CSCE 689: Special Topics in Modern Algorithms for Data Science

Lecture 21

Samson Zhou

Presentation Schedule

- November 27: Chunkai, Jung, Galaxy AI
- November 29: STMI, Anmol, Jason
- December 1: Bokun, Ayesha, Dawei, Lipai

Previously: Semi-streaming Model

- Recall that we have a graph $G = (V = [n],\ E)$
- Suppose $|E| = m$

- The edges of the graph arrive sequentially, i.e., insertion-only model

- We are allowed to use $n \cdot \text{polylog}(n)$ space

- Enough to store a matching, NOT enough to store entire graph, since m can be as large as $O(n^2)$

Last Time: Maximum Matching

- Greedy algorithm is a 2-approximation to the maximum matching that uses $O(n)$ space

- OPEN: Is it possible to achieve C-approximation to the maximum (cardinality) matching using $n \cdot \mathrm{polylog}(n)$ space for $C < 2$?

Last Time: Connectivity

- Connected graph: There exists a path between i and j for any pair $i, j \subseteq V = [n]$ of vertices

- Goal: Given a graph G, determine whether G is a connected graph

Last Time: Spanning Tree

- How to find a spanning tree in the offline setting?

- Minimum spanning tree algorithms (Kruskal, Prim)
  - Kruskal: Greedily add minimum weight edge to spanning forest
  - Prim: Greedily grow minimum spanning tree

Last Time: Connectivity

- Intuition: Greedily add edges to minimum spanning forest

- Algorithm:
  1. Initialize $F = \emptyset$.
  2. For each edge $e = (u, v)$:
     1. If $F \cup (u, v)$ does not contain a cycle, add (u, v) to F: $F \leftarrow F \cup (u, v)$
     2. If $|F| = n - 1$, return GRAPH IS CONNECTED
  3. Return GRAPH IS NOT CONNECTED

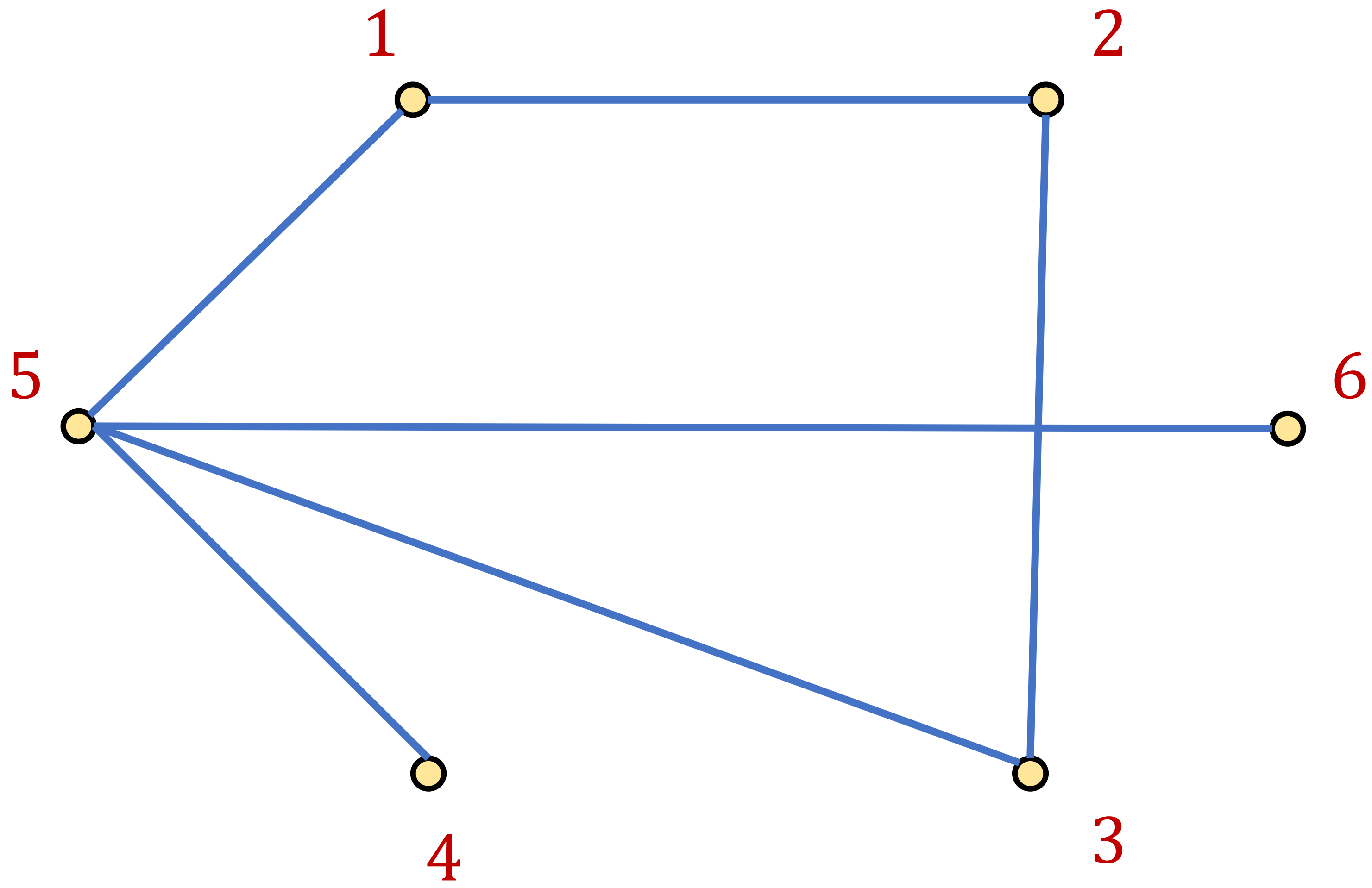
Last Time: Connectivity

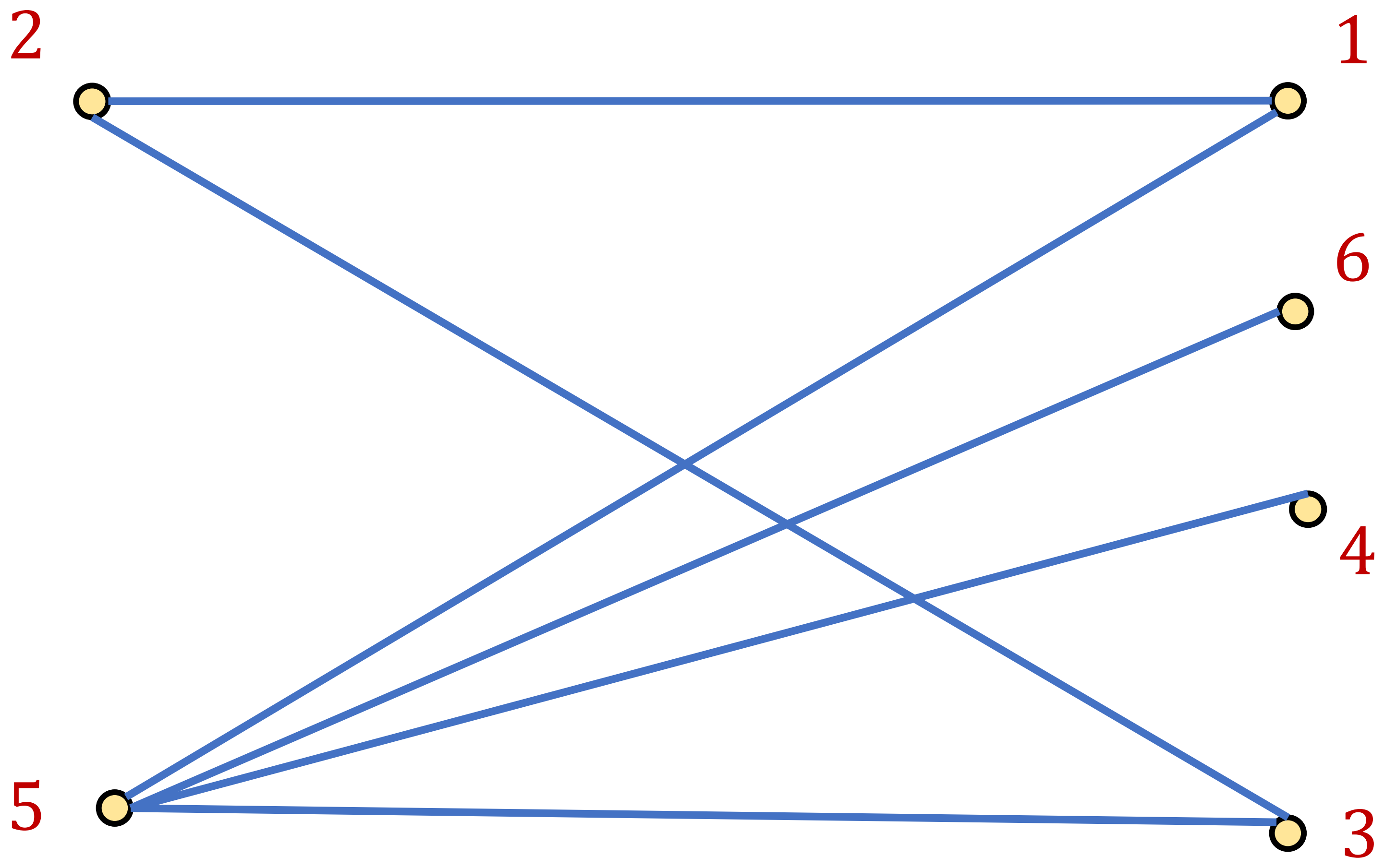
- Algorithm can keep at most n edges, so the total space usage is $O(n)$ words of space.

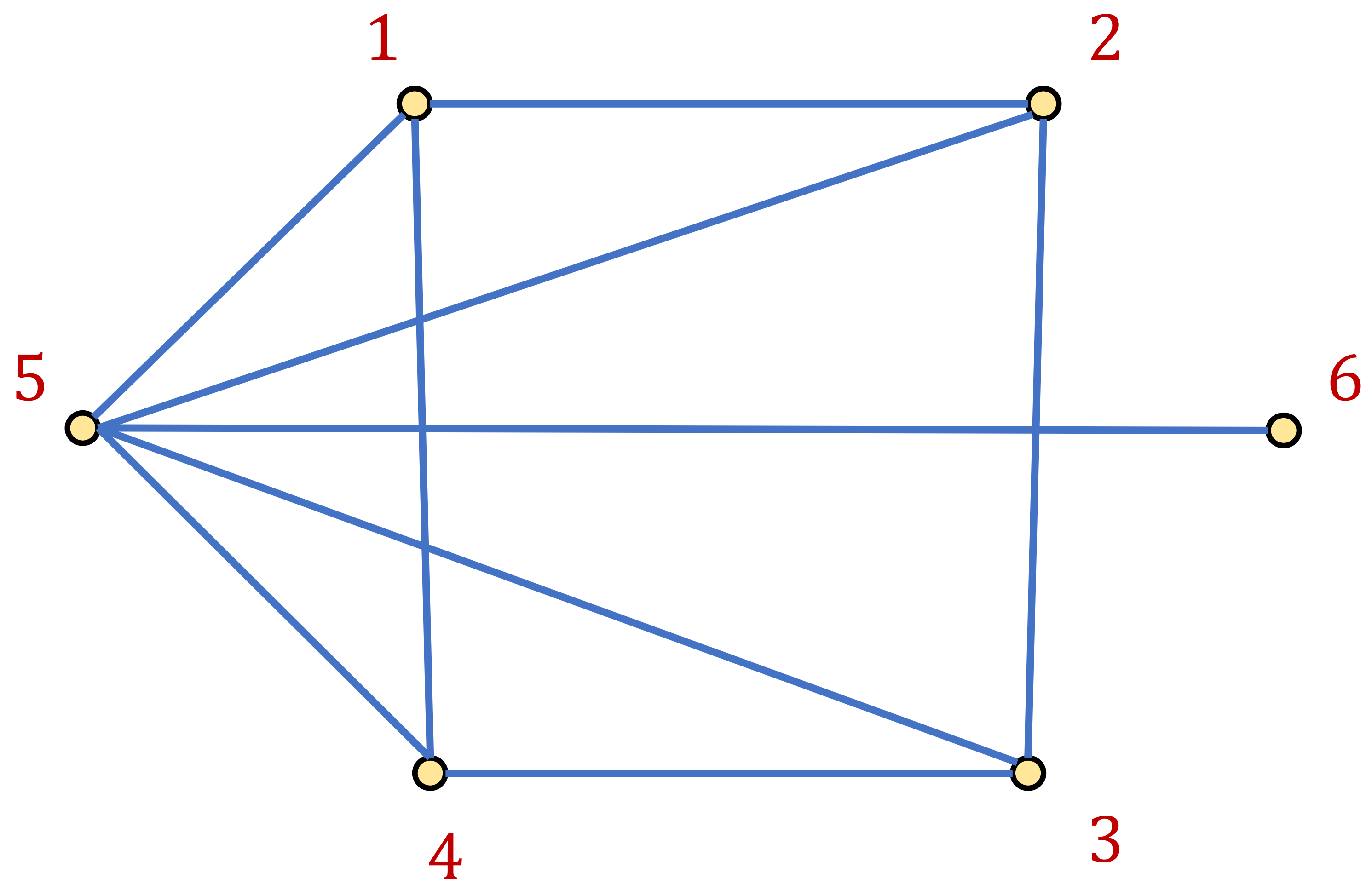
Bipartiteness

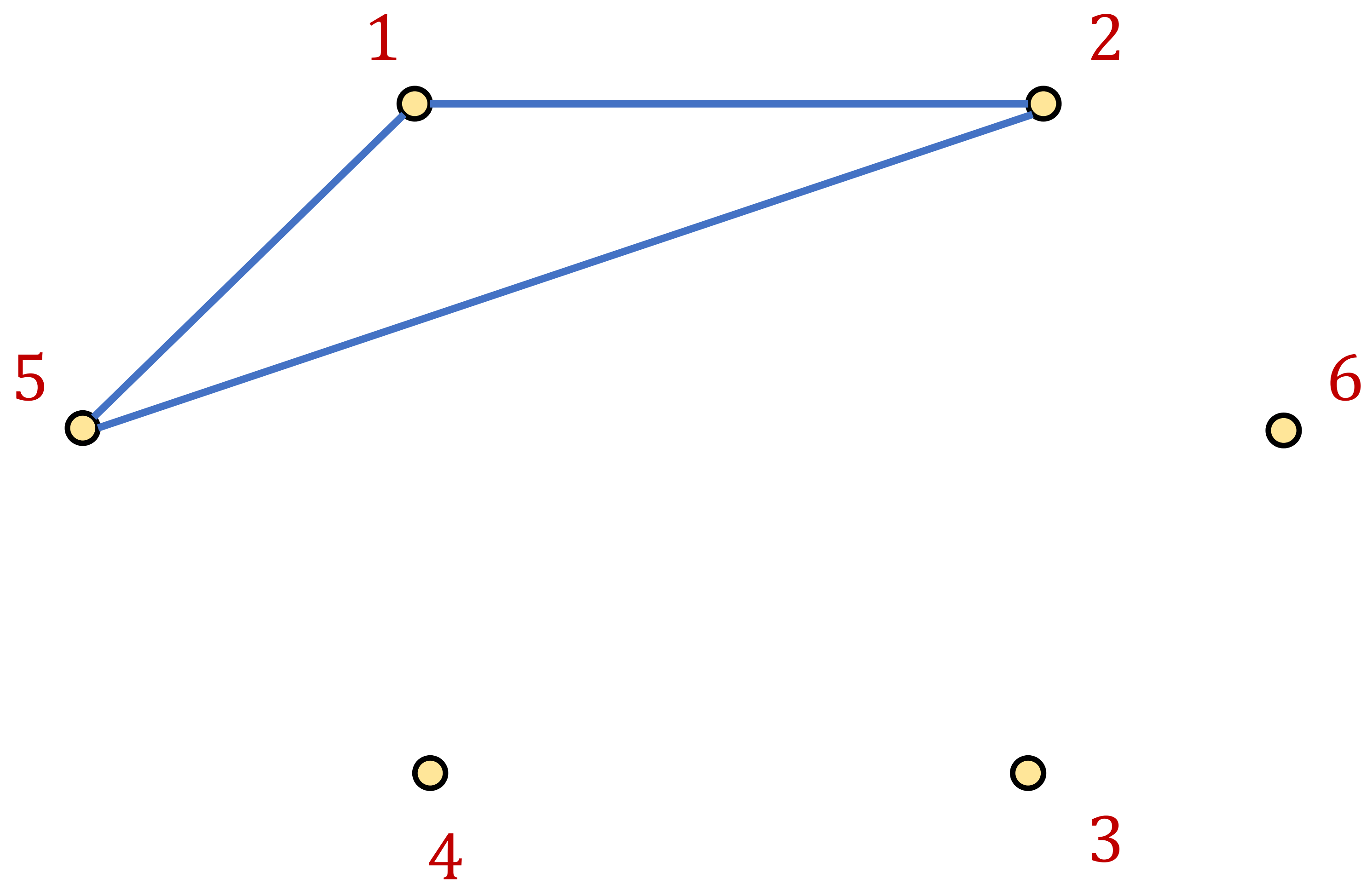
- Bipartite graph: Graph can be partitioned into two disjoint sets L and R so that every edge is between a vertex in L and a vertex in R

- Goal: Given a graph G, determine whether G is a bipartite graph


1
2
5
6
4
3

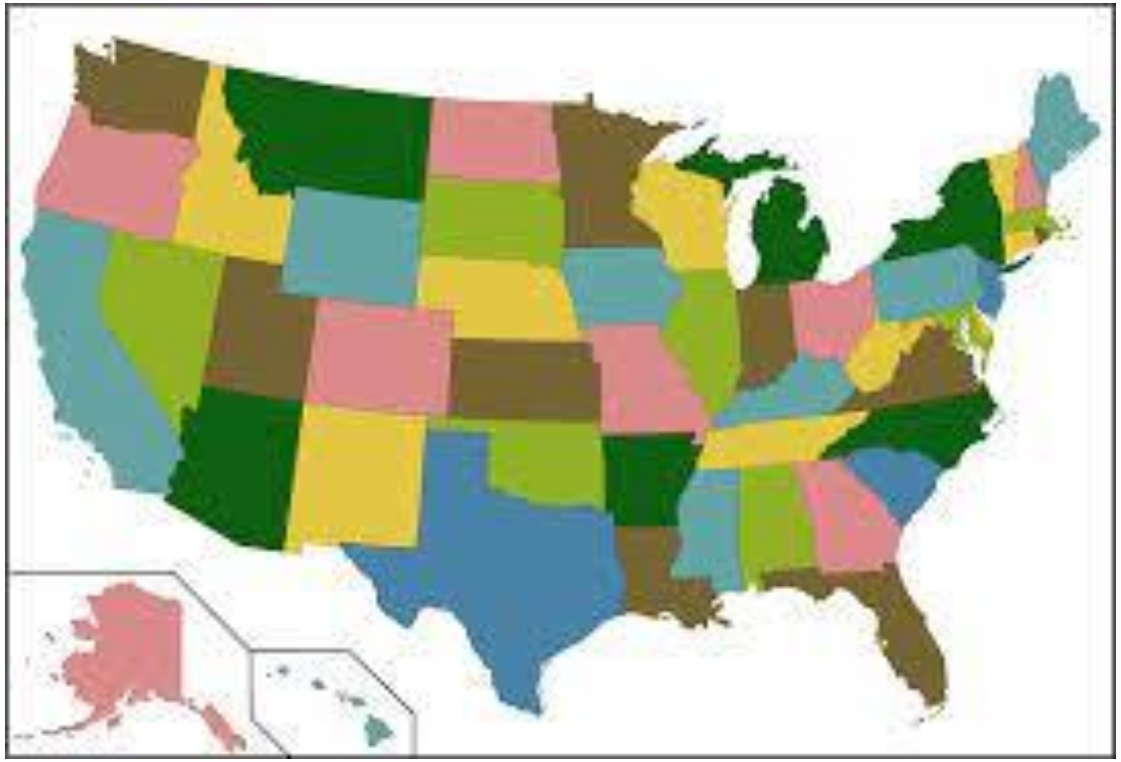

2
1
6
4
5
3

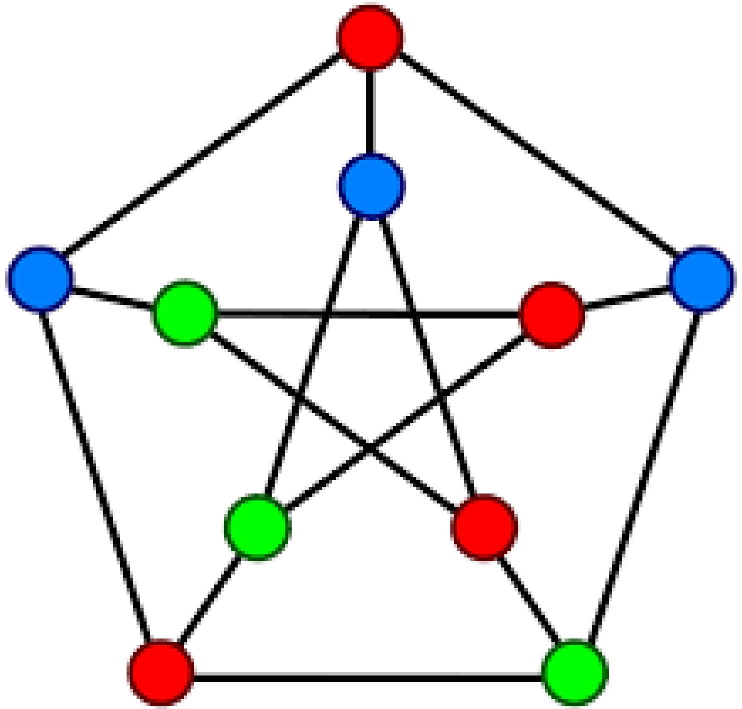

1
2
5
6
4
3


1
2
5
6
4
3

Applications for Bipartiteness Testing

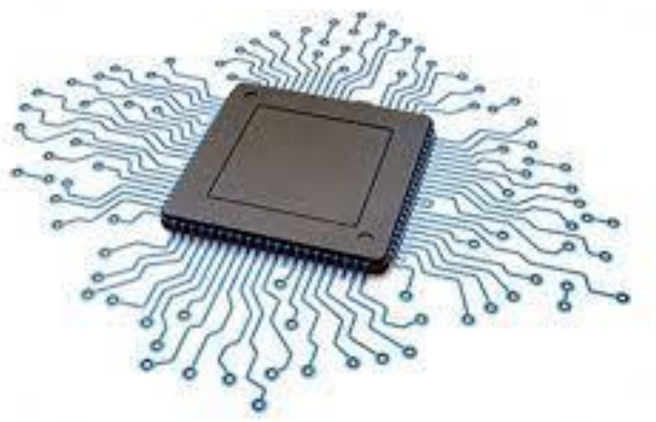
- Graph coloring: You want to color a graph such that no neighboring items share the same color





Applications for Bipartiteness Testing

- Circuit Design: In electrical engineering and VLSI (Very Large Scale Integration) design, you may want to know if a circuit can be optimally partitioned into two complementary parts, which can be achieved by testing the bipartiteness of the circuit's dependency graph



Bipartiteness

- What is a necessary and sufficient condition for bipartiteness?

Bipartiteness

- What is a necessary and sufficient condition for bipartiteness?

- A graph is bipartite if and only if it can be colored using two colors (a coloring of a graph is an assignment of colors to vertices such that no two vertices share the same color)

- A graph is bipartite if and only if it has no odd cycles

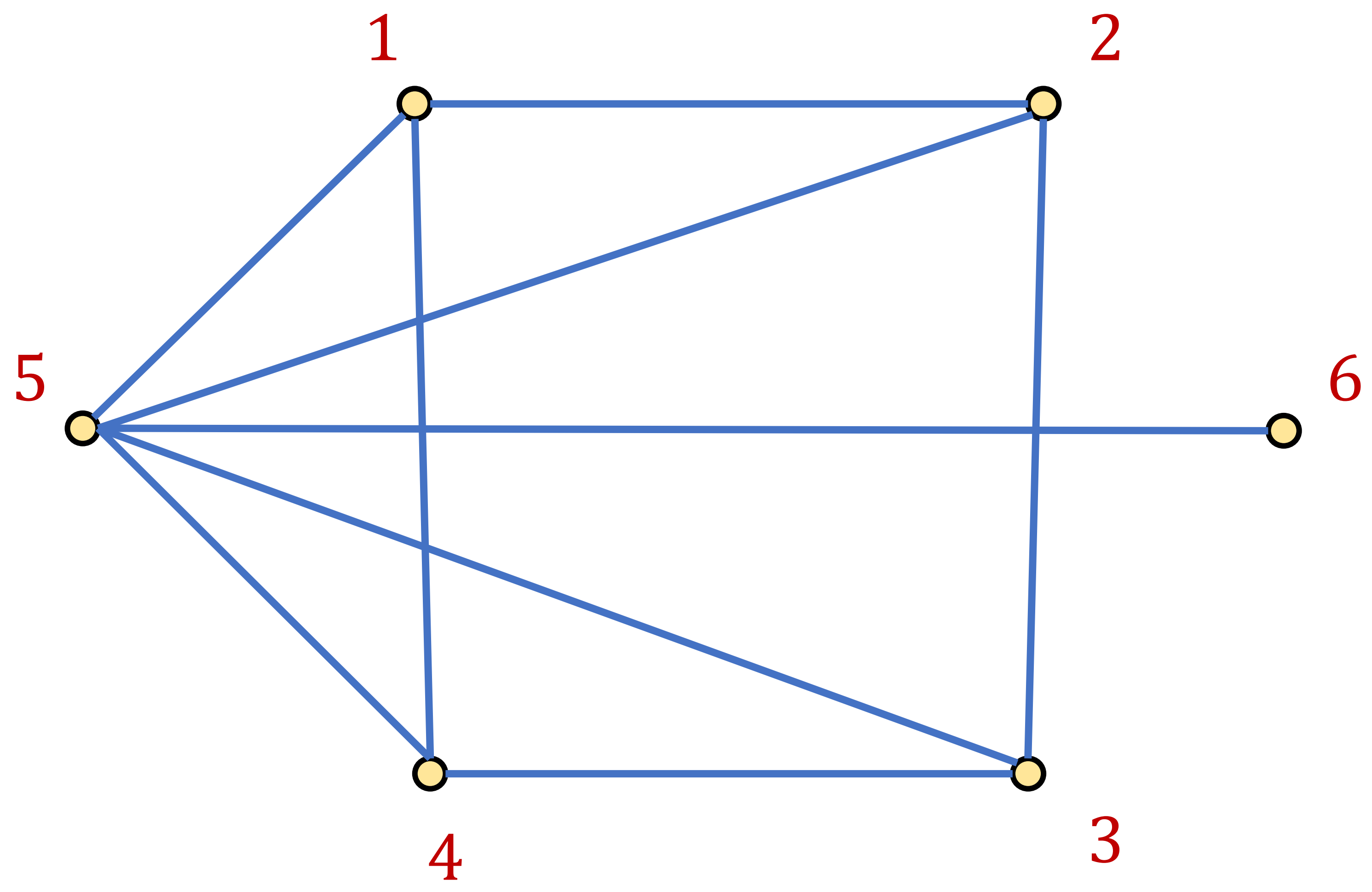
Bipartiteness

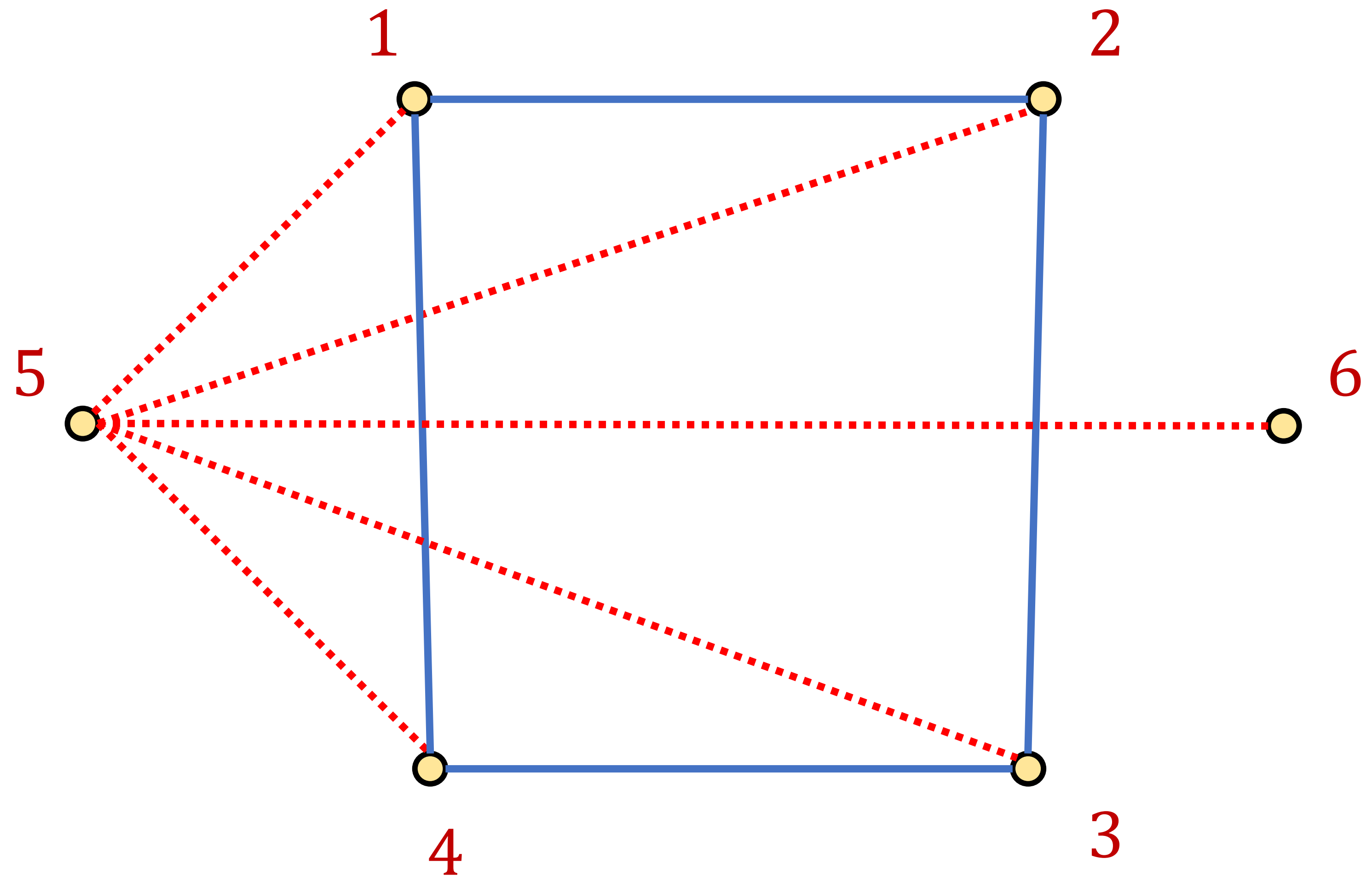
- How to perform bipartiteness testing in the central setting?

Bipartiteness

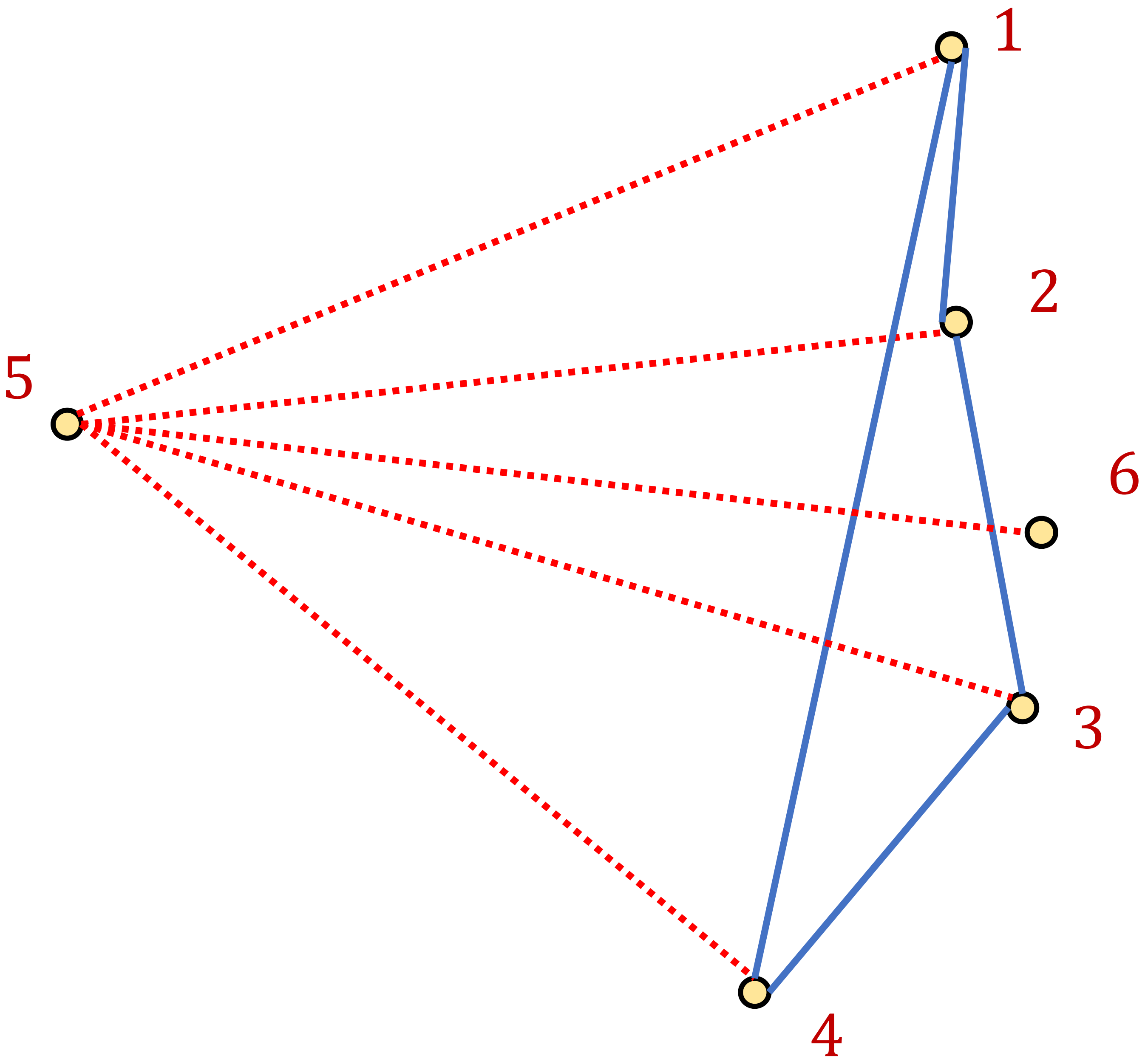
- How to perform bipartiteness testing in the central setting?

- Start at arbitrary vertex, run BFS, and assign alternating levels to different side until there is a contradiction

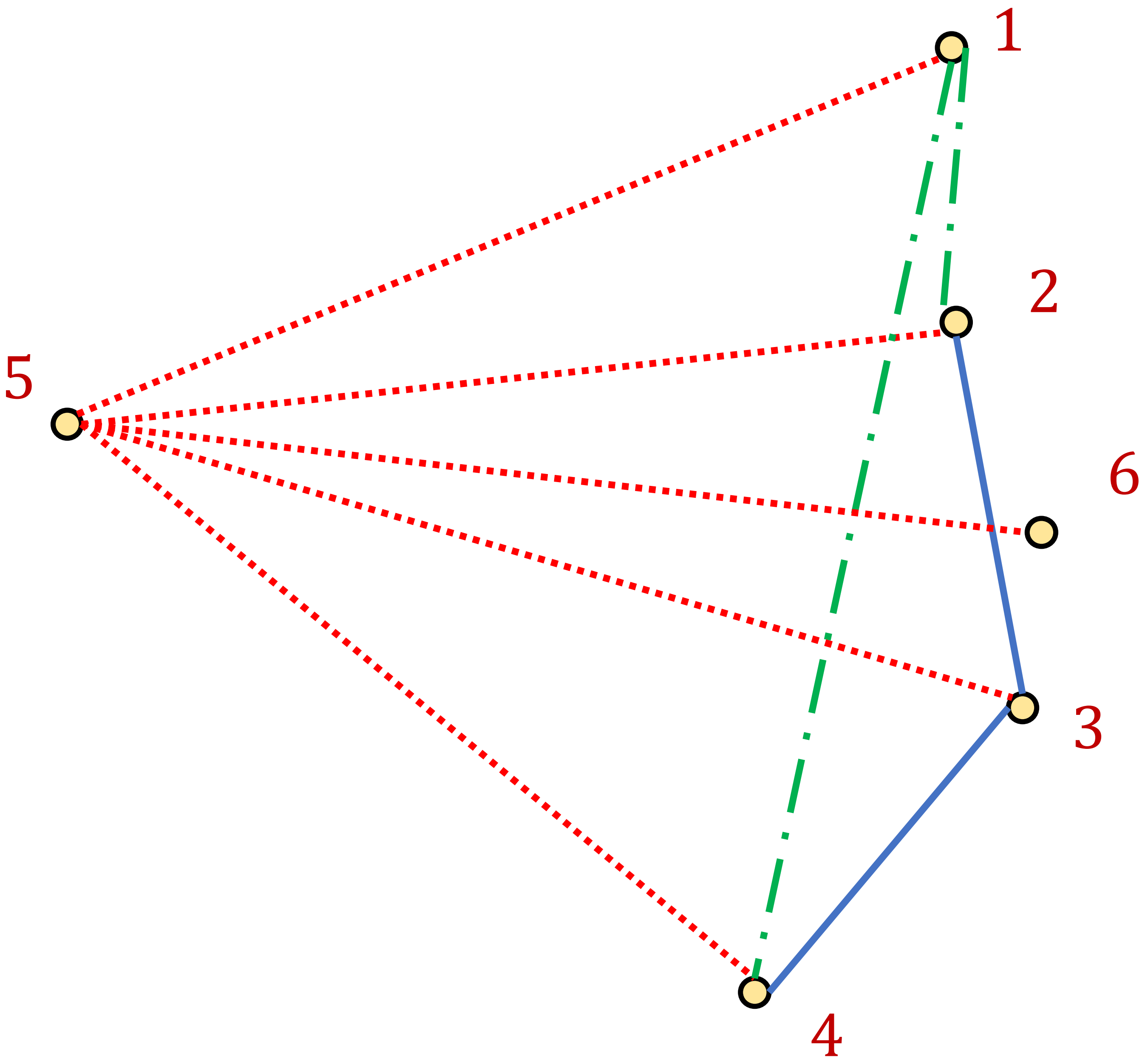

1
2
5
6
4
3



1
2
5
6
4
3



1
2
5
6
3
4



1
2
5
6
3
4

Bipartiteness in the Streaming Model

- Bipartiteness is a monotone property, i.e., additional edges to a graph that is not bipartite will result in a graph that is not bipartite

Bipartiteness in the Streaming Model

- Bipartiteness is a monotone property, i.e., additional edges to a graph that is not bipartite will result in a graph that is not bipartite

Bipartiteness in the Streaming Model

- Intuition: Greedily add edges to minimum spanning forest

- Algorithm:
  1. Initialize $F = \emptyset$.
  2. For each edge $e = (u, v)$:
     1. If $F \cup (u, v)$ does not contain a cycle, add (u, v) to F: $F \leftarrow F \cup (u, v)$
     2. If $F \cup (u, v)$ contains an odd cycle, return GRAPH IS NOT BIPARTITE
  3. Return GRAPH IS BIPARTITE

Bipartiteness in the Streaming Model

- Algorithm maintains a tree (because it does not add any edges that would create cycles)

- Algorithm can keep at most n edges, so the total space usage is $O(n)$ words of space.