Coloring a graph

• Assign a color to each vertex such that
  – Neighboring vertices always have different colors

• Easy with n colors

• Problem is harder with fewer colors
Application of coloring

• Suppose there are restrictions such that certain pairs of nodes must not operate (or access a resource) at the same time
• A coloring gives us sets of nodes that *can* operate at the same time
Example

• Suppose we have a wireless network
• Nearby nodes should not transmit at the same frequency (channel) at the same time
• We can construct a graph where nodes within range of each-other are connected by an edge
• A coloring of this graph is an assignment of communication channels to nodes
  – Such that they will not interfere
Example

• Suppose we have a wireless network
• Nearby nodes should not transmit at the same frequency (channel) at the same time
• We can construct a graph where nodes within range of each other are connected by an edge
• A coloring of this graph is an assignment of communication channels to nodes
  – Such that they will not interfere
• Alternatively, using time division access
  – A coloring is assignment of time slots
Independent set (IS)

• A subset of vertices that can have the same color
  – No two vertices are adjacent
  – In a coloring, vertices of each color form an IS
Maximum independent set (maxIS)

- Independent set of largest possible size

- NP-hard: polynomial time algorithm unlikely
Maximal IS (MIS)

• Independent set such that
  – No other vertex can be added to the set

• MIS can have very few vertices compared to MaxIS
MIS algorithm (synchronous)

• Each vertex has states
  – Undecided (initial)
  – Decided to enter MIS
  – Decided not to enter MIS

• Algorithm (repeated at each node until a decision (enter or not enter)):
  – If a neighbor has decided to enter MIS
    • Decide not to enter
  – If some neighbors are undecided and one or more undecided neighbor has higher id
    • Stay undecided
  – If some neighbors are undecided and none has higher id
    • Decide to enter MIS
MIS algorithm

• Time complexity: $O(n)$

• When nodes are in a chain, sorted by id
MIS

• We want something faster than $O(n)$
Fast-MIS (randomized)

- $d(v)$ is degree of $v$
- Each $v$ marks itself with probability $1/2d(v)$
- If no higher degree neighbor is marked
  - $v$ joins MIS
  - Else $v$ un-marks itself
- Remove all nodes that joined MIS and their neighbors
Fast-MIS

• Run time: $O(\log n)$

• Proof: somewhat long.

• If you want to learn more, see:
  – Alon et al. 1986: A fast and simple randomized parallel algorithm for the maximal independent set problem
  – Slides: http://www.net.t-labs.tu-berlin.de/~stefan/netalg13-6-MIS.pdf