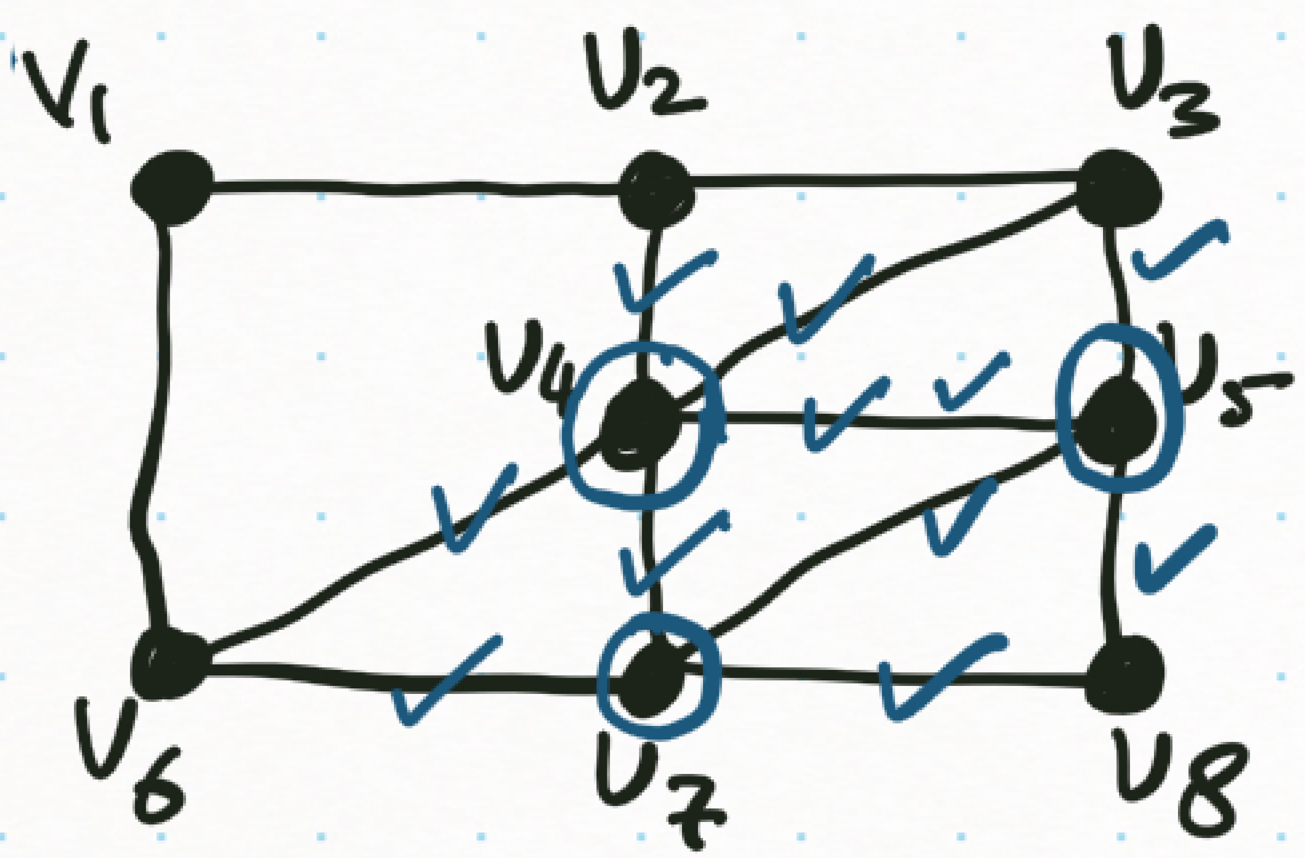
highest degree vertex in the remaining graph ← Pick a vertex which "takes care" of most edges in that step.

Pick v_4

v_5

How would you design a greedy algorithm for finding a vertex cover of G?

Process V(G), one vertex at a time ← But in what order? what greedy criterion?



highest degree vertex in the remaining graph ← [Pick a vertex which "takes care" of most edges in that step.

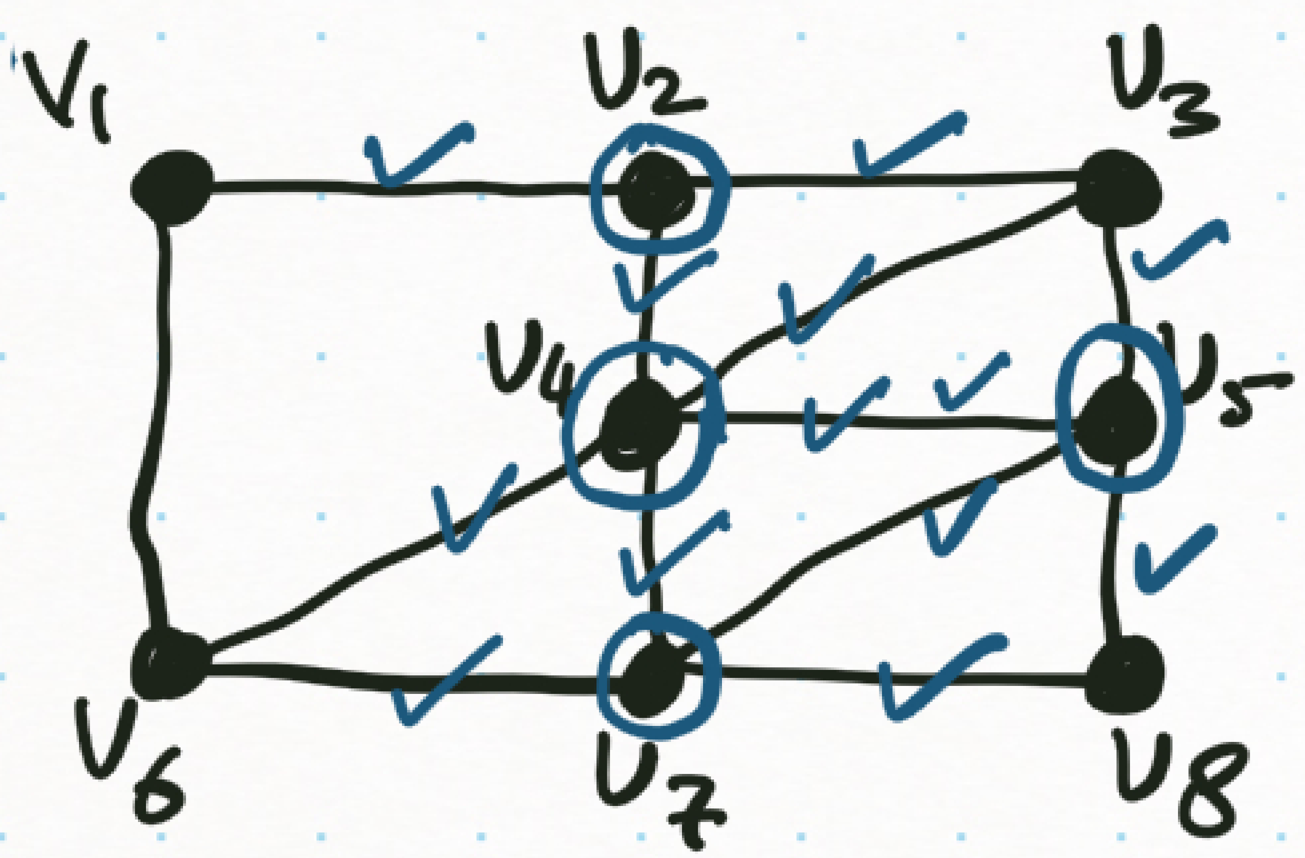
Pick v_4

v_5

v_7

How would you design a greedy algorithm for finding a vertex cover of G?

Process V(G), one vertex at a time ← But in what order? What greedy criterion?



highest degree vertex in the remaining graph ← Pick a vertex which "takes care" of most edges in that step.

Pick v_4

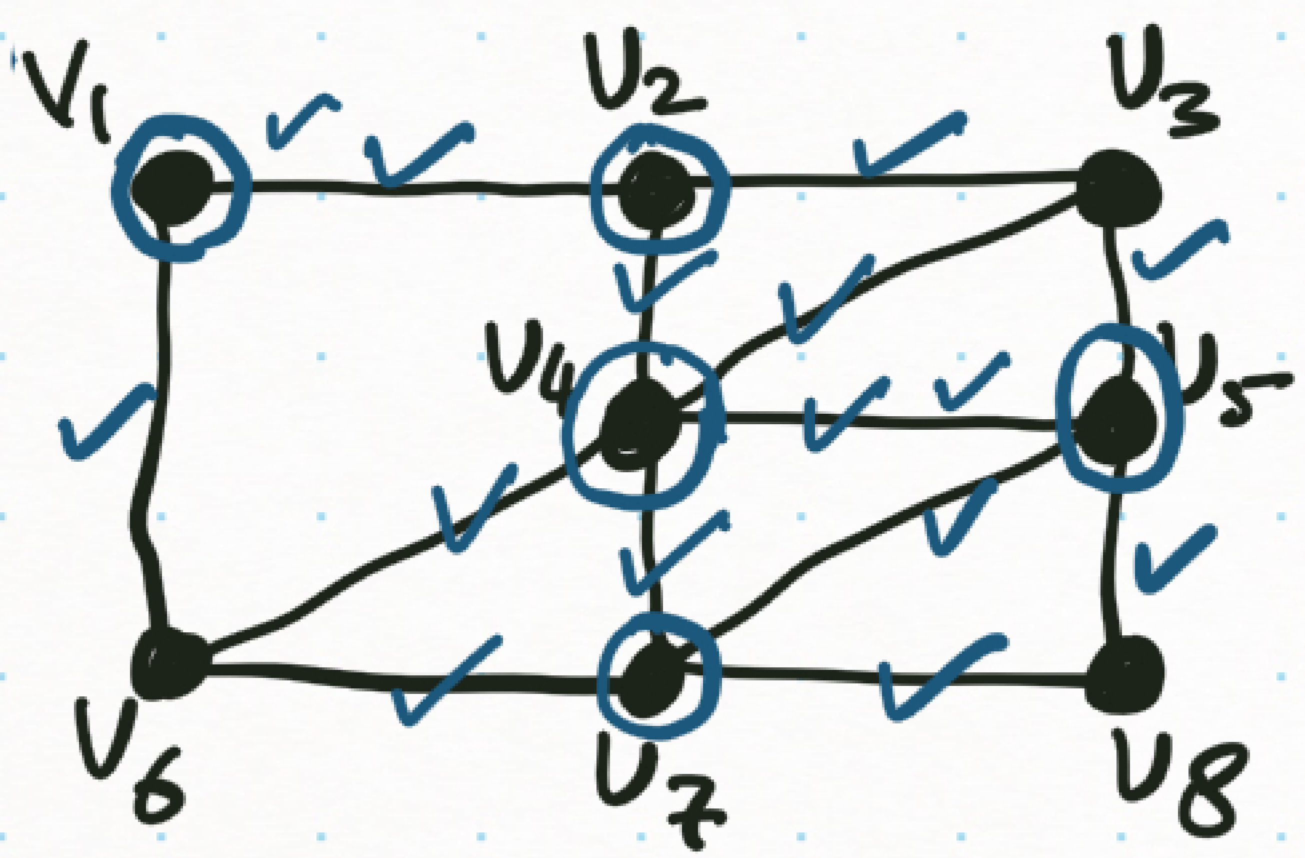
v_5

v_7

v_2

How would you design a greedy algorithm for finding a vertex cover of G?

Process V(G), one vertex at a time ← But in what order? what greedy criterion?



highest degree vertex in the remaining graph ← Pick a vertex which "takes care" of most edges in that step.

Pick
v_4
v_5
v_7
v_2
v_1
} vertex cover

In general graphs, this greedy can give poor solutions.

Vertex cover problem as an optimization problem

Let $G = (V(G), E(G))$ be the given graph.

Suppose $V(G) = \{v_1, v_2, \ldots, v_n\}$.

Vertex cover problem as an optimization problem

Let $G = (V(G), E(G))$ be the given graph.

Suppose $V(G) = \{v_1, v_2, \ldots, v_n\}$.

For each $v_i \in V(G)$, we have to decide whether or not to include it in our set S, vertex cover.

Let $x_i = \begin{cases} 1 & \text{if } v_i \in S \\ 0 & \text{if } v_i \notin S \end{cases}$

Using the variables x_i, how will you model the requirement that each edge must be incident to at least one vertex in S?

Vertex cover problem as an optimization problem

Let $G = (V(G), E(G))$ be the given graph.

Suppose $V(G) = \{v_1, v_2, \ldots, v_n\}$.

For each $v_i \in V(G)$, we have to decide whether or not to include it in our set S, vertex cover.

Let $x_i = \begin{cases} 1 & \text{if } v_i \in S \\ 0 & \text{if } v_i \notin S \end{cases}$

Using the variables x_i, how will you model the requirement that each edge must be incident to at least one vertex in S?

$$x_i + x_j \geq 1 \quad \text{for all } v_iv_j \in E(G)$$

Objective function?

Vertex cover problem as an optimization problem

Let $G = (V(G), E(G))$ be the given graph.

Suppose $V(G) = \{v_1, v_2, \dots, v_n\}$.

For each $v_i \in V(G)$, we have to decide whether or not to include it in our set S, vertex cover.

Let $x_i = \begin{cases} 1 & \text{if } v_i \in S \\ 0 & \text{if } v_i \notin S \end{cases}$ for each $v_i \in V(G)$, $i = 1, 2, \dots, n$.

$$\min \sum_{i=1}^{n} x_i$$

[minimize total # vertices picked]

s.t.

$$x_i + x_j \geq 1 \quad \forall v_i v_j \in E(G)$$

[each edge is "covered" by at least one vertex]

$$x_i \in \{0, 1\} \quad \forall i = 1, \dots, n$$

Vertex cover problem as an optimization problem

Let $G=(V(G),E(G))$ be the given graph.

Suppose $V(G)=\{v_1, v_2, \ldots, v_n\}$.

For each $v_i \in V(G)$, we have to decide whether or not to include it in our set S, vertex cover.

Let $x_i = \begin{cases} 1 & \text{if } v_i \in S \\ 0 & \text{if } v_i \notin S \end{cases}$ for each $v_i \in V(G)$, $i=1,2,\ldots,n$.

$$\min \sum_{i=1}^{n} x_i$$

[minimize total # vertices picked]

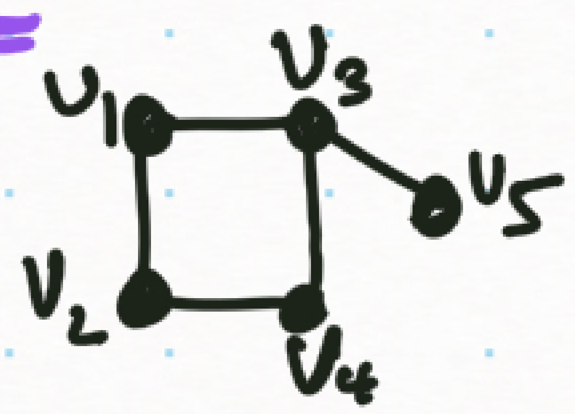
s.t.

$$x_i + x_j \geq 1 \quad \forall v_i v_j \in E(G)$$

[each edge is "covered" by at least one vertex]

$$x_i \in \{0,1\} \quad \forall i = 1, \ldots, n$$

e.g.



min $x_1+x_2+x_3+x_4+x_5$ s.t.

$x_1+x_2 \geq 1$ | $x_2+x_4 \geq 1$ | $x_3+x_5 \geq 1$

$x_1+x_3 \geq 1$ | $x_3+x_4 \geq 1$

$x_1, x_2, x_3, x_4, x_5 \in \{0,1\}$