Randomized Coloring on a Bounded Degree Graph

Christian Wulff-Nilsen
Algorithmic Techniques for Modern Data Models
DTU

October $10, 2025$

Overview for today

- Randomized distributed algorithm

Overview for today

- Randomized distributed algorithm
  - Monte Carlo and Las Vegas algorithms

Overview for today

- Randomized distributed algorithm
  - Monte Carlo and Las Vegas algorithms
- High probability bound

Overview for today

- Randomized distributed algorithm
    - Monte Carlo and Las Vegas algorithms
- High probability bound
- Randomized coloring in bounded-degree graph

Overview for today

- Randomized distributed algorithm
  - Monte Carlo and Las Vegas algorithms
- High probability bound
- Randomized coloring in bounded-degree graph
  - Naive algorithm

Overview for today

- Randomized distributed algorithm
  - Monte Carlo and Las Vegas algorithms
- High probability bound
- Randomized coloring in bounded-degree graph
  - Naive algorithm
  - Refined algorithm

Overview for today

- Randomized distributed algorithm
  - Monte Carlo and Las Vegas algorithms
- High probability bound
- Randomized coloring in bounded-degree graph
  - Naive algorithm
  - Refined algorithm
  - Correctness

Overview for today

- Randomized distributed algorithm
  - Monte Carlo and Las Vegas algorithms
- High probability bound
- Randomized coloring in bounded-degree graph
  - Naive algorithm
  - Refined algorithm
  - Correctness
  - Running time

Randomized Distributed Algorithm

- We consider the PN model (nodes do not have unique identifiers)

Randomized Distributed Algorithm

- We consider the PN model (nodes do not have unique identifiers)
- A *randomized distributed algorithm* (or *randomized PN algorithm*):

Randomized Distributed Algorithm

- We consider the PN model (nodes do not have unique identifiers)
- A *randomized distributed algorithm* (or *randomized PN algorithm*):
  - Each node has access to its own random number generator

Randomized Distributed Algorithm

- We consider the PN model (nodes do not have unique identifiers)
- A *randomized distributed algorithm* (or *randomized PN algorithm*):
  - Each node has access to its own random number generator
  - It thus samples random numbers independently of other nodes

Randomized Distributed Algorithm

- We consider the PN model (nodes do not have unique identifiers)
- A *randomized distributed algorithm* (or *randomized PN algorithm*):
  - Each node has access to its own random number generator
  - It thus samples random numbers independently of other nodes
- Let $n = |V|$

Randomized Distributed Algorithm

- We consider the PN model (nodes do not have unique identifiers)
- A *randomized distributed algorithm* (or *randomized PN algorithm*):
    - Each node has access to its own random number generator
    - It thus samples random numbers independently of other nodes
- Let $n = |V|$
- *Monte Carlo algorithm* with running time $T(n)$ and probability p:

Randomized Distributed Algorithm

- We consider the PN model (nodes do not have unique identifiers)
- A *randomized distributed algorithm* (or *randomized PN algorithm*):
  - Each node has access to its own random number generator
  - It thus samples random numbers independently of other nodes
- Let $n = |V|$
- *Monte Carlo algorithm* with running time $T(n)$ and probability p:
  - always stops in time at most $T(n)$

Randomized Distributed Algorithm

- We consider the PN model (nodes do not have unique identifiers)
- A *randomized distributed algorithm* (or *randomized PN algorithm*):
  - Each node has access to its own random number generator
  - It thus samples random numbers independently of other nodes
- Let $n = |V|$
- *Monte Carlo algorithm* with running time $T(n)$ and probability p:
  - always stops in time at most $T(n)$
  - output is correct with probability at least p

Randomized Distributed Algorithm

- We consider the PN model (nodes do not have unique identifiers)
- A *randomized distributed algorithm* (or *randomized PN algorithm*):
  - Each node has access to its own random number generator
  - It thus samples random numbers independently of other nodes
- Let $n = |V|$
- *Monte Carlo algorithm* with running time $T(n)$ and probability p:
  - always stops in time at most $T(n)$
  - output is correct with probability at least p
- *Las Vegas algorithm* with running time $T(n)$ and probability p:

Randomized Distributed Algorithm

- We consider the PN model (nodes do not have unique identifiers)
- A *randomized distributed algorithm* (or *randomized PN algorithm*):
  - Each node has access to its own random number generator
  - It thus samples random numbers independently of other nodes
- Let $n = |V|$
- *Monte Carlo algorithm* with running time $T(n)$ and probability p:
  - always stops in time at most $T(n)$
  - output is correct with probability at least p
- *Las Vegas algorithm* with running time $T(n)$ and probability p:
  - stops in time at most $T(n)$ with probability at least p

Randomized Distributed Algorithm

- We consider the PN model (nodes do not have unique identifiers)
- A *randomized distributed algorithm* (or *randomized PN algorithm*):
  - Each node has access to its own random number generator
  - It thus samples random numbers independently of other nodes
- Let $n = |V|$
- *Monte Carlo algorithm* with running time $T(n)$ and probability p:
  - always stops in time at most $T(n)$
  - output is correct with probability at least p
- *Las Vegas algorithm* with running time $T(n)$ and probability p:
  - stops in time at most $T(n)$ with probability at least p
  - output is always correct

Randomized Distributed Algorithm

- We consider the PN model (nodes do not have unique identifiers)
- A *randomized distributed algorithm* (or *randomized PN algorithm*):
  - Each node has access to its own random number generator
  - It thus samples random numbers independently of other nodes
- Let $n = |V|$
- *Monte Carlo algorithm* with running time $T(n)$ and probability p:
  - always stops in time at most $T(n)$
  - output is correct with probability at least p
- *Las Vegas algorithm* with running time $T(n)$ and probability p:
  - stops in time at most $T(n)$ with probability at least p
  - output is always correct
- We focus on Las Vegas algorithms in the following

High Probability Bound

- A Las Vegas algorithm is said to stop in time $O(T(n))$ *with high probability (w.h.p.)* if the probability p is of the form

$$p = 1 - 1/n^c$$

  for any chosen constant $c > 0$

High Probability Bound

- A Las Vegas algorithm is said to stop in time $O(T(n))$ *with high probability (w.h.p.)* if the probability p is of the form

$$p = 1 - 1/n^c$$

for any chosen constant $c > 0$
- The constant hidden in $O(T(n))$ depends on c

High Probability Bound

- A Las Vegas algorithm is said to stop in time $O(T(n))$ *with high probability (w.h.p.)* if the probability p is of the form

$$p = 1 - 1/n^c$$

  for any chosen constant $c > 0$
- The constant hidden in $O(T(n))$ depends on c
- Here, “time” means “number of rounds”

Randomized coloring in bounded-degree graph

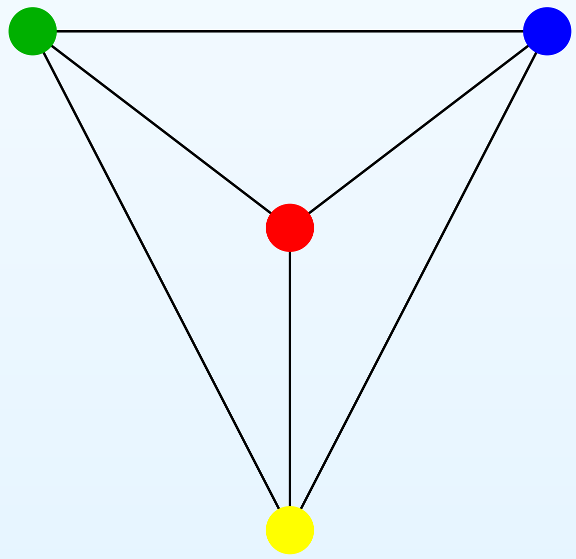
- Problem:
  - Given an n-node graph G of max degree Δ

Randomized coloring in bounded-degree graph

- Problem:
  - Given an n-node graph G of max degree Δ
  - Output a $(\Delta + 1)$-coloring in G

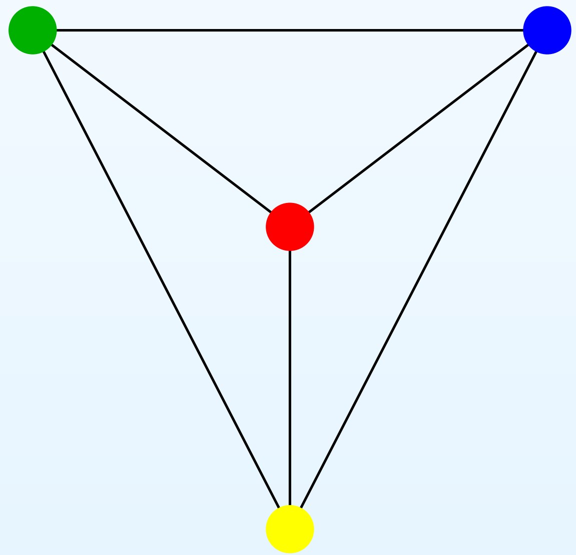
Randomized coloring in bounded-degree graph

- Problem:
  - Given an n-node graph G of max degree Δ
  - Output a $(\Delta + 1)$-coloring in G
- Example with $\Delta = 3$:



Randomized coloring in bounded-degree graph

- Problem:
  - Given an n-node graph G of max degree Δ
  - Output a $(\Delta + 1)$-coloring in G
- Example with $\Delta = 3$:



- We will present a Las Vegas distributed algorithm running in $O(\log n)$ rounds w.h.p.

Naive Algorithm

- Algorithm for a node u:

Naive Algorithm

- Algorithm for a node u:
  - Pick a color $c(u)$ uniformly at random from $\{1, \ldots, \Delta + 1\}$

Naive Algorithm

- Algorithm for a node u:
  - Pick a color $c(u)$ uniformly at random from $\{1, \ldots, \Delta + 1\}$
  - Send $c(u)$ to neighbors

Naive Algorithm

- Algorithm for a node u:
  - Pick a color $c(u)$ uniformly at random from $\{1, \ldots, \Delta + 1\}$
  - Send $c(u)$ to neighbors
  - If $c(u)$ conflicts with colors received by neighbors, repeat

Naive Algorithm

- Algorithm for a node u:
  - Pick a color $c(u)$ uniformly at random from $\{1, \ldots, \Delta + 1\}$
  - Send $c(u)$ to neighbors
  - If $c(u)$ conflicts with colors received by neighbors, repeat
  - Otherwise, keep $c(u)$ and stop

Naive Algorithm

- Algorithm for a node u:
  - Pick a color $c(u)$ uniformly at random from $\{1, \ldots, \Delta + 1\}$
  - Send $c(u)$ to neighbors
  - If $c(u)$ conflicts with colors received by neighbors, repeat
  - Otherwise, keep $c(u)$ and stop
- This is the algorithm for paths (Section 1.5) where $\Delta = 2$ which stopped in $O(\log n)$ rounds w.h.p.

Naive Algorithm

- Algorithm for a node u:
  - Pick a color $c(u)$ uniformly at random from $\{1, \ldots, \Delta + 1\}$
  - Send $c(u)$ to neighbors
  - If $c(u)$ conflicts with colors received by neighbors, repeat
  - Otherwise, keep $c(u)$ and stop
- This is the algorithm for paths (Section 1.5) where $\Delta = 2$ which stopped in $O(\log n)$ rounds w.h.p.
- For larger values of Δ:

Naive Algorithm

- Algorithm for a node u:
  - Pick a color $c(u)$ uniformly at random from $\{1, \ldots, \Delta + 1\}$
  - Send $c(u)$ to neighbors
  - If $c(u)$ conflicts with colors received by neighbors, repeat
  - Otherwise, keep $c(u)$ and stop
- This is the algorithm for paths (Section 1.5) where $\Delta = 2$ which stopped in $O(\log n)$ rounds w.h.p.
- For larger values of Δ:
  - The naive algorithm will be too slow

Naive Algorithm

- Algorithm for a node u:
  - Pick a color $c(u)$ uniformly at random from $\{1, \ldots, \Delta + 1\}$
  - Send $c(u)$ to neighbors
  - If $c(u)$ conflicts with colors received by neighbors, repeat
  - Otherwise, keep $c(u)$ and stop
- This is the algorithm for paths (Section 1.5) where $\Delta = 2$ which stopped in $O(\log n)$ rounds w.h.p.
- For larger values of Δ:
  - The naive algorithm will be too slow
  - We will present a refined version of it

Naive Algorithm

- Algorithm for a node u:
  - Pick a color $c(u)$ uniformly at random from $\{1, \ldots, \Delta + 1\}$
  - Send $c(u)$ to neighbors
  - If $c(u)$ conflicts with colors received by neighbors, repeat
  - Otherwise, keep $c(u)$ and stop
- This is the algorithm for paths (Section 1.5) where $\Delta = 2$ which stopped in $O(\log n)$ rounds w.h.p.
- For larger values of Δ:
  - The naive algorithm will be too slow
  - We will present a refined version of it
  - We will show that its time bound is $O(\log n)$ w.h.p.

Refined Algorithm: Information Maintained at Node u

- Define the *color palette* of $u \in V$ as

$$C(u) = \{1, 2, \ldots, \deg_G(u) + 1\}$$

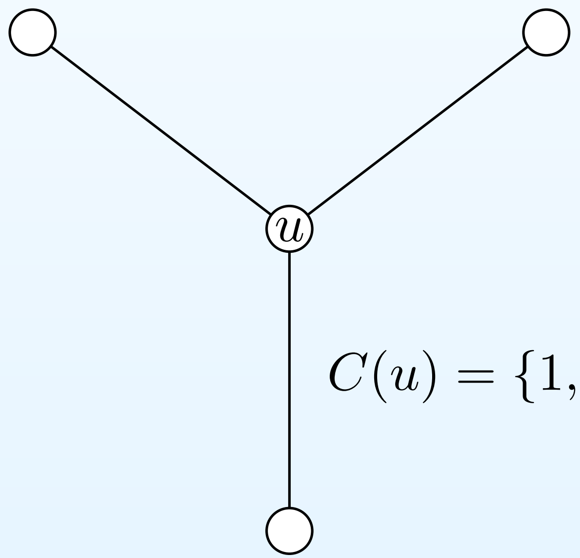
where $\deg_G(u)$ is the degree of u

Refined Algorithm: Information Maintained at Node u

- Define the *color palette* of $u \in V$ as

$$C(u) = \{1, 2, \ldots, \deg_G(u) + 1\}$$

where $\deg_G(u)$ is the degree of u



$C(u) = \{1, 2, 3, 4\}$

Refined Algorithm: Information Maintained at Node u

- Define the *color palette* of $u \in V$ as

$$C(u) = \{1, 2, \ldots, \deg_G(u) + 1\}$$

  where $\deg_G(u)$ is the degree of u
- Each node u maintains:

Refined Algorithm: Information Maintained at Node u

- Define the *color palette* of $u \in V$ as

$$C(u) = \{1, 2, \ldots, \deg_G(u) + 1\}$$

  where $\deg_G(u)$ is the degree of u
- Each node u maintains:
  - State $s(u) \in \{0, 1\}$

Refined Algorithm: Information Maintained at Node u

- Define the *color palette* of $u \in V$ as

$$C(u) = \{1, 2, \ldots, \deg_G(u) + 1\}$$

  where $\deg_G(u)$ is the degree of u
- Each node u maintains:
  - State $s(u) \in \{0, 1\}$
  - Color $c(u) \in \{\perp\} \cup C(u)$

Refined Algorithm: Information Maintained at Node u

- Define the *color palette* of $u \in V$ as

$$C(u) = \{1, 2, \ldots, \deg_G(u) + 1\}$$

  where $\deg_G(u)$ is the degree of u
- Each node u maintains:
  - State $s(u) \in \{0, 1\}$
  - Color $c(u) \in \{\perp\} \cup C(u)$
  - We interpret $c(u) = \perp$ as u not having been assigned a color yet

Refined Algorithm: Information Maintained at Node u

- Define the *color palette* of $u \in V$ as

$$C(u) = \{1, 2, \ldots, \deg_G(u) + 1\}$$

  where $\deg_G(u)$ is the degree of u
- Each node u maintains:
  - State $s(u) \in \{0, 1\}$
  - Color $c(u) \in \{\perp\} \cup C(u)$
  - We interpret $c(u) = \perp$ as u not having been assigned a color yet
- At termination, the $c(u)$-colors will form a valid $(\Delta + 1)$-coloring

Algorithm for Node u

- Round 0: $s(u) \leftarrow 1, c(u) \leftarrow \perp$

Algorithm for Node u

- Round 0: $s(u) \leftarrow 1, c(u) \leftarrow \perp$
- u sends $c(u)$ to its neighbors at the start of every following round

Algorithm for Node u

- Round 0: $s(u) \leftarrow 1, c(u) \leftarrow \perp$
- u sends $c(u)$ to its neighbors at the start of every following round
- Until u stops, it alternates between states 1 and 0:

Algorithm for Node u

- Round 0: $s(u) \leftarrow 1, c(u) \leftarrow \perp$
- u sends $c(u)$ to its neighbors at the start of every following round
- Until u stops, it alternates between states 1 and 0:
  - In odd rounds $(1, 3, 5, \ldots)$, $s(u)$ starts as 1 and ends as 0

Algorithm for Node u

- Round 0: $s(u) \leftarrow 1, c(u) \leftarrow \perp$
- u sends $c(u)$ to its neighbors at the start of every following round
- Until u stops, it alternates between states 1 and 0:
  - In odd rounds $(1, 3, 5, \ldots)$, $s(u)$ starts as 1 and ends as 0
  - In even rounds $(2, 4, 6, \ldots)$, $s(u)$ starts as 0 and ends as 1

Algorithm for Node u

- Round 0: $s(u) \leftarrow 1, c(u) \leftarrow \perp$
- u sends $c(u)$ to its neighbors at the start of every following round
- Until u stops, it alternates between states 1 and 0:
  - In odd rounds $(1, 3, 5, \ldots)$, $s(u)$ starts as 1 and ends as 0
  - In even rounds $(2, 4, 6, \ldots)$, $s(u)$ starts as 0 and ends as 1
- u stops when $s(u) = 1$ and $c(u) \neq \perp$

Algorithm for Node u

- Round 0: $s(u) \leftarrow 1, c(u) \leftarrow \perp$
- u sends $c(u)$ to its neighbors at the start of every following round
- Until u stops, it alternates between states 1 and 0:
  - In odd rounds $(1, 3, 5, \ldots)$, $s(u)$ starts as 1 and ends as 0
  - In even rounds $(2, 4, 6, \ldots)$, $s(u)$ starts as 0 and ends as 1
- u stops when $s(u) = 1$ and $c(u) \neq \perp$
- $s(u)$ thus keeps track of the parity of rounds and if u has stopped

Algorithm for Node u

- Round 0: $s(u) \leftarrow 1, c(u) \leftarrow \perp$
- u sends $c(u)$ to its neighbors at the start of every following round
- Until u stops, it alternates between states 1 and 0:
  - In odd rounds $(1, 3, 5, \ldots)$, $s(u)$ starts as 1 and ends as 0
  - In even rounds $(2, 4, 6, \ldots)$, $s(u)$ starts as 0 and ends as 1
- u stops when $s(u) = 1$ and $c(u) \neq \perp$
- $s(u)$ thus keeps track of the parity of rounds and if u has stopped
- u still sends $c(u)$ to its neighbors after stopping

Algorithm for Node u: Odd Round $(1, 3, 5, \ldots)$

Algorithm for Node u: Odd Round $(1, 3, 5, \ldots)$

- Send $c(u)$ to neighbors

Algorithm for Node u: Odd Round $(1, 3, 5, \ldots)$

- Send $c(u)$ to neighbors
- Assume u has not stopped, that is: $c(u) = \perp$

Algorithm for Node u: Odd Round $(1, 3, 5, \dots)$

- Send $c(u)$ to neighbors
- Assume u has not stopped, that is: $c(u) = \perp$
- $M(u)$: messages (colors) received by neighbors

Algorithm for Node u: Odd Round $(1, 3, 5, \ldots)$

- Send $c(u)$ to neighbors
- Assume u has not stopped, that is: $c(u) = \perp$
- $M(u)$: messages (colors) received by neighbors
- $F(u) = C(u) \setminus M(u)$: *free* colors (currently not used by neighbors)

Algorithm for Node u: Odd Round $(1, 3, 5, \ldots)$

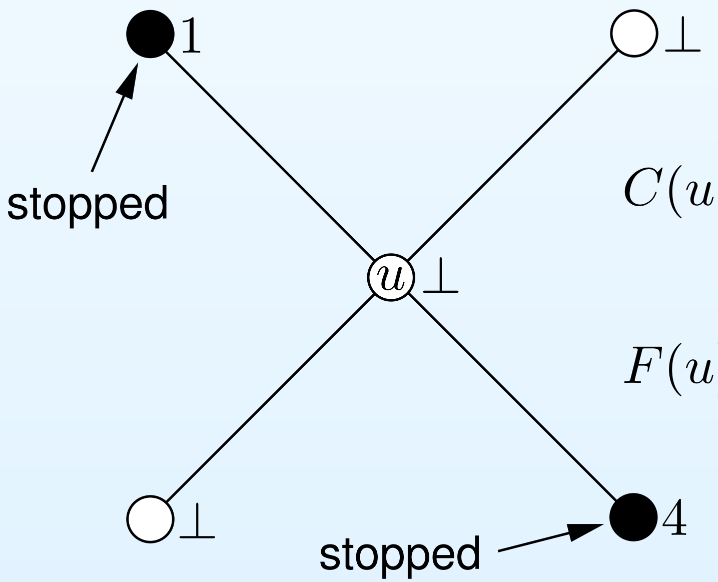
- Send $c(u)$ to neighbors
- Assume u has not stopped, that is: $c(u) = \perp$
- $M(u)$: messages (colors) received by neighbors
- $F(u) = C(u) \setminus M(u)$: *free* colors (currently not used by neighbors)
- With probability $1/2$, pick $c(u)$ from $F(u)$ uniformly at random (otherwise, $c(u)$ remains $\perp$)

Algorithm for Node u: Odd Round $(1, 3, 5, \ldots)$

- Send $c(u)$ to neighbors
- Assume u has not stopped, that is: $c(u) = \perp$
- $M(u)$: messages (colors) received by neighbors
- $F(u) = C(u) \setminus M(u)$: *free* colors (currently not used by neighbors)
- With probability $1/2$, pick $c(u)$ from $F(u)$ uniformly at random (otherwise, $c(u)$ remains $\perp$)
- Switch state: $s(u) \leftarrow 0$

Algorithm for Node u: Odd Round $(1, 3, 5, \ldots)$

- Send $c(u)$ to neighbors
- Assume u has not stopped, that is: $c(u) = \perp$
- $M(u)$: messages (colors) received by neighbors
- $F(u) = C(u) \setminus M(u)$: *free* colors (currently not used by neighbors)
- With probability $1/2$, pick $c(u)$ from $F(u)$ uniformly at random (otherwise, $c(u)$ remains $\perp$)
- Switch state: $s(u) \leftarrow 0$

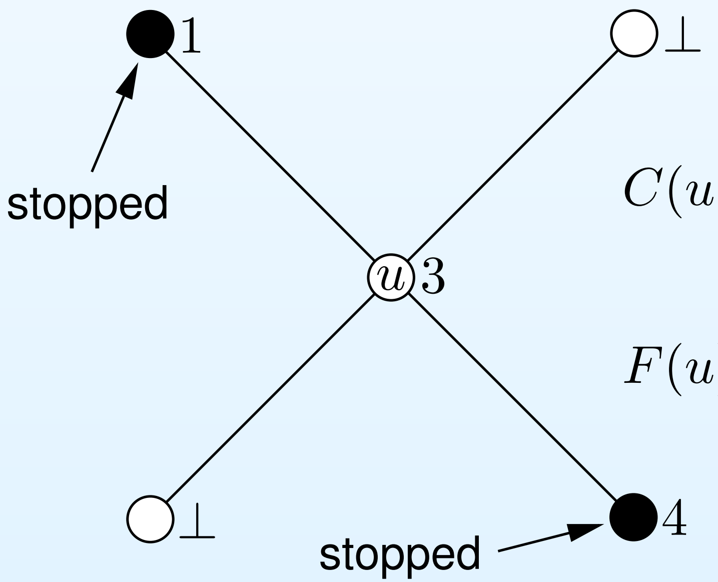


$C(u) = \{1, 2, 3, 4, 5\}$

$F(u) = \{2, 3, 5\}$

Algorithm for Node u: Odd Round $(1, 3, 5, \ldots)$

- Send $c(u)$ to neighbors
- Assume u has not stopped, that is: $c(u) = \perp$
- $M(u)$: messages (colors) received by neighbors
- $F(u) = C(u) \setminus M(u)$: *free* colors (currently not used by neighbors)
- With probability $1/2$, pick $c(u)$ from $F(u)$ uniformly at random (otherwise, $c(u)$ remains $\perp$)
- Switch state: $s(u) \leftarrow 0$



$C(u) = \{1, 2, 3, 4, 5\}$

$F(u) = \{2, 3, 5\}$

Algorithm for Node u: Even Round $(2, 4, 6, \ldots)$

Algorithm for Node u: Even Round $(2, 4, 6, \ldots)$

- Send $c(u)$ to neighbors

Algorithm for Node u: Even Round $(2, 4, 6, \ldots)$

- Send $c(u)$ to neighbors
- $M(u)$: messages (colors) received by neighbors

Algorithm for Node u: Even Round $(2, 4, 6, \ldots)$

- Send $c(u)$ to neighbors
- $M(u)$: messages (colors) received by neighbors
- If $c(u) \in M(u)$ (a conflict), set $c(u) \leftarrow \perp$

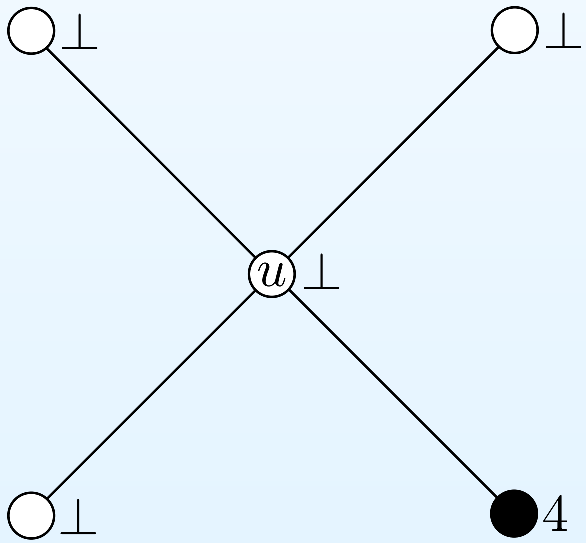
Algorithm for Node u: Even Round $(2, 4, 6, \ldots)$

- Send $c(u)$ to neighbors
- $M(u)$: messages (colors) received by neighbors
- If $c(u) \in M(u)$ (a conflict), set $c(u) \leftarrow \perp$
- Switch state: $s(u) \leftarrow 1$

Algorithm for Node u: Even Round $(2, 4, 6, \ldots)$

- Send $c(u)$ to neighbors
- $M(u)$: messages (colors) received by neighbors
- If $c(u) \in M(u)$ (a conflict), set $c(u) \leftarrow \perp$
- Switch state: $s(u) \leftarrow 1$

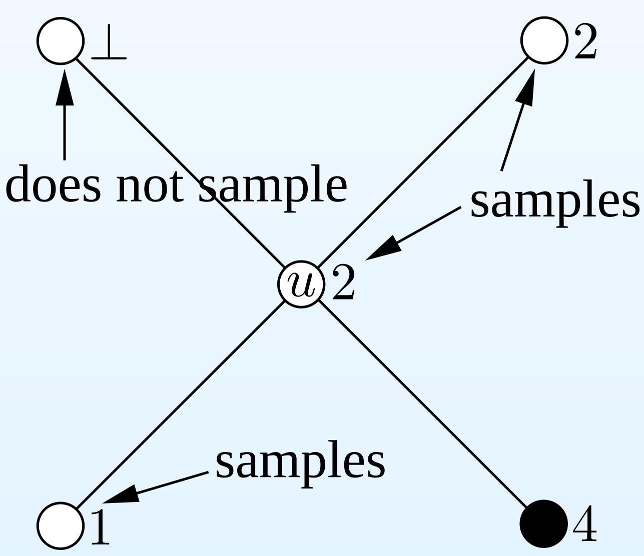
Odd round i:



Algorithm for Node u: Even Round $(2, 4, 6, \ldots)$

- Send $c(u)$ to neighbors
- $M(u)$: messages (colors) received by neighbors
- If $c(u) \in M(u)$ (a conflict), set $c(u) \leftarrow \perp$
- Switch state: $s(u) \leftarrow 1$

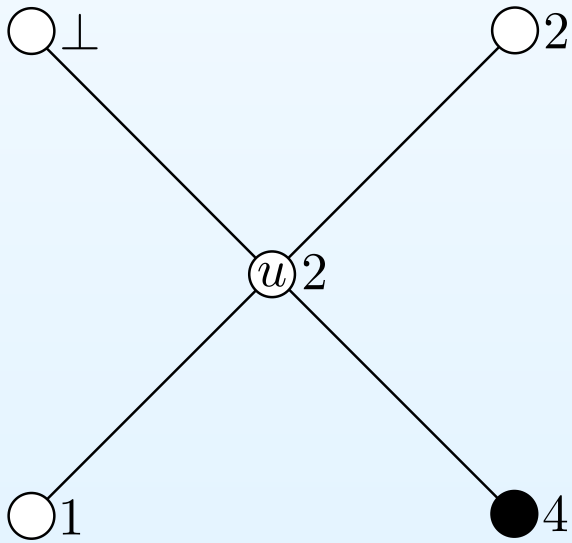
Odd round i:



Algorithm for Node u: Even Round $(2, 4, 6, \ldots)$

- Send $c(u)$ to neighbors
- $M(u)$: messages (colors) received by neighbors
- If $c(u) \in M(u)$ (a conflict), set $c(u) \leftarrow \perp$
- Switch state: $s(u) \leftarrow 1$

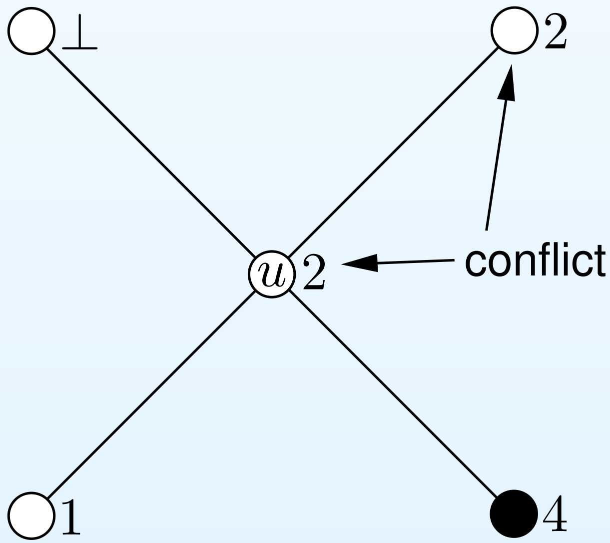
Even round $i + 1$:



Algorithm for Node u: Even Round $(2, 4, 6, \ldots)$

- Send $c(u)$ to neighbors
- $M(u)$: messages (colors) received by neighbors
- If $c(u) \in M(u)$ (a conflict), set $c(u) \leftarrow \perp$
- Switch state: $s(u) \leftarrow 1$

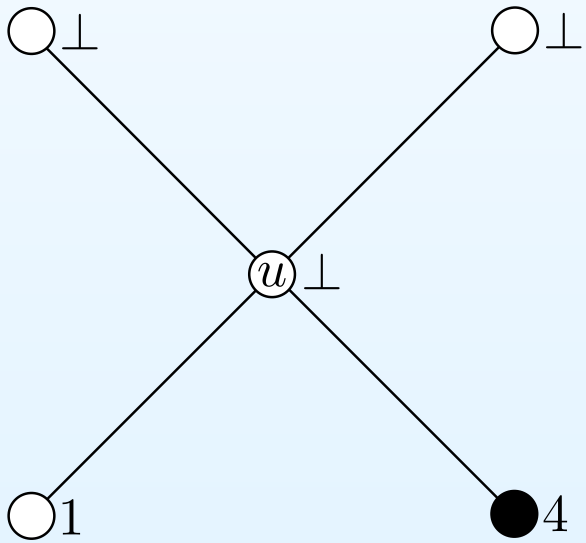
Even round $i+1$:



Algorithm for Node u: Even Round $(2, 4, 6, \ldots)$

- Send $c(u)$ to neighbors
- $M(u)$: messages (colors) received by neighbors
- If $c(u) \in M(u)$ (a conflict), set $c(u) \leftarrow \perp$
- Switch state: $s(u) \leftarrow 1$

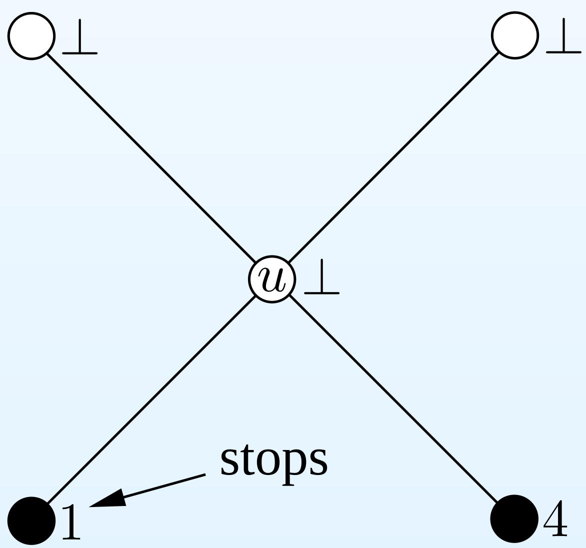
Even round $i + 1$:



Algorithm for Node u: Even Round $(2, 4, 6, \ldots)$

- Send $c(u)$ to neighbors
- $M(u)$: messages (colors) received by neighbors
- If $c(u) \in M(u)$ (a conflict), set $c(u) \leftarrow \perp$
- Switch state: $s(u) \leftarrow 1$

Even round $i + 1$:



Algorithm for Node u: Even Round $(2, 4, 6, \ldots)$

- Send $c(u)$ to neighbors
- $M(u)$: messages (colors) received by neighbors
- If $c(u) \in M(u)$ (a conflict), set $c(u) \leftarrow \perp$
- Switch state: $s(u) \leftarrow 1$

- Recall that u stops when $s(u) = 1$ and $c(u) \neq \perp$

Algorithm for Node u: Even Round $(2, 4, 6, \ldots)$

- Send $c(u)$ to neighbors
- $M(u)$: messages (colors) received by neighbors
- If $c(u) \in M(u)$ (a conflict), set $c(u) \leftarrow \perp$
- Switch state: $s(u) \leftarrow 1$

- Recall that u stops when $s(u) = 1$ and $c(u) \neq \perp$
- Thus, u stops at the end of an even round if:

Algorithm for Node u: Even Round $(2, 4, 6, \ldots)$

- Send $c(u)$ to neighbors
- $M(u)$: messages (colors) received by neighbors
- If $c(u) \in M(u)$ (a conflict), set $c(u) \leftarrow \perp$
- Switch state: $s(u) \leftarrow 1$

- Recall that u stops when $s(u) = 1$ and $c(u) \neq \perp$
- Thus, u stops at the end of an even round if:
  - it was assigned a color in the previous (odd) round, and

Algorithm for Node u: Even Round $(2, 4, 6, \ldots)$

- Send $c(u)$ to neighbors
- $M(u)$: messages (colors) received by neighbors
- If $c(u) \in M(u)$ (a conflict), set $c(u) \leftarrow \perp$
- Switch state: $s(u) \leftarrow 1$

- Recall that u stops when $s(u) = 1$ and $c(u) \neq \perp$
- Thus, u stops at the end of an even round if:
  - it was assigned a color in the previous (odd) round, and
  - there is no conflict in the current (even) round

Correctness

- A node u only stops when it:

Correctness

- A node u only stops when it:
  - has a color from $C(u)$ (assigned in an odd round)

Correctness

- A node u only stops when it:
  - has a color from $C(u)$ (assigned in an odd round)
  - has no conflicts with neighbors (in the even round where it stops)

Correctness

- A node u only stops when it:
  - has a color from $C(u)$ (assigned in an odd round)
  - has no conflicts with neighbors (in the even round where it stops)
- u keeps its color once stopped

Correctness

- A node u only stops when it:
  - has a color from $C(u)$ (assigned in an odd round)
  - has no conflicts with neighbors (in the even round where it stops)
- u keeps its color once stopped
- If a neighbor v changes its color in an odd round after u stopped:

Correctness

- A node u only stops when it:
  - has a color from $C(u)$ (assigned in an odd round)
  - has no conflicts with neighbors (in the even round where it stops)
- u keeps its color once stopped
- If a neighbor v changes its color in an odd round after u stopped:
  - u sends $c(u)$ to v in that round so $c(u) \in M(v)$

Correctness

- A node u only stops when it:
  - has a color from $C(u)$ (assigned in an odd round)
  - has no conflicts with neighbors (in the even round where it stops)
- u keeps its color once stopped
- If a neighbor v changes its color in an odd round after u stopped:
  - u sends $c(u)$ to v in that round so $c(u) \in M(v)$
  - v then picks a color from $F(v) = C(v) \setminus M(v)$ so $c(v) \neq c(u)$

Correctness

- A node u only stops when it:
  - has a color from $C(u)$ (assigned in an odd round)
  - has no conflicts with neighbors (in the even round where it stops)
- u keeps its color once stopped
- If a neighbor v changes its color in an odd round after u stopped:
  - u sends $c(u)$ to v in that round so $c(u) \in M(v)$
  - v then picks a color from $F(v) = C(v) \setminus M(v)$ so $c(v) \neq c(u)$
- Thus, if the algorithm terminates, it outputs a valid coloring

Correctness

- A node u only stops when it:
  - has a color from $C(u)$ (assigned in an odd round)
  - has no conflicts with neighbors (in the even round where it stops)
- u keeps its color once stopped
- If a neighbor v changes its color in an odd round after u stopped:
  - u sends $c(u)$ to v in that round so $c(u) \in M(v)$
  - v then picks a color from $F(v) = C(v) \setminus M(v)$ so $c(v) \neq c(u)$
- Thus, if the algorithm terminates, it outputs a valid coloring
- Since each $c(u) \in C(u) = \{1, \ldots, \deg_G(u) + 1\}$, the output is a $(\Delta + 1)$-coloring

Competitors in Odd Round

- Recall that in an odd round for a node u that has not stopped:

Competitors in Odd Round

- Recall that in an odd round for a node u that has not stopped:
  - $F(u) = C(u) \setminus M(u)$: *free* colors

Competitors in Odd Round

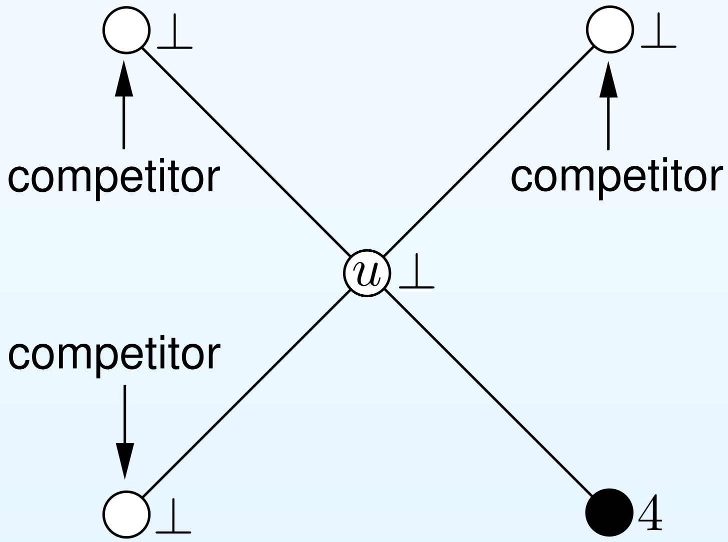
- Recall that in an odd round for a node u that has not stopped:
  - $F(u) = C(u) \setminus M(u)$: *free* colors
  - With probability $1/2$, pick $c(u)$ from $F(u)$ uniformly at random

Competitors in Odd Round

- Recall that in an odd round for a node u that has not stopped:
  - $F(u) = C(u) \setminus M(u)$: *free* colors
  - With probability $1/2$, pick $c(u)$ from $F(u)$ uniformly at random
- $K(u)$: *competitors* of u, neighbors that have not stopped

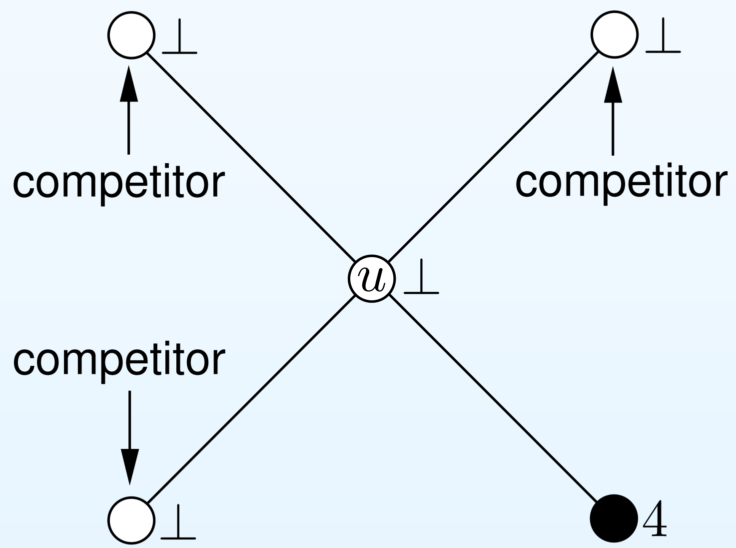
Competitors in Odd Round

- Recall that in an odd round for a node u that has not stopped:
  - $F(u) = C(u) \setminus M(u)$: *free* colors
  - With probability $1/2$, pick $c(u)$ from $F(u)$ uniformly at random
- $K(u)$: *competitors* of u, neighbors that have not stopped



Competitors in Odd Round

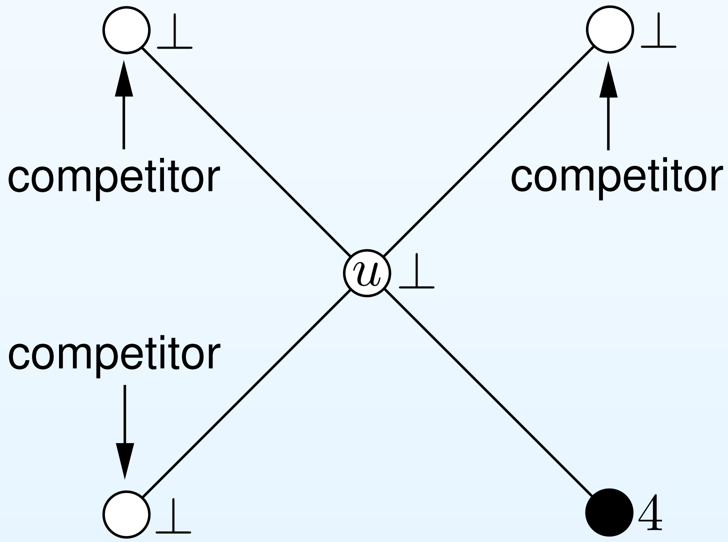
- Recall that in an odd round for a node u that has not stopped:
  - $F(u) = C(u) \setminus M(u)$: *free* colors
  - With probability $1/2$, pick $c(u)$ from $F(u)$ uniformly at random
- $K(u)$: *competitors* of u, neighbors that have not stopped



- Competitors are the only ones that u can have a color conflict with at the end of the round:

Competitors in Odd Round

- Recall that in an odd round for a node u that has not stopped:
    - $F(u) = C(u) \setminus M(u)$: *free* colors
    - With probability $1/2$, pick $c(u)$ from $F(u)$ uniformly at random
- $K(u)$: *competitors* of u, neighbors that have not stopped



- Competitors are the only ones that u can have a color conflict with at the end of the round:
    - All other neighbors have colors in $M(u)$ which are not in $F(u)$

Showing $O(\log n)$ Rounds w.h.p.: Proof Sketch

- In an odd round, assume u picks a color

Showing $O(\log n)$ Rounds w.h.p.: Proof Sketch

- In an odd round, assume u picks a color
- We expect only half the competitors of u to also pick a color

Showing $O(\log n)$ Rounds w.h.p.: Proof Sketch

- In an odd round, assume u picks a color
- We expect only half the competitors of u to also pick a color
- Thus, at least half of the free colors for u should not give conflicts

Showing $O(\log n)$ Rounds w.h.p.: Proof Sketch

- In an odd round, assume u picks a color
- We expect only half the competitors of u to also pick a color
- Thus, at least half of the free colors for u should not give conflicts
- So if u picks a color (probability $1/2$), it should have no conflicts with probability at least $1/2$

Showing $O(\log n)$ Rounds w.h.p.: Proof Sketch

- In an odd round, assume u picks a color
- We expect only half the competitors of u to also pick a color
- Thus, at least half of the free colors for u should not give conflicts
- So if u picks a color (probability $1/2$), it should have no conflicts with probability at least $1/2$
- We therefore expect at least $1/4$ of the nodes u to stop at the end of each even round

Showing $O(\log n)$ Rounds w.h.p.: Proof Sketch

- In an odd round, assume u picks a color
- We expect only half the competitors of u to also pick a color
- Thus, at least half of the free colors for u should not give conflicts
- So if u picks a color (probability $1/2$), it should have no conflicts with probability at least $1/2$
- We therefore expect at least $1/4$ of the nodes u to stop at the end of each even round
- If this occurs, the algorithm stops after $O(\log n)$ rounds

Showing $O(\log n)$ Rounds w.h.p.: Proof Sketch

- In an odd round, assume u picks a color
- We expect only half the competitors of u to also pick a color
- Thus, at least half of the free colors for u should not give conflicts
- So if u picks a color (probability $1/2$), it should have no conflicts with probability at least $1/2$
- We therefore expect at least $1/4$ of the nodes u to stop at the end of each even round
- If this occurs, the algorithm stops after $O(\log n)$ rounds
- We now formally prove that it stops in $O(\log n)$ rounds w.h.p.

Number of Free Colors

- Recall that in an odd round for a node u that has not stopped:
  - $F(u) = C(u) \setminus M(u)$: *free* colors
  - $K(u)$: competitors of u, neighbors that have not stopped

Number of Free Colors

- Recall that in an odd round for a node u that has not stopped:
  - $F(u) = C(u) \setminus M(u)$: *free* colors
  - $K(u)$: competitors of u, neighbors that have not stopped
- Consider the beginning of an odd round

Number of Free Colors

- Recall that in an odd round for a node u that has not stopped:
  - $F(u) = C(u) \setminus M(u)$: *free* colors
  - $K(u)$: competitors of u, neighbors that have not stopped
- Consider the beginning of an odd round
- Let $k = |K(u)|$ be the number of competitors of u

Number of Free Colors

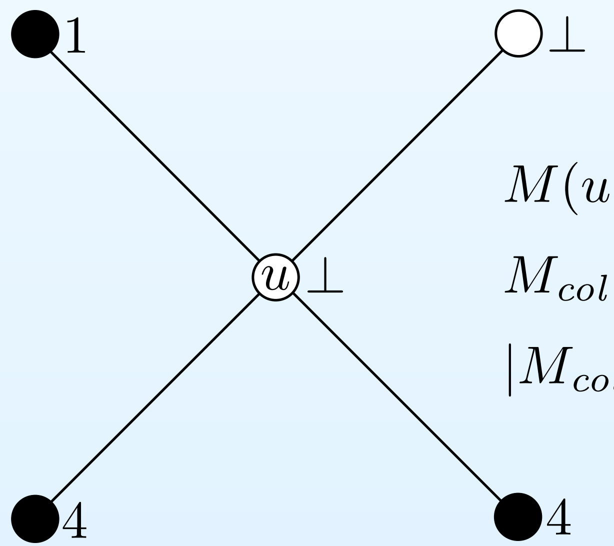
- Recall that in an odd round for a node u that has not stopped:
  - $F(u) = C(u) \setminus M(u)$: *free* colors
  - $K(u)$: competitors of u, neighbors that have not stopped
- Consider the beginning of an odd round
- Let $k = |K(u)|$ be the number of competitors of u
- We have $c(v) = \perp$ for each $v \in K(u)$

Number of Free Colors

- Recall that in an odd round for a node u that has not stopped:
  - $F(u) = C(u) \setminus M(u)$: *free* colors
  - $K(u)$: competitors of u, neighbors that have not stopped
- Consider the beginning of an odd round
- Let $k = |K(u)|$ be the number of competitors of u
- We have $c(v) = \perp$ for each $v \in K(u)$
- $M_{col}(u) = M(u) \setminus \{\perp\}$ thus has $\leq \deg_G(u) - k$ distinct colors

Number of Free Colors

- Recall that in an odd round for a node u that has not stopped:
  - $F(u) = C(u) \setminus M(u)$: *free* colors
  - $K(u)$: competitors of u, neighbors that have not stopped
- Consider the beginning of an odd round
- Let $k = |K(u)|$ be the number of competitors of u
- We have $c(v) = \perp$ for each $v \in K(u)$
- $M_{col}(u) = M(u) \setminus \{\perp\}$ thus has $\leq \deg_G(u) - k$ distinct colors



$M(u) = \{1, 4, \perp\}$

$M_{col}(u) = \{1, 4\}$

$|M_{col}(u)| = 2 \leq 4 - 1$

Number of Free Colors

- Recall that in an odd round for a node u that has not stopped:
  - $F(u) = C(u) \setminus M(u)$: *free* colors
  - $K(u)$: competitors of u, neighbors that have not stopped
- Consider the beginning of an odd round
- Let $k = |K(u)|$ be the number of competitors of u
- We have $c(v) = \perp$ for each $v \in K(u)$
- $M_{col}(u) = M(u) \setminus \{\perp\}$ thus has $\leq \deg_G(u) - k$ distinct colors
- Letting $f = |F(u)|$, we then get

Number of Free Colors

- Recall that in an odd round for a node u that has not stopped:
  - $F(u) = C(u) \setminus M(u)$: *free* colors
  - $K(u)$: competitors of u, neighbors that have not stopped
- Consider the beginning of an odd round
- Let $k = |K(u)|$ be the number of competitors of u
- We have $c(v) = \perp$ for each $v \in K(u)$
- $M_{col}(u) = M(u) \setminus \{\perp\}$ thus has $\leq \deg_G(u) - k$ distinct colors
- Letting $f = |F(u)|$, we then get

$$f = |C(u) \setminus M(u)|$$

Number of Free Colors

- Recall that in an odd round for a node u that has not stopped:
  - $F(u) = C(u) \setminus M(u)$: *free* colors
  - $K(u)$: competitors of u, neighbors that have not stopped
- Consider the beginning of an odd round
- Let $k = |K(u)|$ be the number of competitors of u
- We have $c(v) = \perp$ for each $v \in K(u)$
- $M_{col}(u) = M(u) \setminus \{\perp\}$ thus has $\leq \deg_G(u) - k$ distinct colors
- Letting $f = |F(u)|$, we then get

$$\begin{aligned} f &= |C(u) \setminus M(u)| \\ &= |C(u) \setminus M_{col}(u)| \end{aligned}$$

Number of Free Colors

- Recall that in an odd round for a node u that has not stopped:
  - $F(u) = C(u) \setminus M(u)$: *free* colors
  - $K(u)$: competitors of u, neighbors that have not stopped
- Consider the beginning of an odd round
- Let $k = |K(u)|$ be the number of competitors of u
- We have $c(v) = \perp$ for each $v \in K(u)$
- $M_{col}(u) = M(u) \setminus \{\perp\}$ thus has $\leq \deg_G(u) - k$ distinct colors
- Letting $f = |F(u)|$, we then get

$$\begin{aligned} f &= |C(u) \setminus M(u)| \\ &= |C(u) \setminus M_{col}(u)| \\ &\geq |C(u)| - |M_{col}(u)| \end{aligned}$$

Number of Free Colors

- Recall that in an odd round for a node u that has not stopped:
  - $F(u) = C(u) \setminus M(u)$: *free* colors
  - $K(u)$: competitors of u, neighbors that have not stopped
- Consider the beginning of an odd round
- Let $k = |K(u)|$ be the number of competitors of u
- We have $c(v) = \perp$ for each $v \in K(u)$
- $M_{col}(u) = M(u) \setminus \{\perp\}$ thus has $\leq \deg_G(u) - k$ distinct colors
- Letting $f = |F(u)|$, we then get

$$\begin{aligned} f &= |C(u) \setminus M(u)| \\ &= |C(u) \setminus M_{col}(u)| \\ &\geq |C(u)| - |M_{col}(u)| \\ &= (\deg_G(u) + 1) - |M_{col}(u)| \end{aligned}$$

Number of Free Colors

- Recall that in an odd round for a node u that has not stopped:
    - $F(u) = C(u) \setminus M(u)$: *free* colors
    - $K(u)$: competitors of u, neighbors that have not stopped
- Consider the beginning of an odd round
- Let $k = |K(u)|$ be the number of competitors of u
- We have $c(v) = \perp$ for each $v \in K(u)$
- $M_{col}(u) = M(u) \setminus \{\perp\}$ thus has $\leq \deg_G(u) - k$ distinct colors
- Letting $f = |F(u)|$, we then get

$$\begin{aligned} f &= |C(u) \setminus M(u)| \\ &= |C(u) \setminus M_{col}(u)| \\ &\geq |C(u)| - |M_{col}(u)| \\ &= (\deg_G(u) + 1) - |M_{col}(u)| \\ &\geq (\deg_G(u) + 1) - (\deg_G(u) - k) \end{aligned}$$

Number of Free Colors

- Recall that in an odd round for a node u that has not stopped:
  - $F(u) = C(u) \setminus M(u)$: *free* colors
  - $K(u)$: competitors of u, neighbors that have not stopped
- Consider the beginning of an odd round
- Let $k = |K(u)|$ be the number of competitors of u
- We have $c(v) = \perp$ for each $v \in K(u)$
- $M_{col}(u) = M(u) \setminus \{\perp\}$ thus has $\leq \deg_G(u) - k$ distinct colors
- Letting $f = |F(u)|$, we then get

$$\begin{aligned}
f &= |C(u) \setminus M(u)| \\
&= |C(u) \setminus M_{col}(u)| \\
&\geq |C(u)| - |M_{col}(u)| \\
&= (\deg_G(u) + 1) - |M_{col}(u)| \\
&\geq (\deg_G(u) + 1) - (\deg_G(u) - k) \\
&= k + 1
\end{aligned}$$

Probability of Conflict in an Odd Round

- Have shown: number of free colors $f \geq k + 1$

Probability of Conflict in an Odd Round

- Have shown: number of free colors $f \geq k + 1$
- We consider what happens *at the end* of an odd round

Probability of Conflict in an Odd Round

- Have shown: number of free colors $f \geq k + 1$
- We consider what happens *at the end* of an odd round
- Let v be a competitor of u and assume u picks a color: $c(u) \neq \perp$

Probability of Conflict in an Odd Round

- Have shown: number of free colors $f \geq k + 1$
- We consider what happens *at the end* of an odd round
- Let v be a competitor of u and assume u picks a color: $c(u) \neq \perp$
- The probability of a conflict conditioned on v also picking a color:

$$\mathrm{P}[c(u) = c(v) \mid c(u), c(v) \neq \perp] \leq \frac{1}{f}$$

Probability of Conflict in an Odd Round

- Have shown: number of free colors $f \geq k+1$
- We consider what happens *at the end* of an odd round
- Let v be a competitor of u and assume u picks a color: $c(u) \neq \perp$
- The probability of a conflict conditioned on v also picking a color:

$$\mathrm{P}[c(u) = c(v) \mid c(u), c(v) \neq \perp] \leq \frac{1}{f}$$

- This follows since at most 1 of the f choices for u gives a conflict

Probability of Conflict in an Odd Round

- Have shown: number of free colors $f \geq k + 1$
- We consider what happens *at the end* of an odd round
- Let v be a competitor of u and assume u picks a color: $c(u) \neq \perp$
- The probability of a conflict conditioned on v also picking a color:

$$\mathrm{P}[c(u) = c(v) \mid c(u), c(v) \neq \perp] \leq \frac{1}{f}$$

- This follows since at most 1 of the f choices for u gives a conflict
- Since v picks a color with probability $1/2$,

$$\begin{aligned}
&\mathrm{P}[c(u) = c(v) \mid c(u) \neq \perp] \\
&\quad = \mathrm{P}[c(u) = c(v) \mid c(u), c(v) \neq \perp] \cdot \mathrm{P}[c(v) \neq \perp]
\end{aligned}$$

Probability of Conflict in an Odd Round

- Have shown: number of free colors $f \geq k+1$
- We consider what happens *at the end* of an odd round
- Let v be a competitor of u and assume u picks a color: $c(u) \neq \perp$
- The probability of a conflict conditioned on v also picking a color:

$$\mathrm{P}[c(u) = c(v) \mid c(u), c(v) \neq \perp] \leq \frac{1}{f}$$

- This follows since at most 1 of the f choices for u gives a conflict
- Since v picks a color with probability $1/2$,

$$\begin{aligned} &\mathrm{P}[c(u) = c(v) \mid c(u) \neq \perp] \\ &\quad = \mathrm{P}[c(u) = c(v) \mid c(u), c(v) \neq \perp] \cdot \mathrm{P}[c(v) \neq \perp] \\ &\quad \leq \frac{1}{f} \cdot \mathrm{P}[c(v) \neq \perp] \end{aligned}$$

Probability of Conflict in an Odd Round

- Have shown: number of free colors $f \geq k+1$
- We consider what happens *at the end* of an odd round
- Let v be a competitor of u and assume u picks a color: $c(u) \neq \perp$
- The probability of a conflict conditioned on v also picking a color:

$$\mathrm{P}[c(u) = c(v) \mid c(u), c(v) \neq \perp] \leq \frac{1}{f}$$

- This follows since at most 1 of the f choices for u gives a conflict
- Since v picks a color with probability $1/2$,

$$\begin{aligned} &\mathrm{P}[c(u) = c(v) \mid c(u) \neq \perp] \\ &\quad = \mathrm{P}[c(u) = c(v) \mid c(u), c(v) \neq \perp] \cdot \mathrm{P}[c(v) \neq \perp] \\ &\quad \leq \frac{1}{f} \cdot \mathrm{P}[c(v) \neq \perp] = \frac{1}{2f} \end{aligned}$$

Probability of Conflict in an Odd Round: Union Bound

- Have shown $f \geq k+1$ and

$$\mathrm{P}[c(u) = c(v) \mid c(u) \neq \perp] \leq \frac{1}{2f}$$

Probability of Conflict in an Odd Round: Union Bound

- Have shown $f \geq k+1$ and

$$\mathrm{P}[c(u) = c(v) \mid c(u) \neq \perp] \leq \frac{1}{2f}$$

- Since $f \geq k+1$, a union bound shows that the probability of a conflict between u and at least one competitor v is

$$\mathrm{P}\left[\bigcup_{v \in K(u)} \{c(u) = c(v) \mid c(u) \neq \perp\}\right]$$

Probability of Conflict in an Odd Round: Union Bound

- Have shown $f \geq k+1$ and

$$\mathrm{P}[c(u) = c(v) \mid c(u) \neq \perp] \leq \frac{1}{2f}$$

- Since $f \geq k+1$, a union bound shows that the probability of a conflict between u and at least one competitor v is

$$\mathrm{P}\left[\bigcup_{v \in K(u)} \{c(u) = c(v) \mid c(u) \neq \perp\}\right] \leq \frac{k}{2f}$$

Probability of Conflict in an Odd Round: Union Bound

- Have shown $f \geq k+1$ and

$$\mathrm{P}[c(u) = c(v) \mid c(u) \neq \perp] \leq \frac{1}{2f}$$

- Since $f \geq k+1$, a union bound shows that the probability of a conflict between u and at least one competitor v is

$$\mathrm{P}\left[\bigcup_{v \in K(u)} \{c(u) = c(v) \mid c(u) \neq \perp\}\right] \leq \frac{k}{2f} \leq \frac{k}{2(k+1)}$$

Probability of Conflict in an Odd Round: Union Bound

- Have shown $f \geq k+1$ and

$$\mathrm{P}[c(u) = c(v) \mid c(u) \neq \perp] \leq \frac{1}{2f}$$

- Since $f \geq k+1$, a union bound shows that the probability of a conflict between u and at least one competitor v is

$$\mathrm{P}\left[\bigcup_{v \in K(u)} \{c(u) = c(v) \mid c(u) \neq \perp\}\right] \leq \frac{k}{2f} \leq \frac{k}{2(k+1)} < \frac{1}{2}$$

Probability of Node u Stopping in a Given Round

- Have shown: if u picks a color, it is in conflict with at least one competitor with probability $< \frac{1}{2}$

Probability of Node u Stopping in a Given Round

- Have shown: if u picks a color, it is in conflict with at least one competitor with probability $< \frac{1}{2}$
- Hence, if u picks a color, it has no conflict with probability $> \frac{1}{2}$

Probability of Node u Stopping in a Given Round

- Have shown: if u picks a color, it is in conflict with at least one competitor with probability $< \frac{1}{2}$
- Hence, if u picks a color, it has no conflict with probability $> \frac{1}{2}$
- Probability that u picks a color: $\frac{1}{2}$

Probability of Node u Stopping in a Given Round

- Have shown: if u picks a color, it is in conflict with at least one competitor with probability $< \frac{1}{2}$
- Hence, if u picks a color, it has no conflict with probability $> \frac{1}{2}$
- Probability that u picks a color: $\frac{1}{2}$
- Probability that u picks a color *and* has no conflict:

$$\mathrm{P}[u \text{ has no conflict} \mid c(u) \neq \perp] \cdot \mathrm{P}[c(u) \neq \perp] > \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$$

Probability of Node u Stopping in a Given Round

- Have shown: if u picks a color, it is in conflict with at least one competitor with probability $< \frac{1}{2}$
- Hence, if u picks a color, it has no conflict with probability $> \frac{1}{2}$
- Probability that u picks a color: $\frac{1}{2}$
- Probability that u picks a color *and* has no conflict:

$$\mathrm{P}[u \text{ has no conflict} \mid c(u) \neq \perp] \cdot \mathrm{P}[c(u) \neq \perp] > \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$$

- Recall that u stops in the following even round if it picks a color which does not conflict with any competitors

Probability of Node u Stopping in a Given Round

- Have shown: if u picks a color, it is in conflict with at least one competitor with probability $< \frac{1}{2}$
- Hence, if u picks a color, it has no conflict with probability $> \frac{1}{2}$
- Probability that u picks a color: $\frac{1}{2}$
- Probability that u picks a color *and* has no conflict:

$$\mathrm{P}[u \text{ has no conflict} \mid c(u) \neq \perp] \cdot \mathrm{P}[c(u) \neq \perp] > \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$$

- Recall that u stops in the following even round if it picks a color which does not conflict with any competitors
- Probability of this: $> \frac{1}{4}$

Bounding the Number of Rounds

- Have shown: in every *second* round, u stops with probability $> \frac{1}{4}$

Bounding the Number of Rounds

- Have shown: in every *second* round, u stops with probability $> \frac{1}{4}$
- Probability that u has not stopped after $t \in \{2, 4, \ldots\}$ rounds is less than

$$(1 - 1/4)^{t/2} = (3/4)^{t/2}$$

Bounding the Number of Rounds

- Have shown: in every *second* round, u stops with probability $> \frac{1}{4}$
- Probability that u has not stopped after $t \in \{2, 4, \ldots\}$ rounds is less than

$$(1 - 1/4)^{t/2} = (3/4)^{t/2}$$

- By a union bound over all $u \in V$, the probability that the algorithm has not stopped after t rounds is less than

$$n(3/4)^{t/2}$$

Bounding the Number of Rounds

- Have shown: in every *second* round, u stops with probability $> \frac{1}{4}$
- Probability that u has not stopped after $t \in \{2, 4, \ldots\}$ rounds is less than

$$(1 - 1/4)^{t/2} = (3/4)^{t/2}$$

- By a union bound over all $u \in V$, the probability that the algorithm has not stopped after t rounds is less than

$$n(3/4)^{t/2}$$

- We pick t so that this is n^{-c} for constant $c > 0$:

$$n(3/4)^{t/2} = n^{-c}$$

Bounding the Number of Rounds

- Have shown: in every *second* round, u stops with probability $> \frac{1}{4}$
- Probability that u has not stopped after $t \in \{2, 4, \ldots\}$ rounds is less than

$$(1 - 1/4)^{t/2} = (3/4)^{t/2}$$

- By a union bound over all $u \in V$, the probability that the algorithm has not stopped after t rounds is less than

$$n(3/4)^{t/2}$$

- We pick t so that this is n^{-c} for constant $c > 0$:

$$n(3/4)^{t/2} = n^{-c} \Leftrightarrow$$
$$(4/3)^{t/2} = n^{c+1}$$

Bounding the Number of Rounds

- Have shown: in every *second* round, u stops with probability $> \frac{1}{4}$
- Probability that u has not stopped after $t \in \{2, 4, \ldots\}$ rounds is less than

$$(1 - 1/4)^{t/2} = (3/4)^{t/2}$$

- By a union bound over all $u \in V$, the probability that the algorithm has not stopped after t rounds is less than

$$n(3/4)^{t/2}$$

- We pick t so that this is n^{-c} for constant $c > 0$:

$$\begin{aligned} n(3/4)^{t/2} &= n^{-c} \Leftrightarrow \\ (4/3)^{t/2} &= n^{c+1} \Leftrightarrow \\ t &= 2(c+1)\log_{4/3} n \end{aligned}$$

Running Time With High Probability

- Have shown: the probability that the algorithm does not stop within $t = 2(c+1)\log_{4/3} n$ rounds is less than n^{-c}

Running Time With High Probability

- Have shown: the probability that the algorithm does not stop within $t = 2(c+1)\log_{4/3} n$ rounds is less than n^{-c}
- Thus, the algorithm runs in $O(\log n)$ time w.h.p.