CSCE 689: Special Topics in Modern Algorithms for Data Science

Lecture 20

Samson Zhou

Presentation Schedule

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

Last Time: 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 $n \cdot \mathrm{polylog}(n)$ 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

- How to find maximum matching?
- An *alternating path* is any path of edges that alternates between edges in and not in the matching
- An *augmenting path* is any alternating path of edges that does not start and does not end at a vertex in the matching
- "Flipping" all the edges in an augmenting path increases the matching size

Maximal Matching

- A maximal matching is a matching M of G such that any additional edges would no longer be a matching

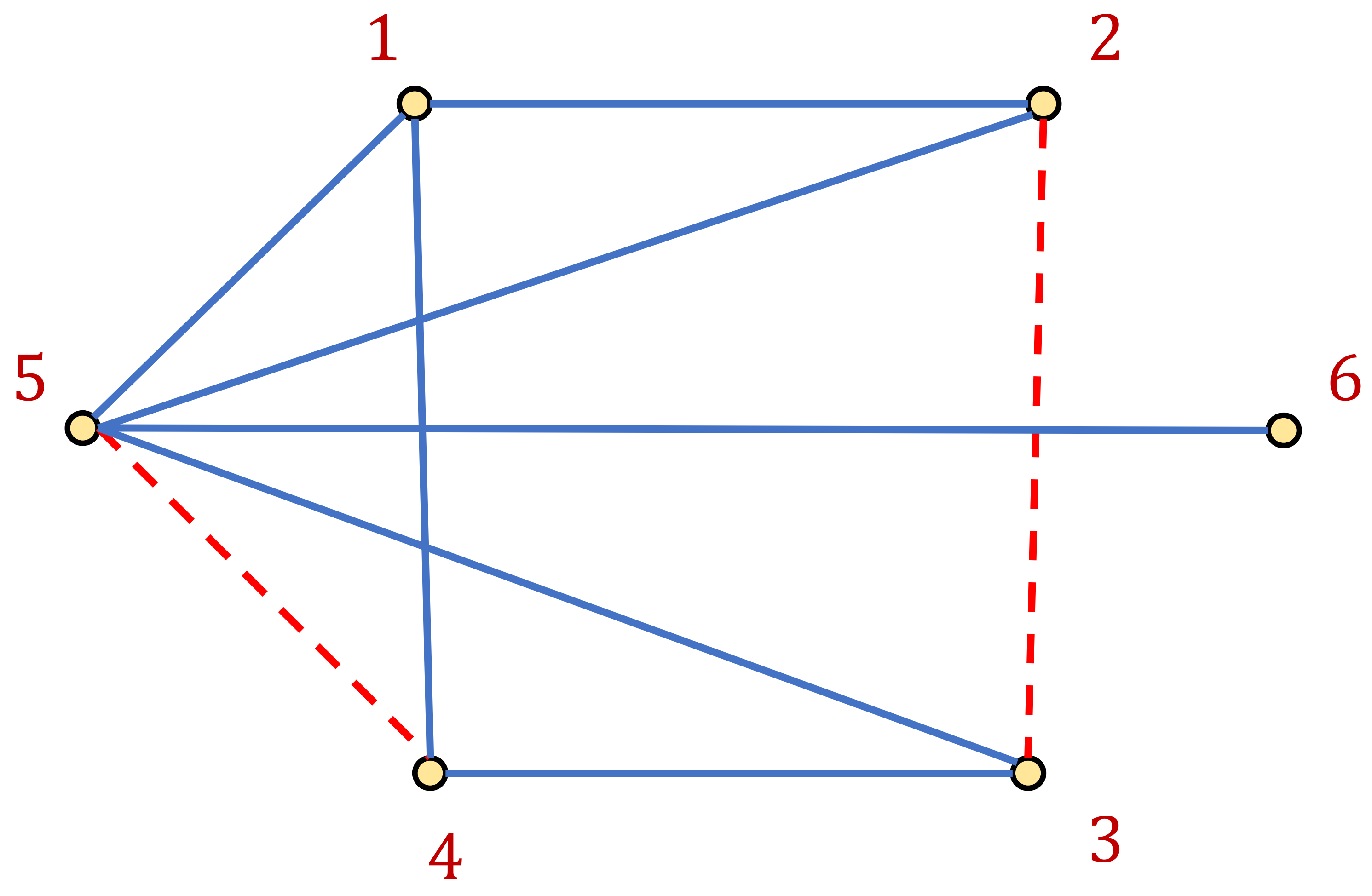
Maximal Matching

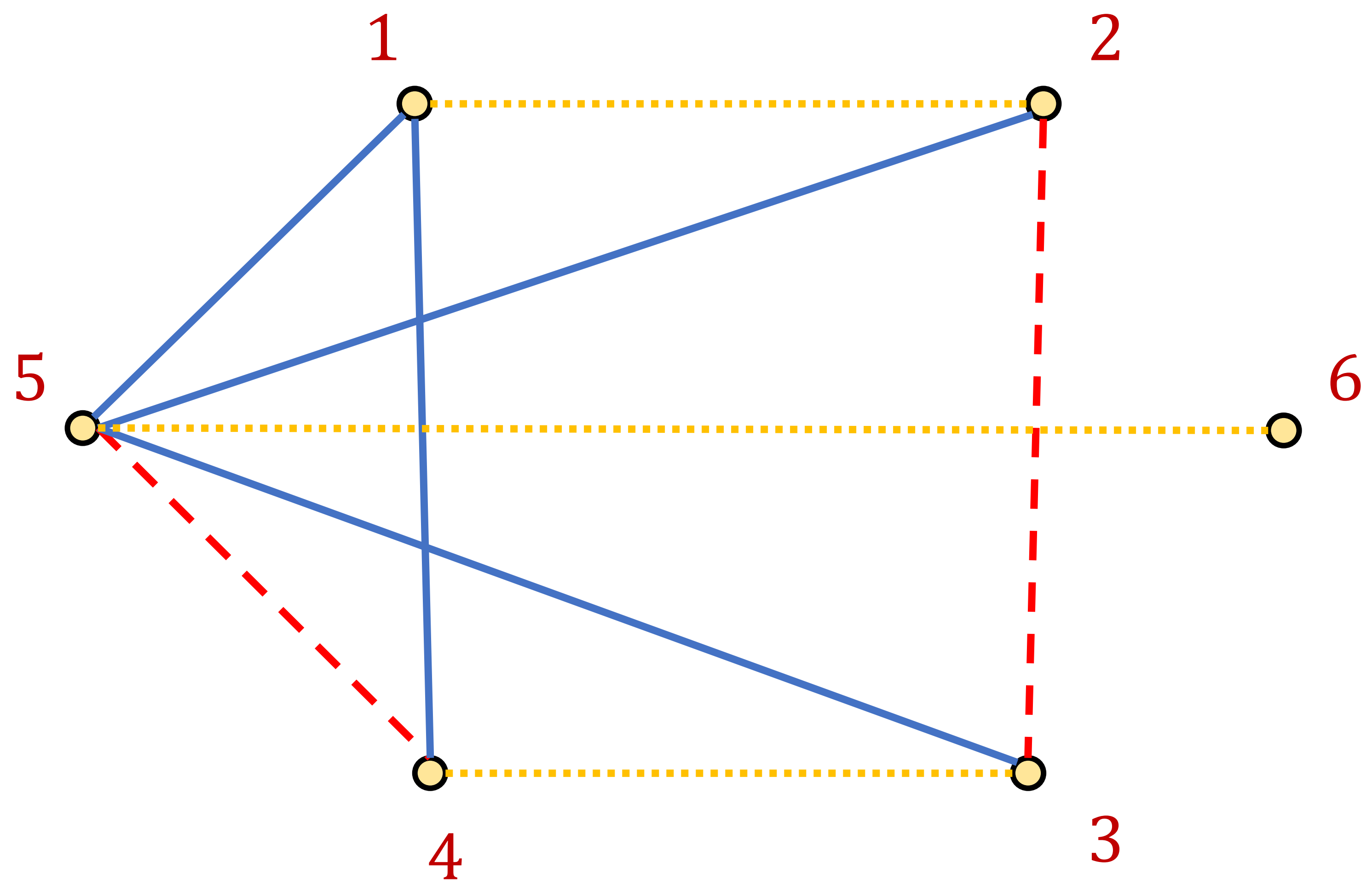
- What if we just wanted to find a maximal matching?

- Greedy algorithm: Add each unmatched edge e in the stream to the matching M

Maximal Matching

- Claim: Each maximal matching is a 2-approximation to the maximum matching


1
2
5
6
4
3


1
2
5
6
4
3

Maximal Matching

- Claim: Each maximal matching is a 2-approximation to the maximum matching

- Observation: Each edge e' of M' can be incident to at most 2 edges of M^*

- Observation: Each edge e of the maximum matching M^* must be incident to some edge e' of any maximal matching M'

Maximal Matching

- Claim: Each maximal matching is a 2-approximation to the maximum matching

- Intuition: Each edge e of the maximum matching M^* must be incident to some edge e' of any maximal matching M' BUT each edge e' of M' can be incident to at most 2 edges of M^*

Maximal Matching

- Charging argument: Give each edge e' of the maximal matching M' two dollars

Maximal Matching

- Charging argument: Give each edge e' of the maximal matching M' two dollars

- Observation: Each edge e' of M' can be incident to at most 2 edges of M^*
- Enough money for each edge e' to pay for the adjacent edges in M^*

- Observation: Each edge e of the maximum matching M^* must be incident to some edge e' of any maximal matching M'
- All edges of M^* have been paid by some edge of M'

Maximal Matching

- For each edge e' of M', let $N_1(e')$ and $N_2(e')$ be the incident edges of M^* (we can say $N_2(e')$ is empty if e' is not incident to two edges)

$$\begin{aligned} 2|M'| &= \Sigma_{e' \in M'} 2|e'| \\ &\geq \Sigma_{e' \in M'} [|N_1(e')| + |N_2(e')|] \\ &\geq \Sigma_{e \in M^*} |e| \\ &= |M^*| \end{aligned}$$

Matchings in the Semi-Streaming Model

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

Matchings in the Semi-Streaming Model

- In a weighted graph, each edge can have weights in $\{1, \ldots, N\}$ for some $N = \mathrm{poly}(n)$
- The weight of a matching is the sum of the weights of the edges

- Can we run the greedy algorithm? NO / YES

Matchings in the Semi-Streaming Model

- For $i = 0,1,\dots,\log_{(1+\varepsilon)} N$, let S_i be the substream that contains edges with weights between $(1+\varepsilon)^i$ and $(1+\varepsilon)^{i+1}$
- Let M_i be a maximal matching obtained by using greedy algorithm on S_i
- Let M be obtained by greedily adding edges in M_i for $i = \log_{(1+\varepsilon)} N, \dots, 1,0$

- Intuition: Each edge e of matching M can "block" at most two edges of M_i, each of these two edges can "block" at most two edges in the best matching M^*

Matchings in the Semi-Streaming Model

- For $i = 0,1,\dots,\log_{(1+\varepsilon)} N$, let S_i be the substream that contains edges with weights between $(1+\varepsilon)^i$ and $(1+\varepsilon)^{i+1}$
- Let M_i be a maximal matching obtained by using greedy algorithm on S_i
- Let M be obtained by greedily adding edges in M_i for $i = \log_{(1+\varepsilon)} N, \dots, 1,0$

- Algorithm is a $(4+\varepsilon)$-approximation to the maximum weighted matching in the semi-streaming model [CrouchStubbs14]

Matchings in the Semi-Streaming Model

- 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 $C < 2$?

Matchings in the Semi-Streaming Model

- In a weighted graph, each edge can have weights in $\{1, \ldots, N\}$ for some $N = \text{poly}(n)$
- The weight of a matching is the sum of the weights of the edges

- OPEN: Is it possible to achieve C-approximation to the maximum weighted matching using $n \cdot \text{polylog}(n)$ space for $C < 2$?
- Algorithm: There exists a $(2 + \varepsilon)$-approximation to the maximum weighted matching in the semi-streaming model [PazSchwartzman17]

Matchings in the Semi-Streaming Model

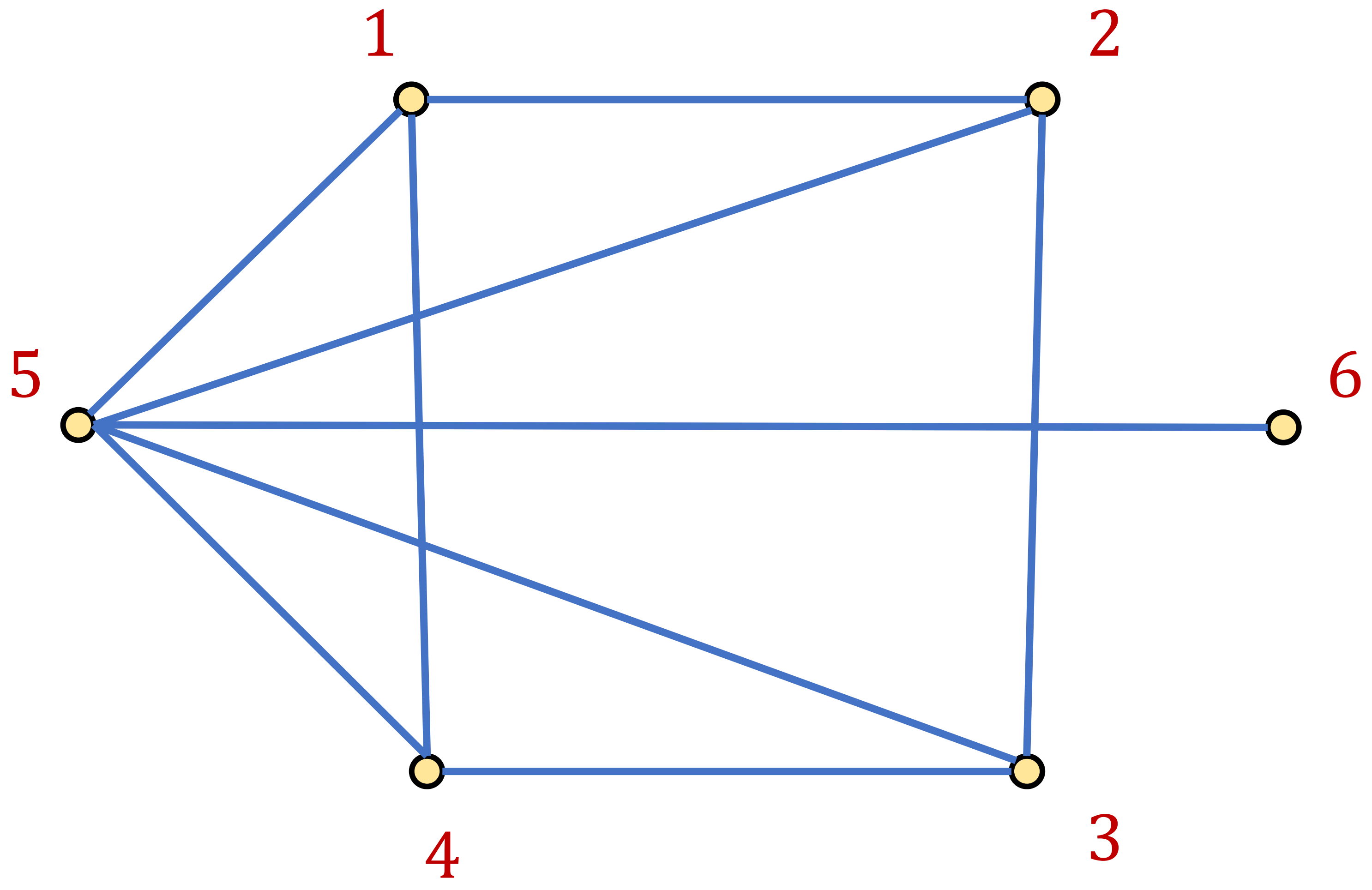
- A function is submodular if it satisfies the "diminishing gains" property: $f(S \cup \{x\}) - f(S) \geq f(T \cup \{x\}) - f(T)$ for all $T \subseteq S$, x
- Maximize a submodular function across all matchings on a graph

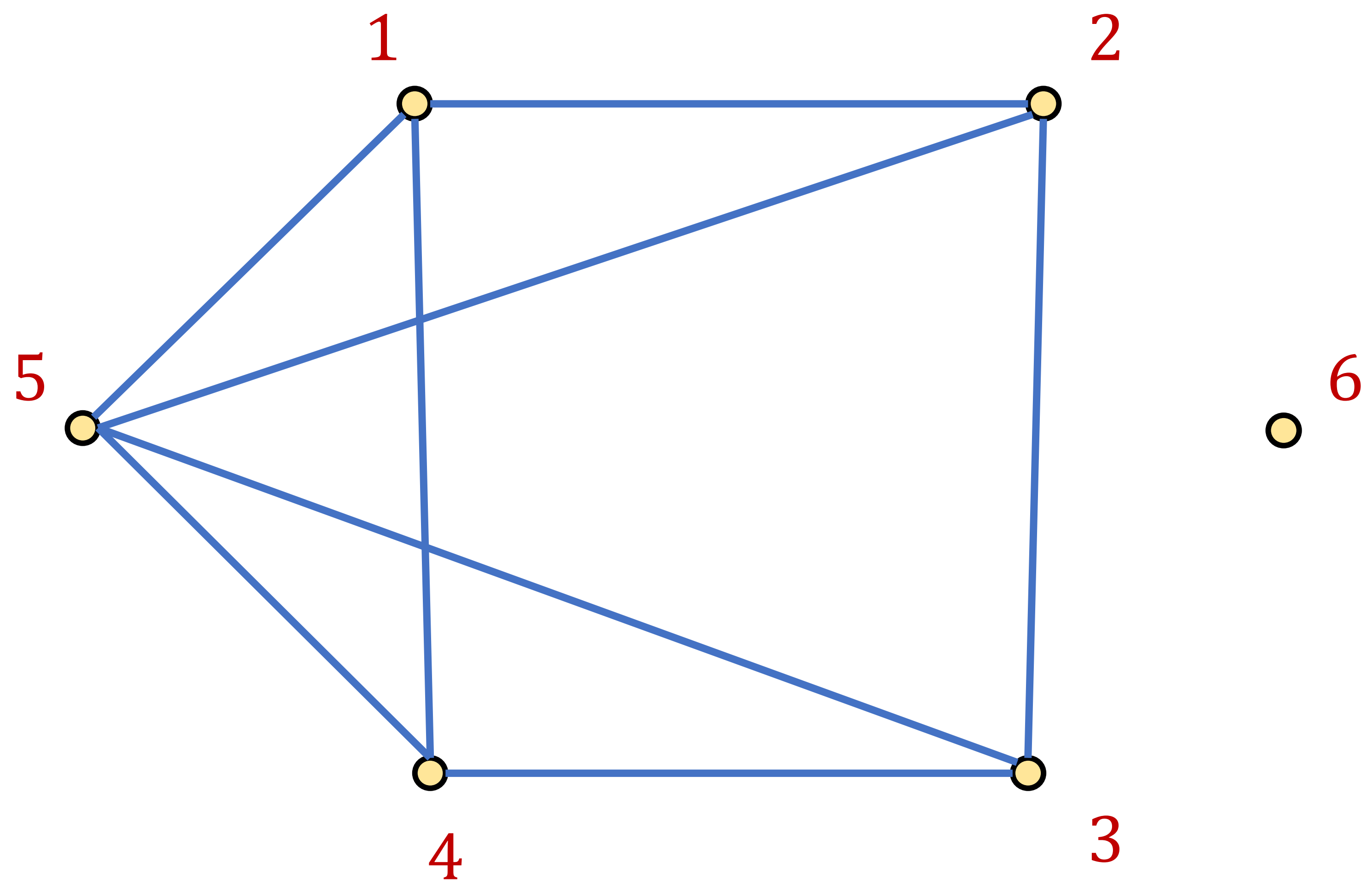
- OPEN: Is it possible to achieve C-approximation to the maximum submodular matching using $n \cdot \text{polylog}(n)$ space for $C < 2$?
- Algorithm: There exists a $\left(3 + 2\sqrt{2}\right) \approx 5.828$-approximation to the maximum weighted matching in the semi-streaming model [LevinWajc21]

Connectivity

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

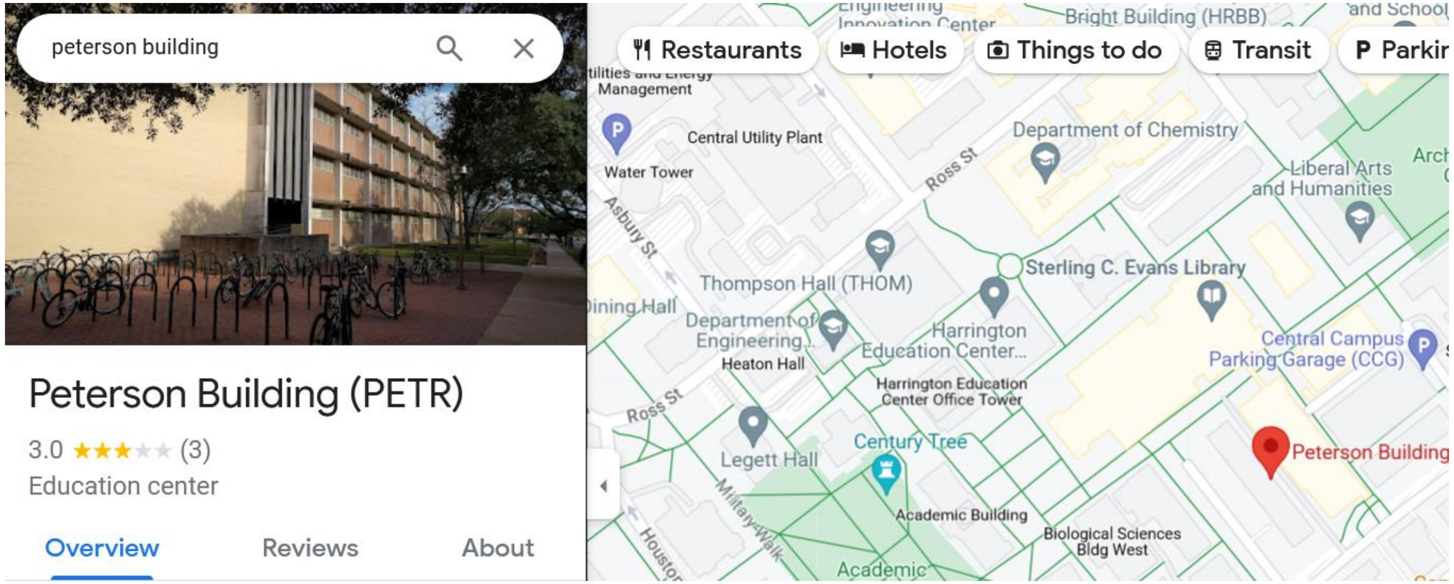
- Goal: Given a graph G, determine whether G is a connected graph

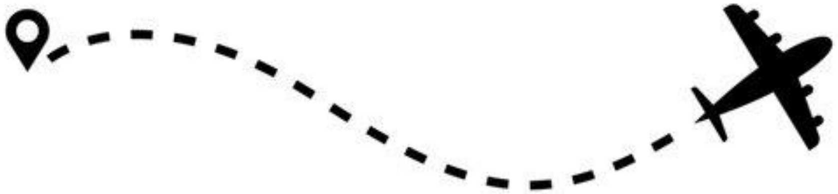

1
2
5
6
4
3


1
2
5
6
4
3

Connectivity

- Transportation networks: Analyzing the connectivity of transportation networks, e.g., roads, railways, flight routes, is critical for optimizing routes, planning public transportation, identifying congested areas, and ensuring efficient travel





Connectivity

- Electrical power grids: Determining the connectivity of an electric power grid is essential for ensuring a reliable and resilient power supply. Identifying isolated components helps in quickly restoring power after outages.

Spanning Forest

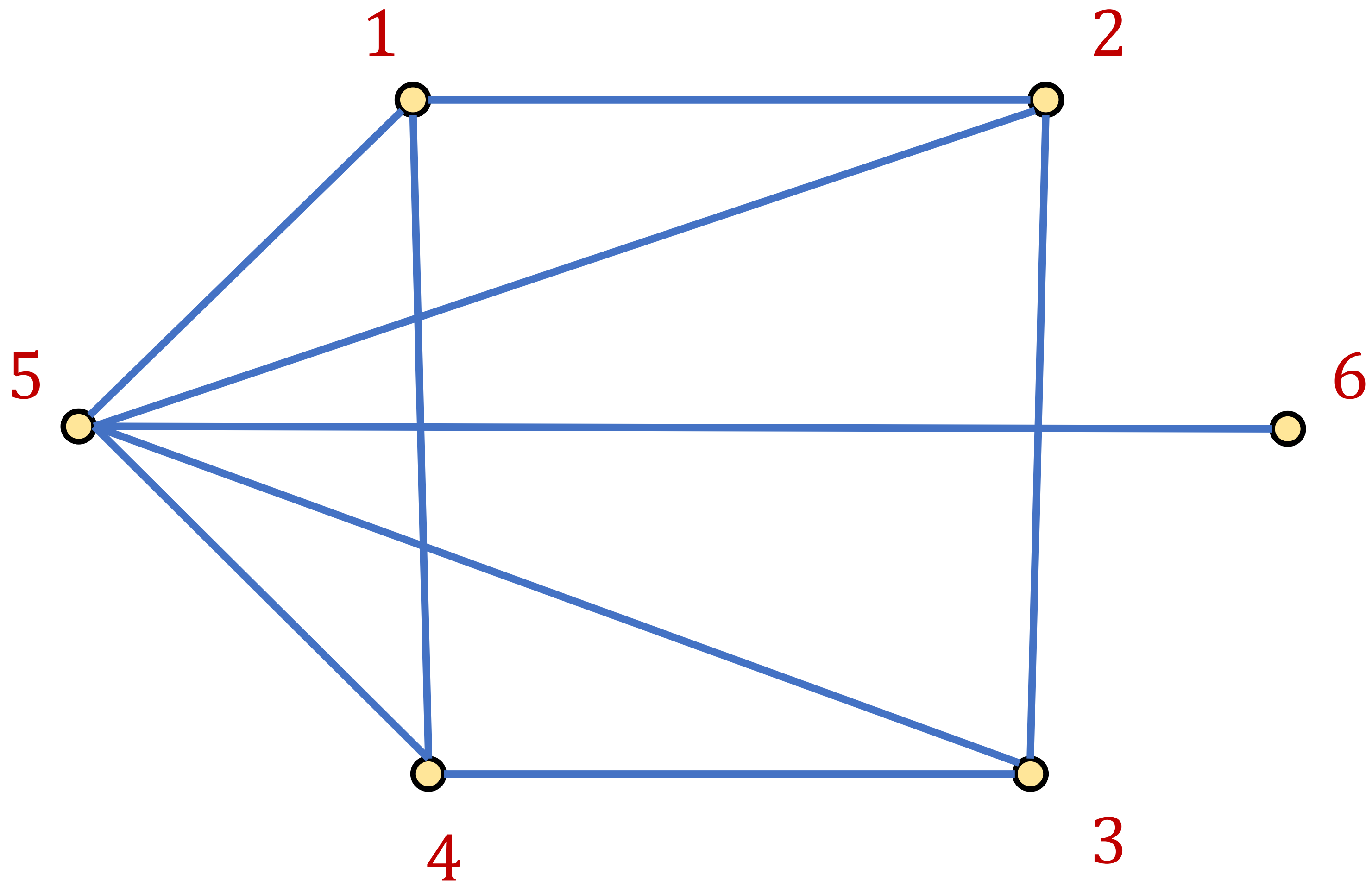
- Spanning tree: A subgraph of G that is a tree and contains all the vertices of the graph G
- Spanning forest: A subgraph of G that is a union of trees that contains all the vertices of the graph G

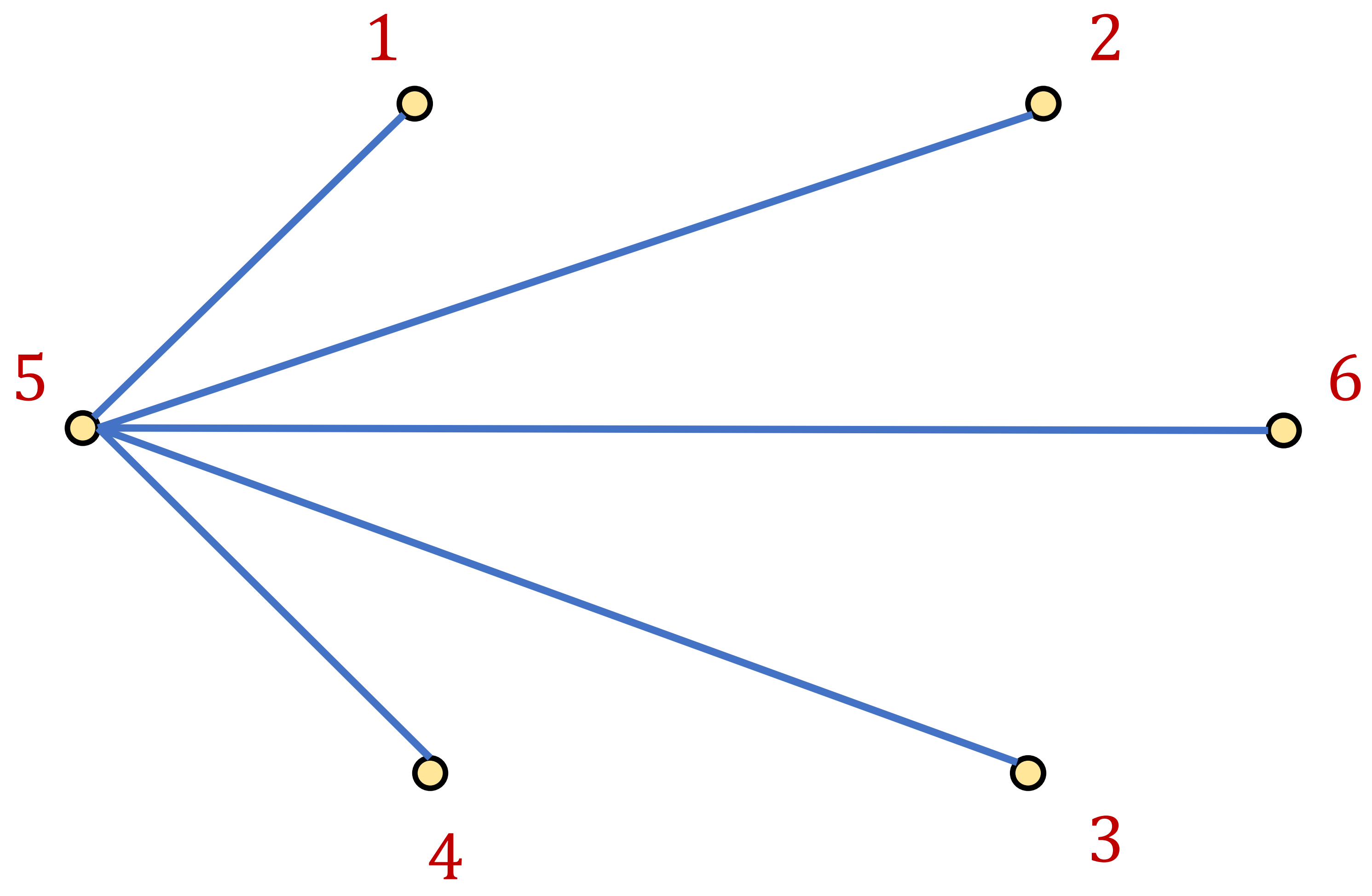
Spanning Forest

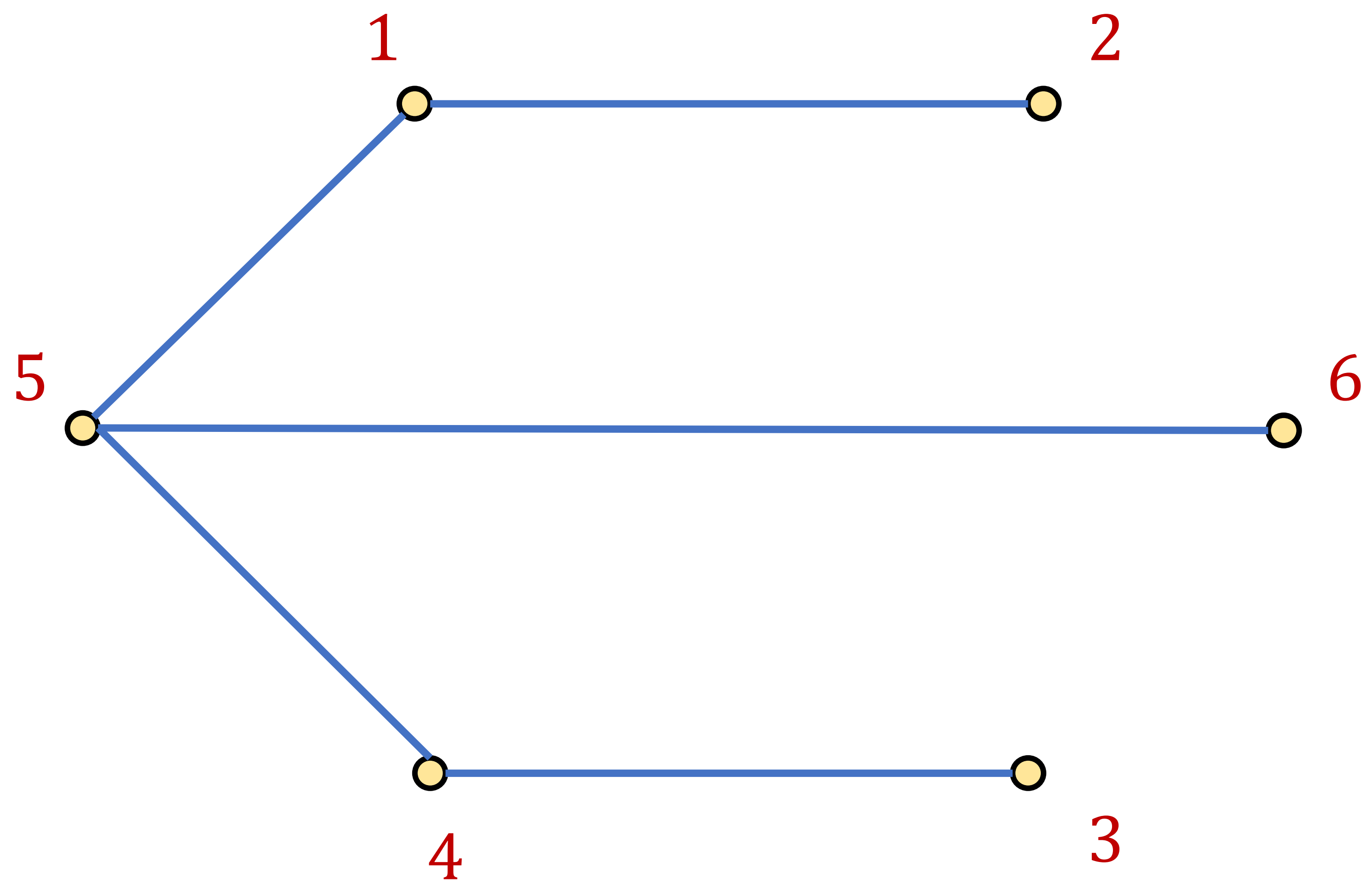
- Spanning tree: A subgraph of G that is a tree and contains all the vertices of the graph G

- Spanning forest: A subgraph of G that is a union of trees that contains all the vertices of the graph G

- Observation: A graph G is connected if and only if G has a spanning tree


1
2
5
6
4
3


1
2
5
6
4
3


1
2
5
6
4
3

Spanning Tree

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

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

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

Connectivity

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