CSCE 658: Randomized Algorithms

Lecture 11

Samson Zhou

Class Logistics

- March 5: Lecture canceled, i.e., do NOT show up to HRBB 126 (unless you want to see an empty classroom)

Previously in the Streaming Model

- Reservoir sampling
- Heavy-hitters
  - Misra-Gries
  - CountMin
  - CountSketch
- Moment estimation
  - AMS algorithm
- Sparse recovery
- Distinct elements estimation

Reservoir Sampling

- Suppose we see a stream of elements from $[n]$. How do we uniformly sample one of the positions of the stream?

47 72 81 10 14 33 51 29 54 9 36 46 10

Heavy-Hitters (Frequent Items)

- Given a set S of m elements from $[n]$, let f_i be the frequency of element i. (How often it appears)
- Let L_p be the norm of the frequency vector:

$$L_p = \left(f_1^p + f_2^p + \cdots + f_n^p\right)^{1/p}$$

- Goal: Given a set S of m elements from $[n]$ and a threshold ε, output the elements i such that $f_i > \varepsilon\, L_p$...and no elements j such that $f_j < \frac{\varepsilon}{2}\, L_p$ (we saw algorithms for $p = 1$ and $p = 2$)
- Motivation: DDoS prevention, iceberg queries

Frequency Moments (L_p Norm)

- Given a set S of m elements from $[n]$, let f_i be the frequency of element i. (How often it appears)
- Let F_p be the frequency moment of the vector:

$$F_p = f_1^p + f_2^p + \cdots + f_n^p$$

- Goal: Given a set S of m elements from $[n]$ and an accuracy parameter ε, output a $(1+\varepsilon)$-approximation to F_p
- Motivation: Entropy estimation, linear regression

The Streaming Model

- So far, all questions have been *statistical*

- What other questions can be asked? (Think in general, outside of the streaming model)

The Streaming Model

- So far, all questions have been *statistical*

- What other questions can be asked? (Think in general, outside of the streaming model)

- Algebraic, geometric

The Streaming Model

- So far, all questions have been *statistical*

- What other questions can be asked? (Think in general, outside of the streaming model)

- Algebraic, geometric

TODAY

Graph Theory

- Suppose we have a graph G with vertex set V and edge set E
- Let $V = [n]$ for simplicity, so each vertex is an integer from 1 to n
- Then each edge $e \in E$ can be written as $e = (u, v)$ for $u, v \in [n]$
- In other words, each edge is a pair of integers from 1 to n

Graph Theory

- For today, we will assume a simple, undirected, unweighted graph

- Graph has no self-loops, no multi-edges

- Edges are undirected

- Each edge has weight 1

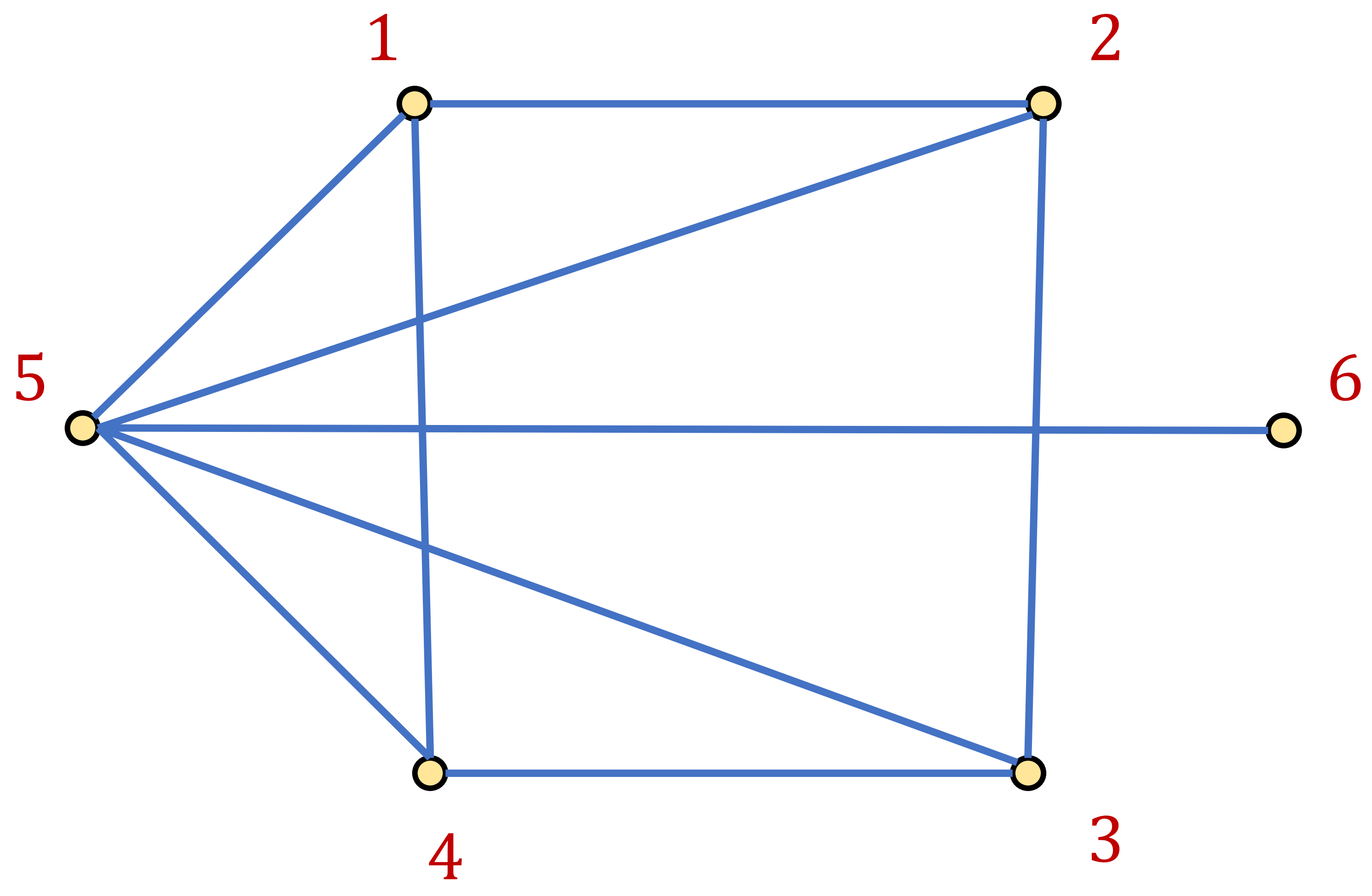
Semi-streaming Model

- 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

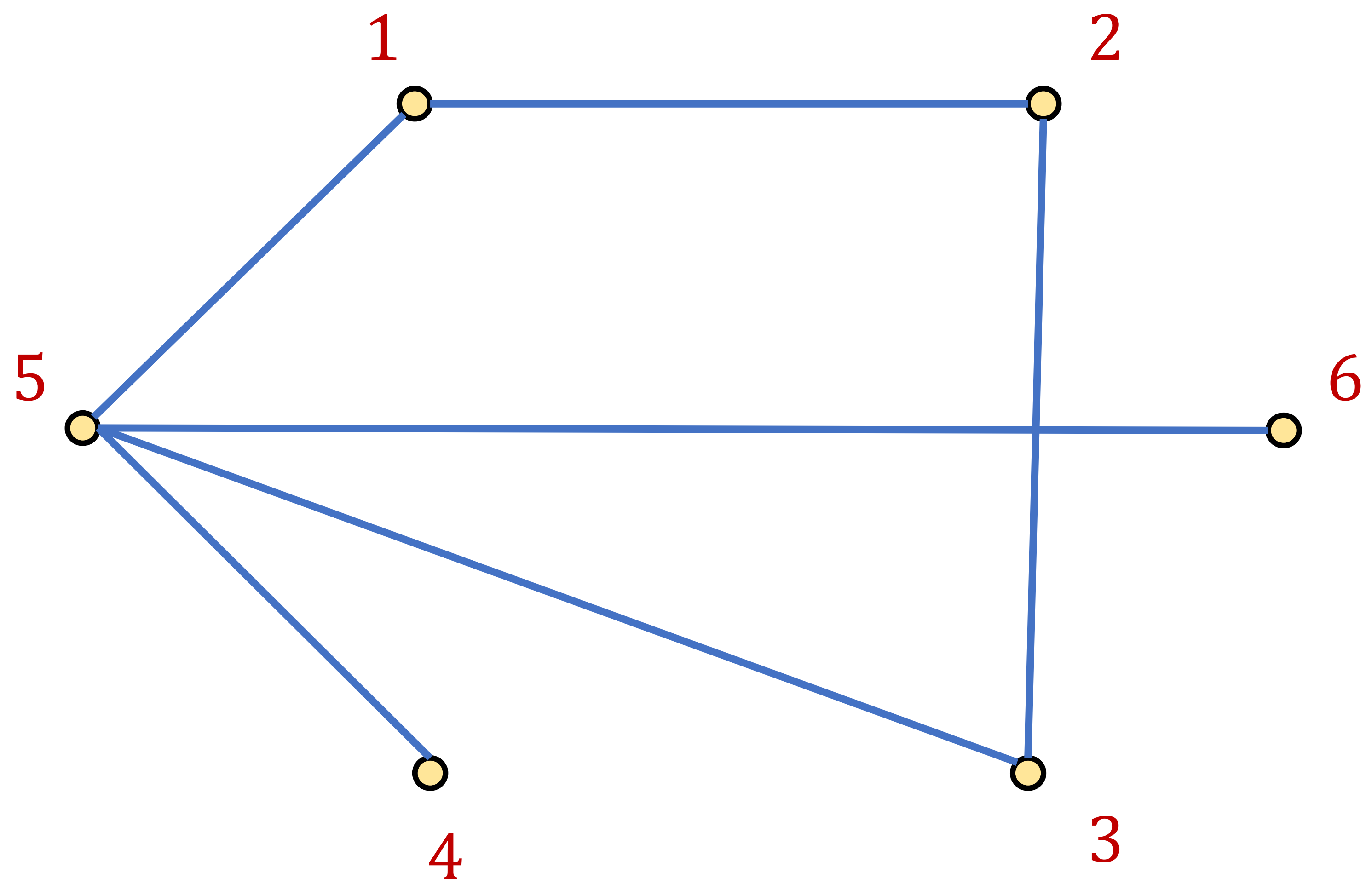
- Enough to store things like a matching, a spanning tree, NOT enough to store entire graph, since m can be as large as $O(n^2)$

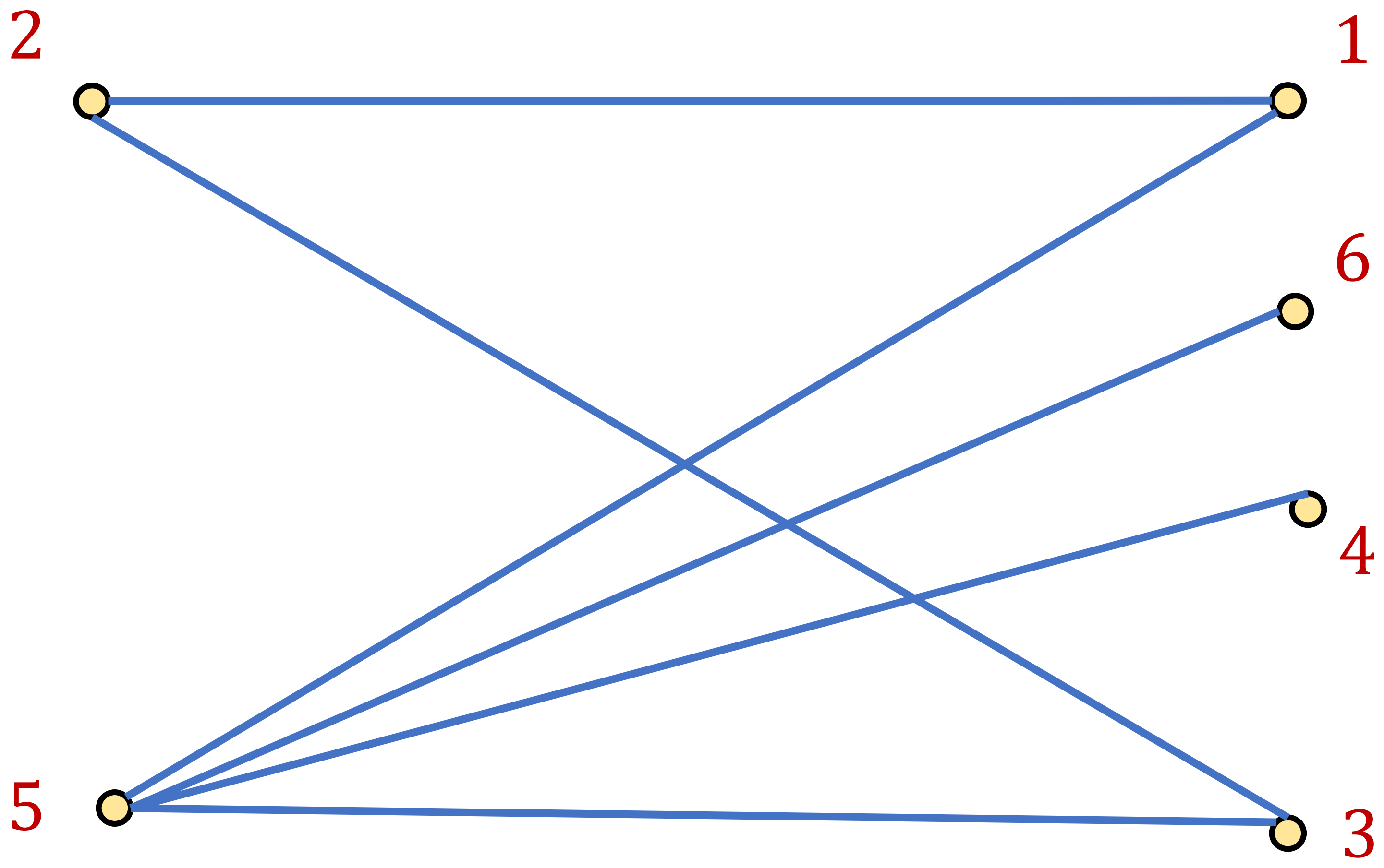

1
2
5
6
4
3

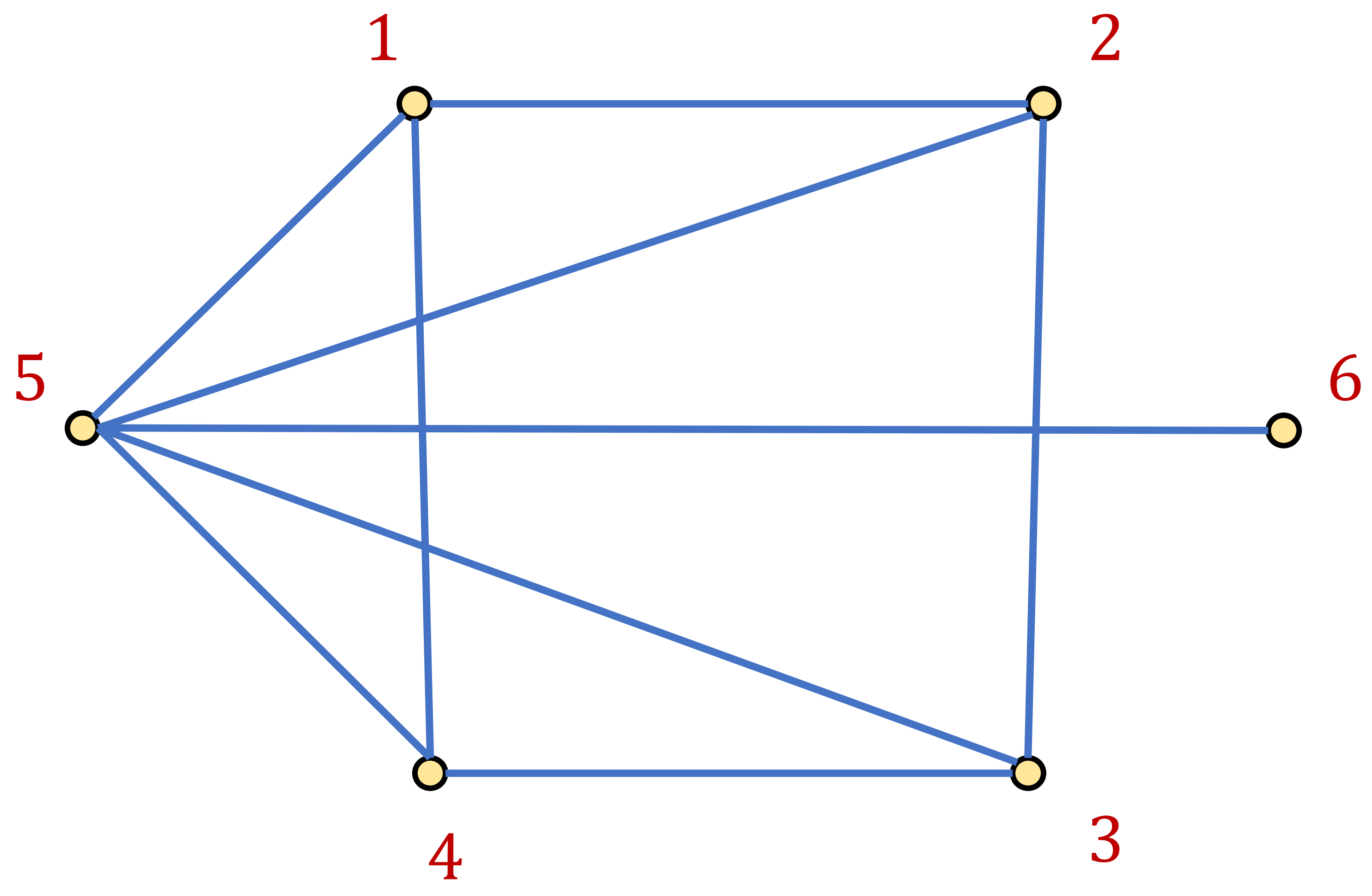
Bipartiteness

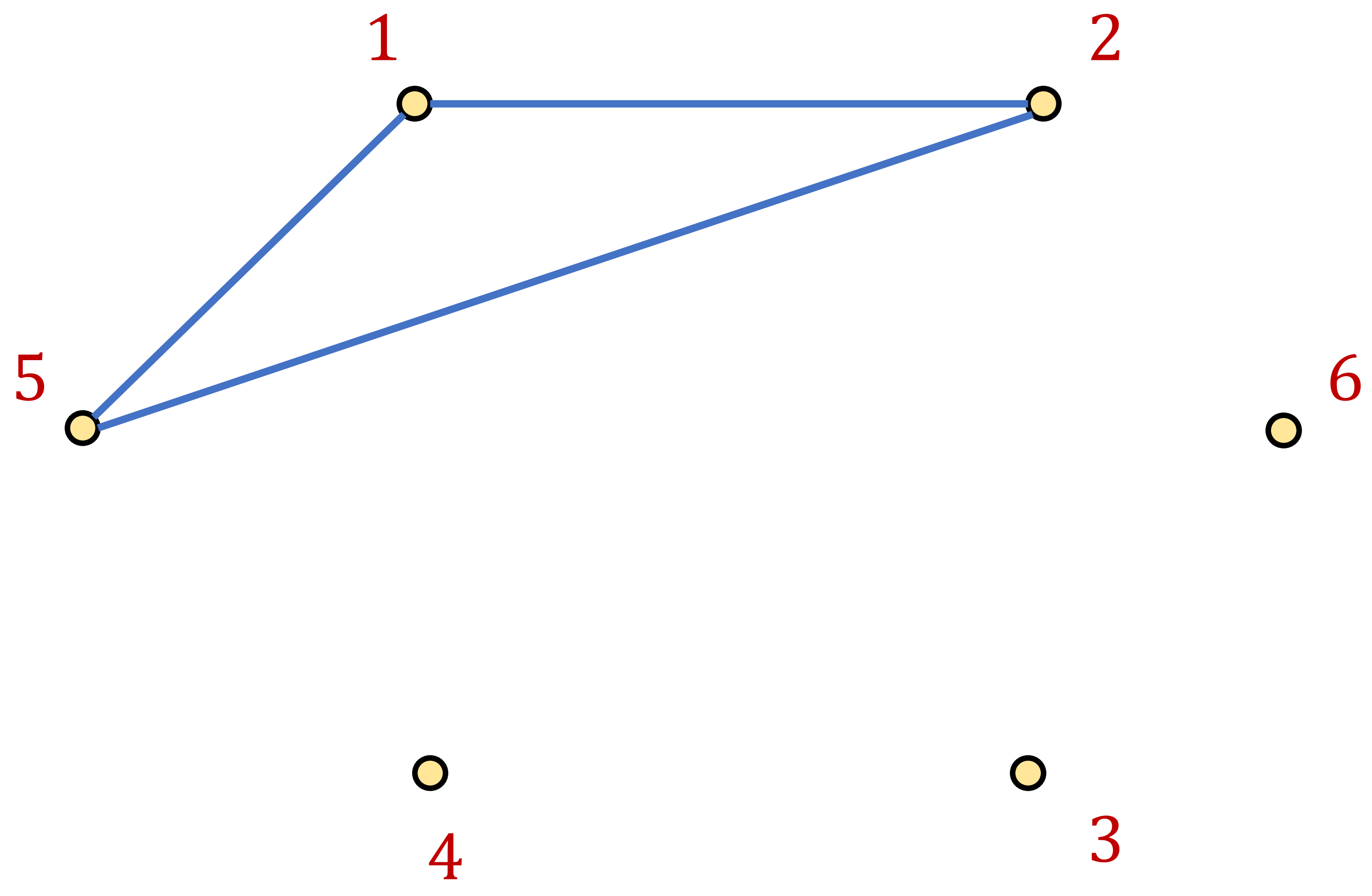
- Bipartite graph: Graph can be partitioned into two disjoint sets L and R so that every edge is between a vertex in L and a vertex in R

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


1
2
5
6
4
3

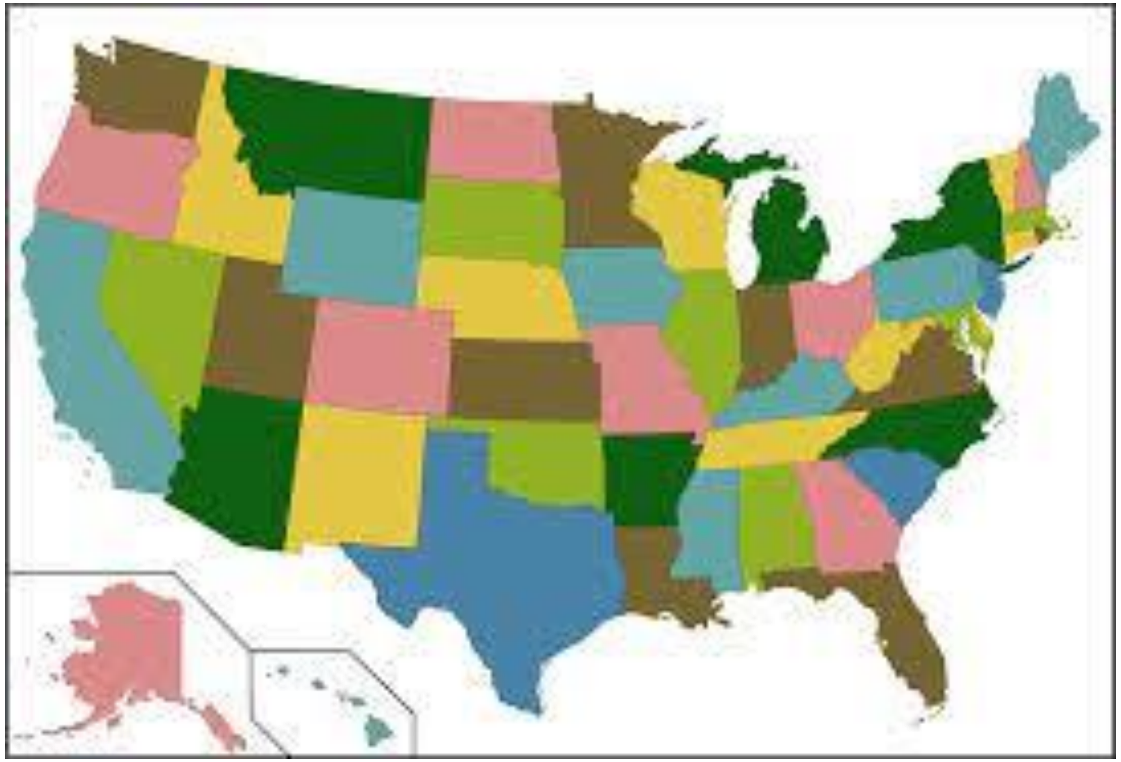

2
1
6
4
5
3

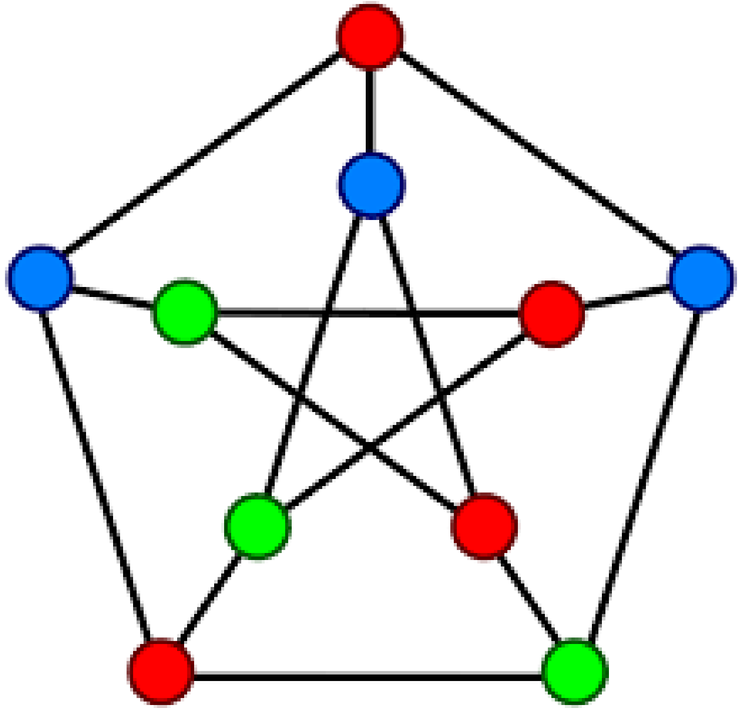

1
2
5
6
4
3


1
2
5
6
4
3

Applications for Bipartiteness Testing

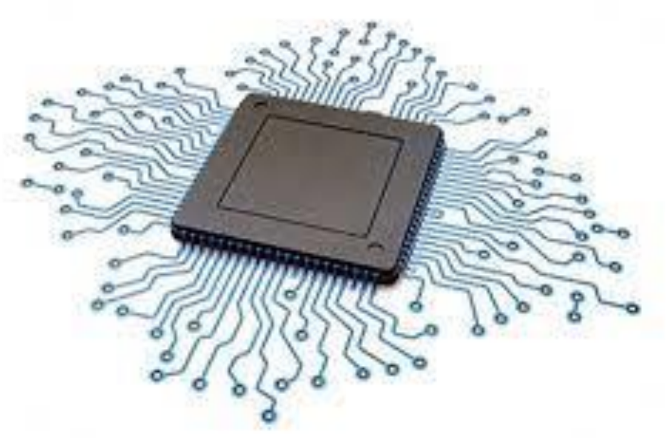
- Graph coloring: You want to color a graph such that no neighboring items share the same color





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

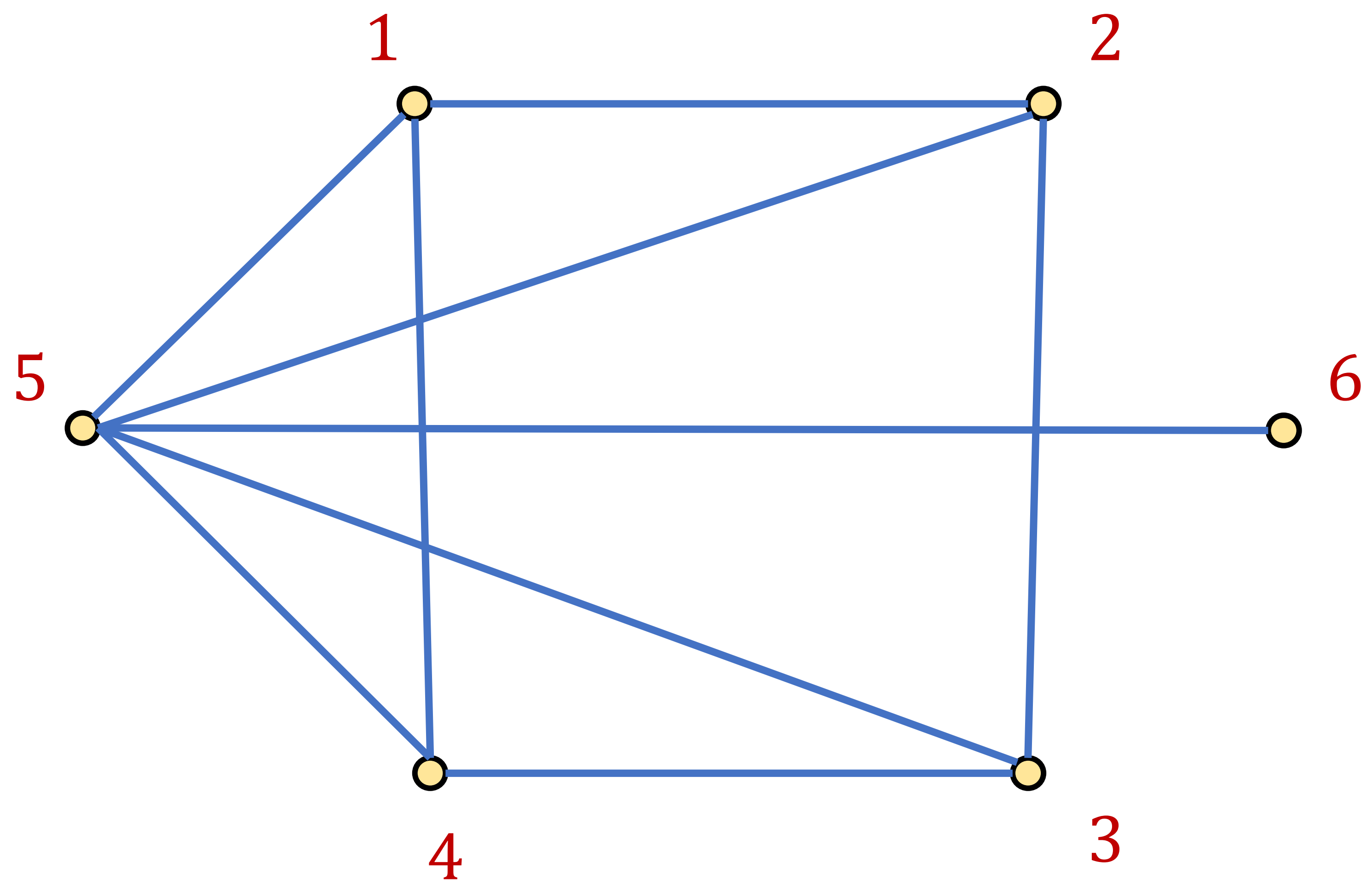
Bipartiteness

- How to perform bipartiteness testing in the central setting?

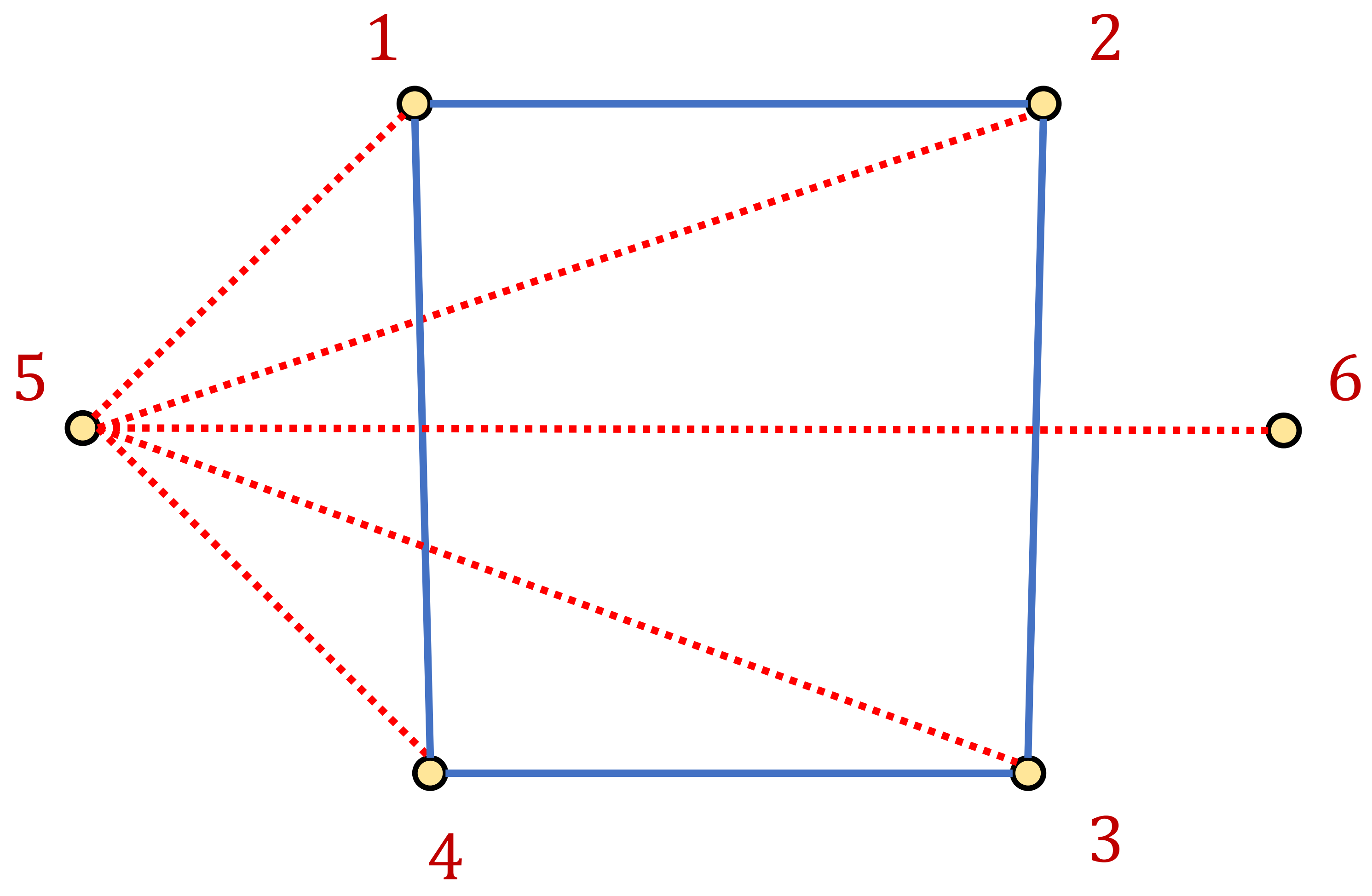
Bipartiteness

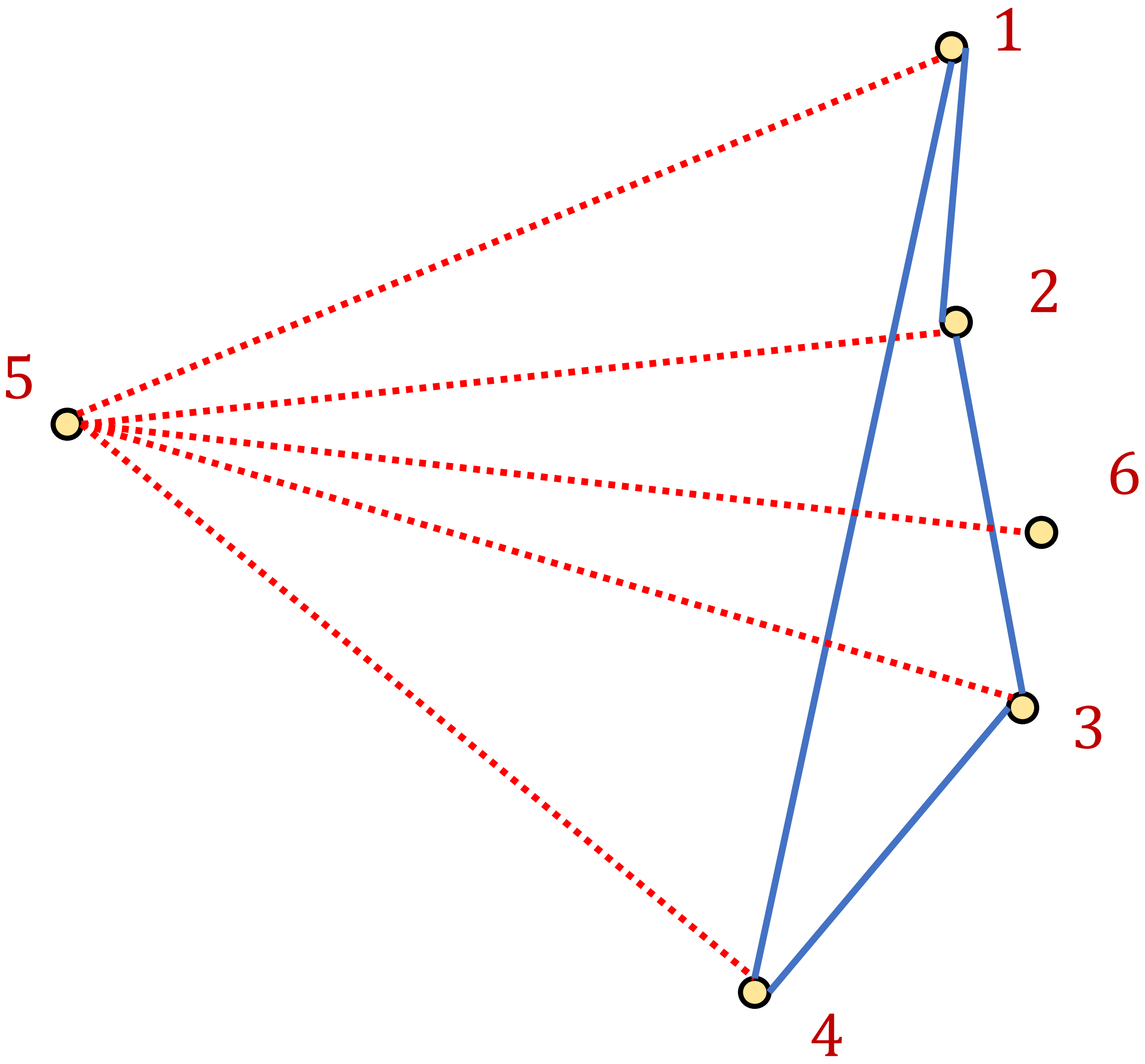
- How to perform bipartiteness testing in the central setting?

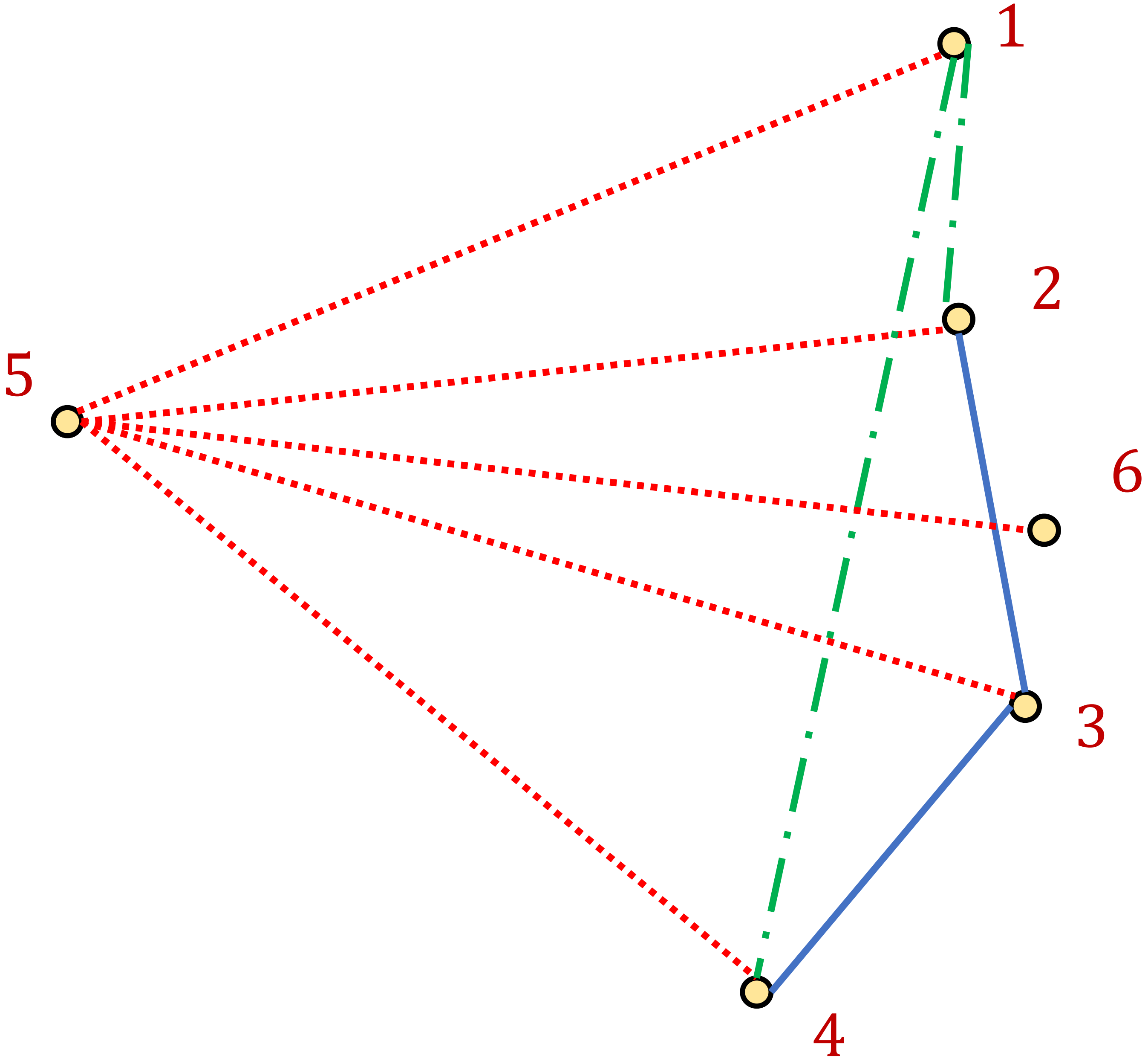
- Start at arbitrary vertex, run BFS, and assign alternating levels to different side until there is a contradiction



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

- 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

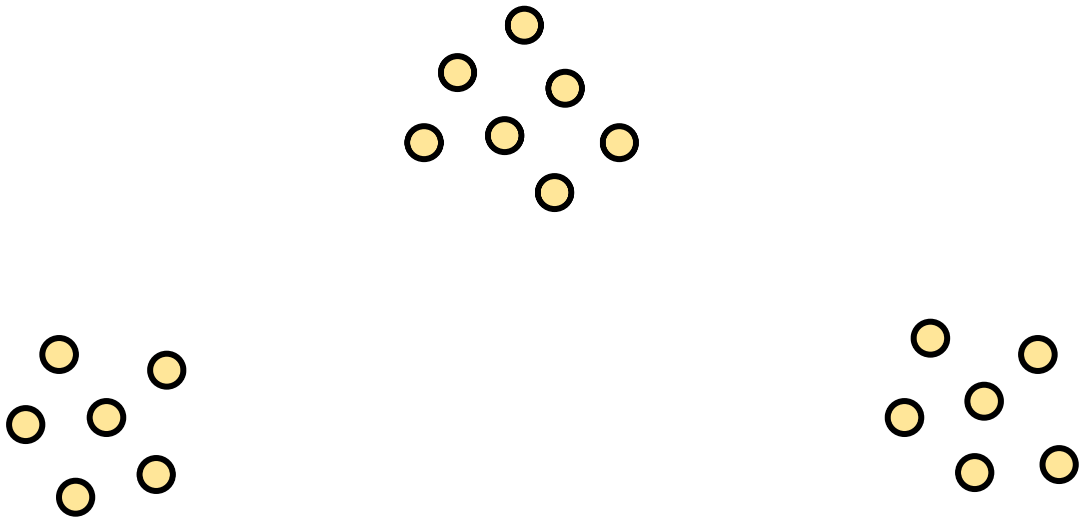
- Algorithm maintains a tree (because it does not add any edges that would create cycles)

- How many edges does the algorithm keep?

Bipartiteness in the Streaming Model

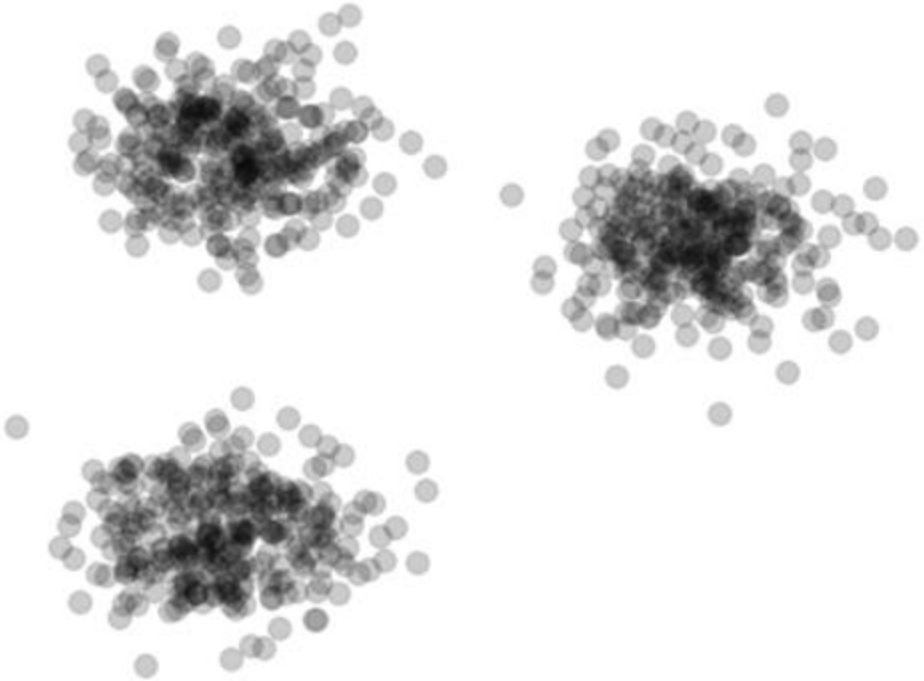
- Algorithm maintains a tree (because it does not add any edges that would create cycles)

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



Clustering

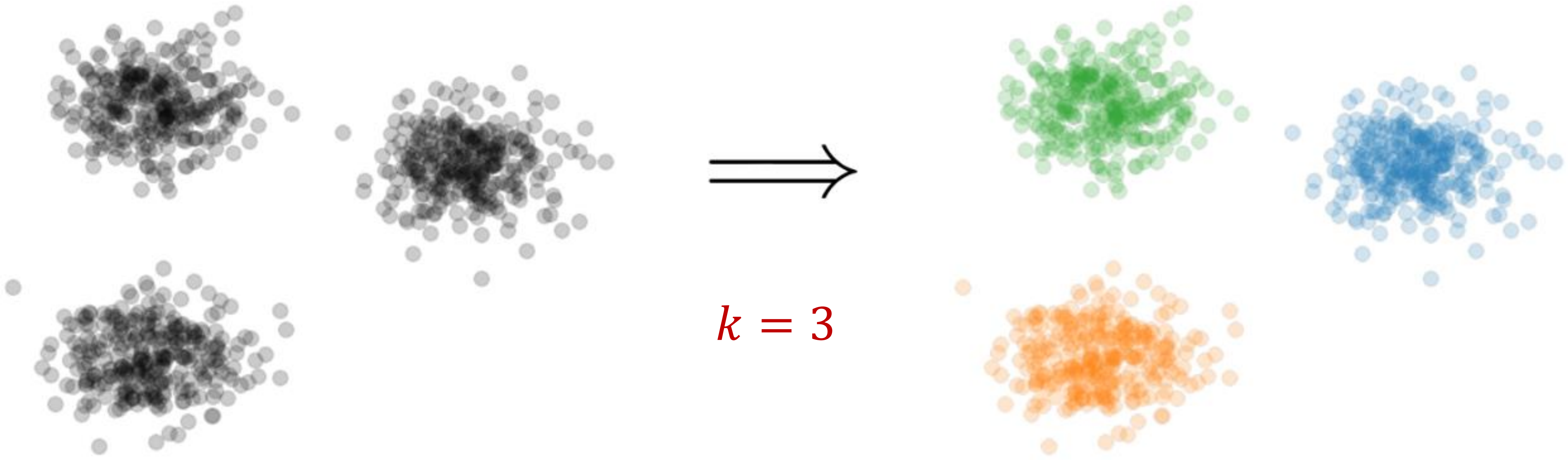
- Goal: Given input dataset X, partition X so that “similar” points are in the same cluster and “different” points are in different clusters



$\Longrightarrow$

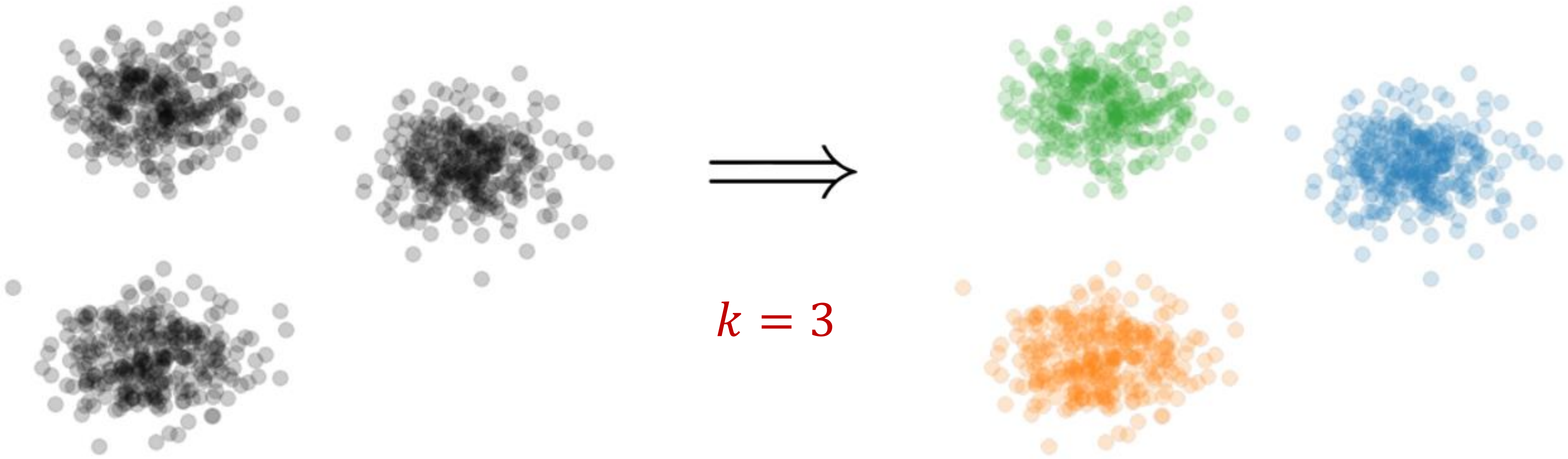
k-Clustering

- Goal: Given input dataset X, partition X so that "similar" points are in the same cluster and "different" points are in different clusters
- There can be at most k different clusters



k-Clustering

- Question: How do we measure the “quality” of each clustering?



k-Clustering

- Question: How do we measure the "quality" of each clustering?
- Assign a "center" c_i to each cluster
- Have a cost function induced by c_i for all of the points P_i assigned to cluster i

k-Clustering

- Question: How do we measure the "quality" of each clustering?

- Assign a "center" c_i to each cluster

- Have a cost function induced by c_i for all of the points P_i assigned to cluster i
  - Assume points are in metric space with distance function $\mathrm{dist}(\cdot,\cdot)$
  - Define $\mathrm{Cost}(P_i, c_i)$ to be a function of $\{\mathrm{dist}(x, c_i)\}_{x \in P_i}$

k-Clustering

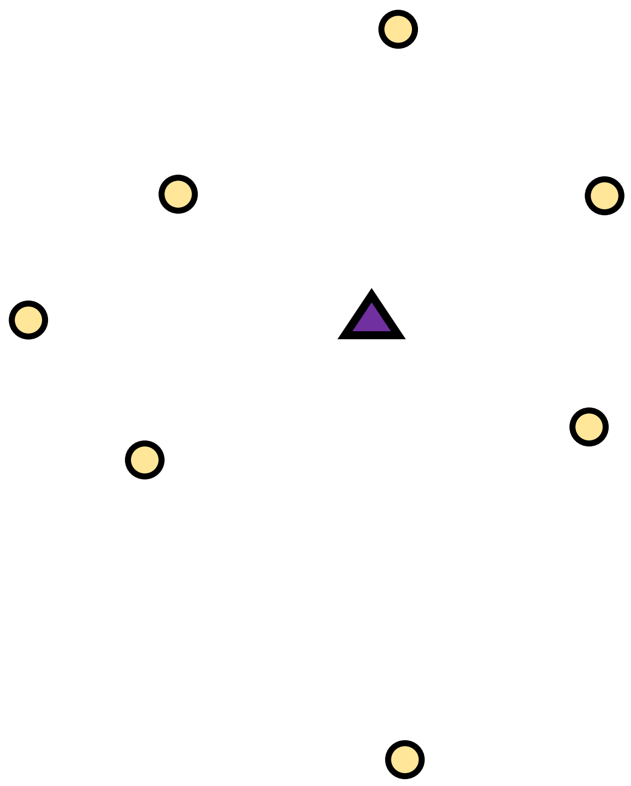
- Question: How do we measure the "quality" of each clustering?

- Have a cost function induced by c_i for all of the points P_i assigned to cluster i
  - Define $\mathrm{Cost}(P_i, c_i)$ to be a function of $\{\mathrm{dist}(x, c_i)\}_{x \in P_i}$

- Suppose the set of centers is $C = \{c_1, \dots, c_k\}$
  - Define clustering cost $\mathrm{Cost}(X, C)$ to be a function of $\{\mathrm{dist}(x, C)\}_{x \in C}$

k-Clustering

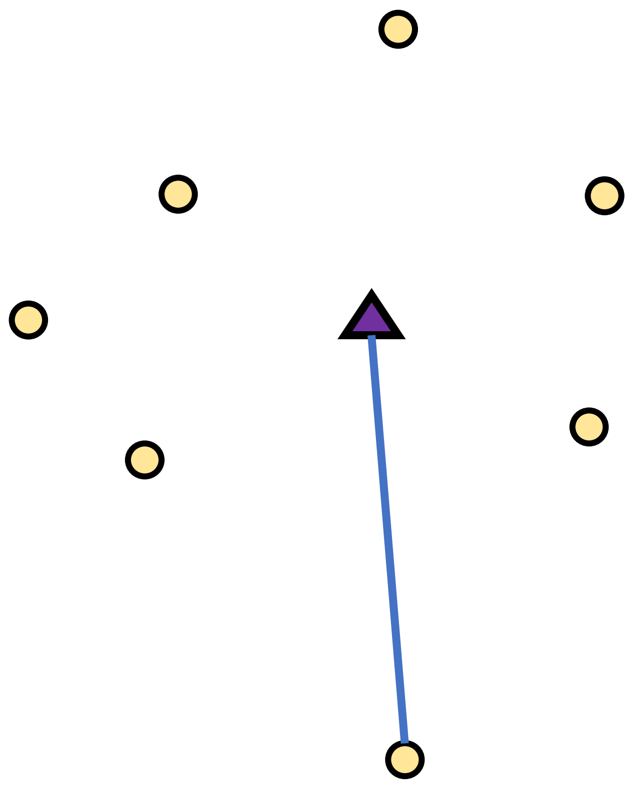
- Define clustering cost $\text{Cost}(X, C)$ to be a function of $\{\text{dist}(x, C)\}_{x \in C}$



k-Clustering

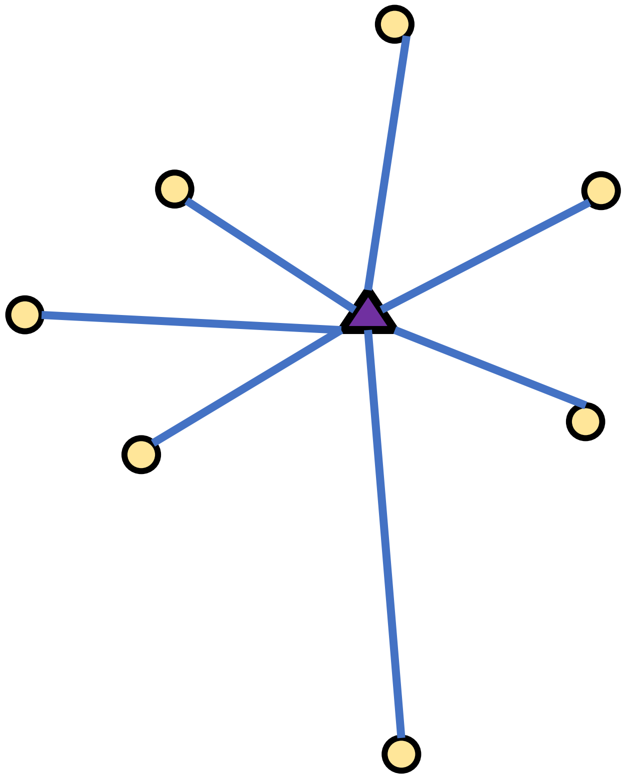
- Define clustering cost $\mathrm{Cost}(X, C)$ to be a function of $\{\mathrm{dist}(x, C)\}_{x \in C}$

- k-center: $\mathrm{Cost}(X, C) = \max_{x \in X} \mathrm{dist}(x, C)$



k-Clustering

- Define clustering cost $\mathrm{Cost}(X, C)$ to be a function of $\{\mathrm{dist}(x, C)\}_{x \in C}$

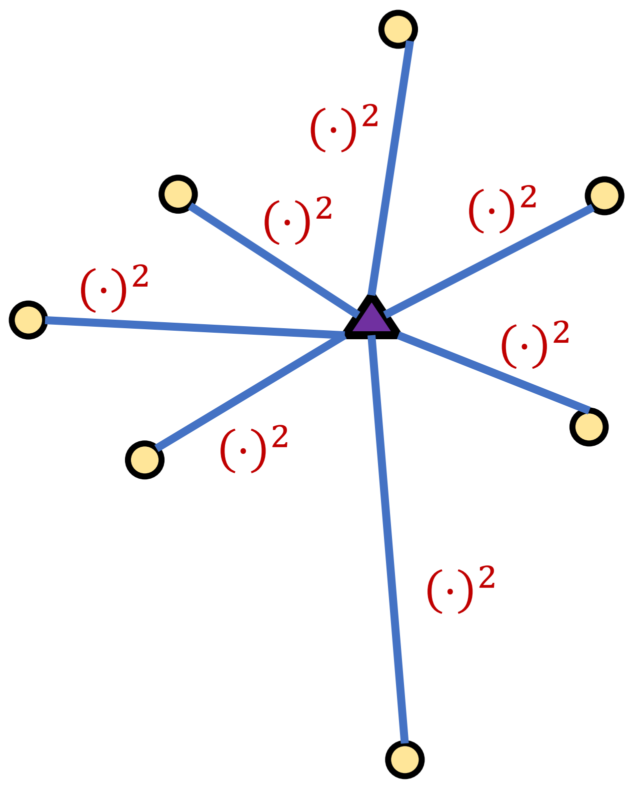


- k-center: $\mathrm{Cost}(X, C) = \max_{x \in X} \mathrm{dist}(x, C)$
- k-median: $\mathrm{Cost}(X, C) = \sum_{x \in X} \mathrm{dist}(x, C)$

k-Clustering

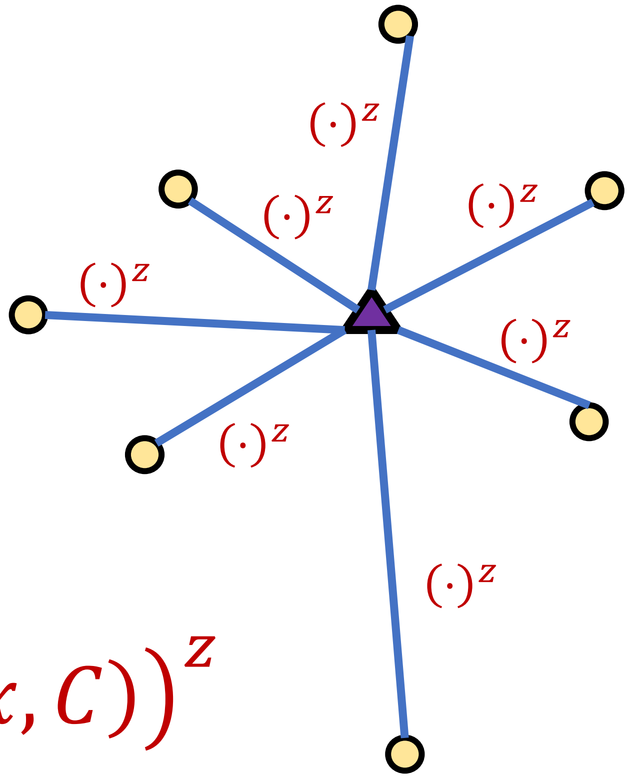
- Define clustering cost $\mathrm{Cost}(X, C)$ to be a function of $\{\mathrm{dist}(x, C)\}_{x \in C}$

- k-center: $\mathrm{Cost}(X, C) = \max_{x \in X} \mathrm{dist}(x, C)$
- k-median: $\mathrm{Cost}(X, C) = \sum_{x \in X} \mathrm{dist}(x, C)$
- k-means: $\mathrm{Cost}(X, C) = \sum_{x \in X} \left(\mathrm{dist}(x, C)\right)^2$



k-Clustering

- Define clustering cost $\mathrm{Cost}(X, C)$ to be a function of $\{\mathrm{dist}(x, C)\}_{x \in C}$
- k-center: $\mathrm{Cost}(X, C) = \max_{x \in X} \mathrm{dist}(x, C)$
- k-median: $\mathrm{Cost}(X, C) = \sum_{x \in X} \mathrm{dist}(x, C)$
- k-means: $\mathrm{Cost}(X, C) = \sum_{x \in X} \left(\mathrm{dist}(x, C)\right)^2$
- (k, z)-clustering: $\mathrm{Cost}(X, C) = \sum_{x \in X} \left(\mathrm{dist}(x, C)\right)^z$



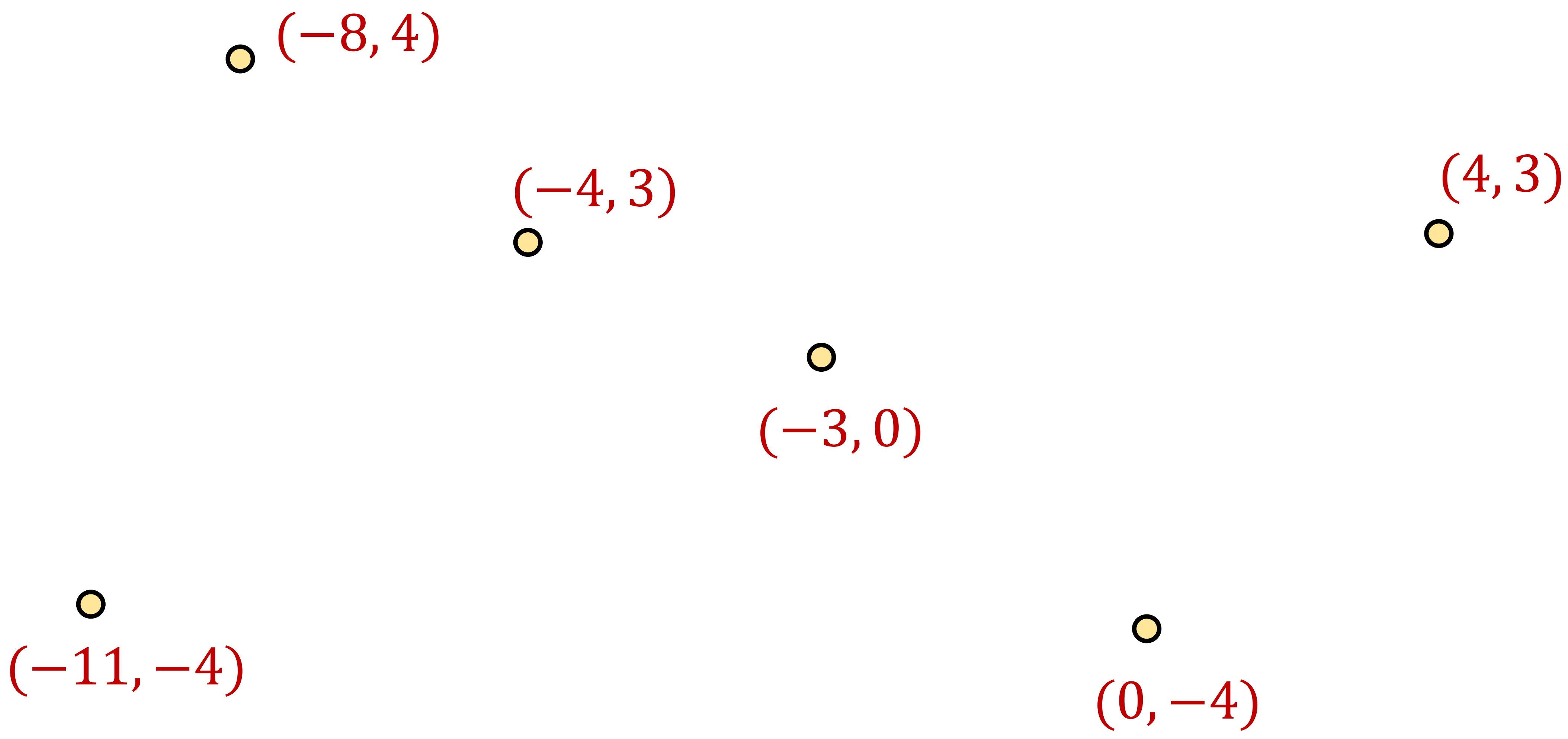
Euclidean k-Clustering

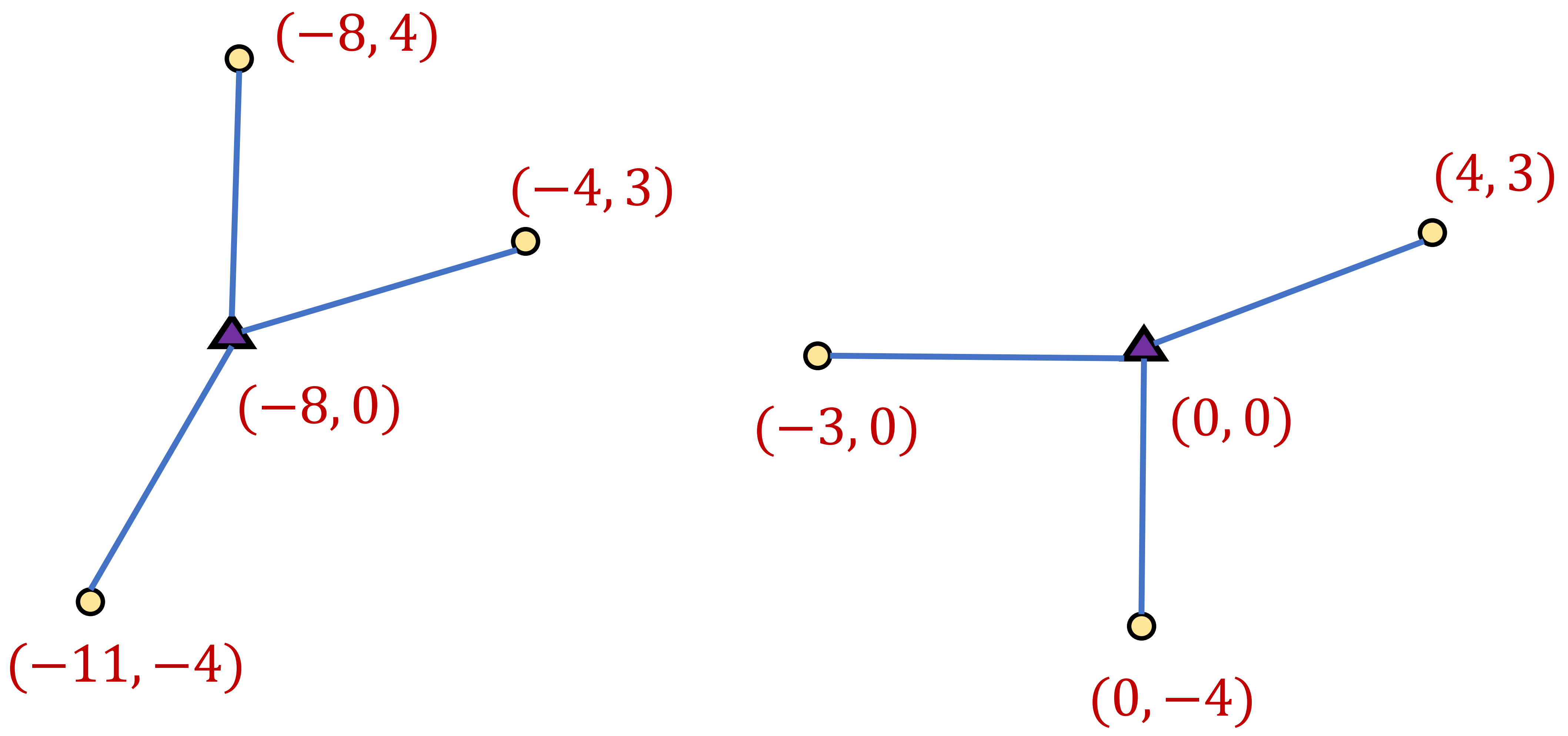
- For Euclidean k-clustering, input points $X = x_1, \dots, x_n$ are in $\mathbb{R}^d$ (for us, they will be in $[\Delta]^d := \{1,2, \dots, \Delta\}^d$)

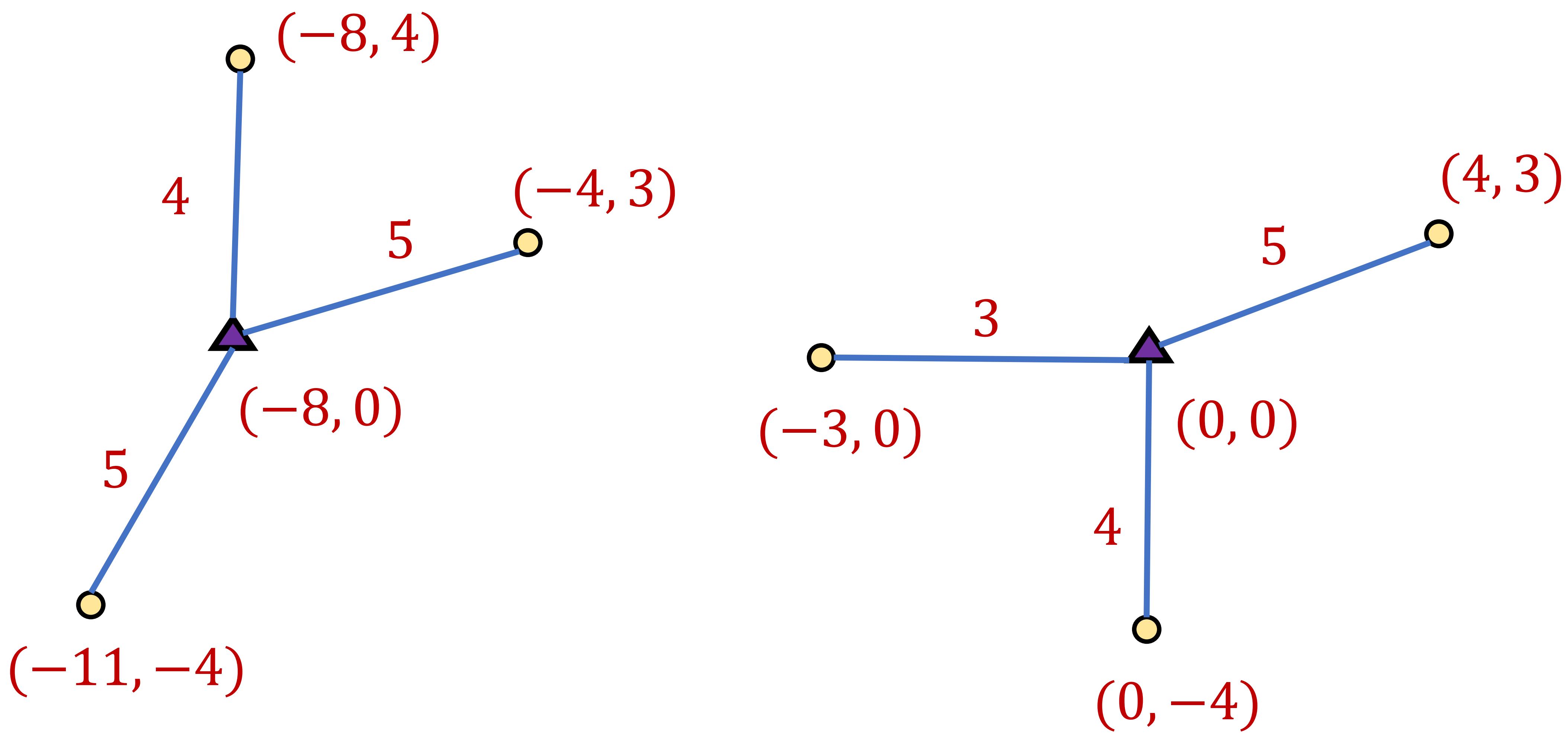
- $\mathrm{dist}(x,y) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_d - y_d)^2}$ is the Euclidean distance

- (k,z)-clustering problem:

$$\min_{C:|C|\le k} \mathrm{Cost}(X,C) = \min_{C:|C|\le k} \Sigma_{x\in X} \left(\mathrm{dist}(x,C)\right)^z$$

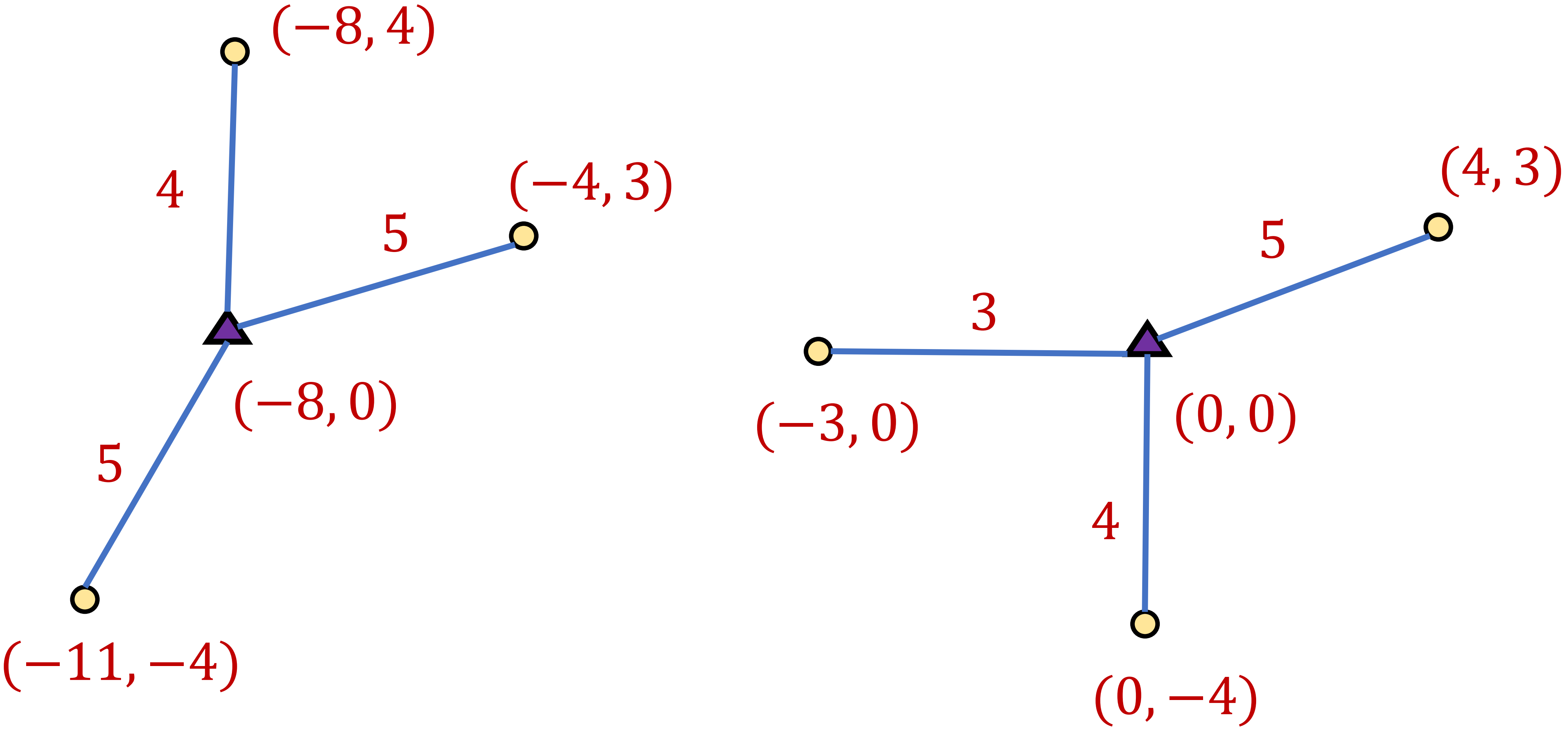

$(-8, 4)$
$(-4, 3)$
$(4, 3)$
$(-3, 0)$
$(-11, -4)$
$(0, -4)$


$(-8, 4)$
$(-4, 3)$
$(-8, 0)$
$(-11, -4)$
$(4, 3)$
$(-3, 0)$
$(0, 0)$
$(0, -4)$

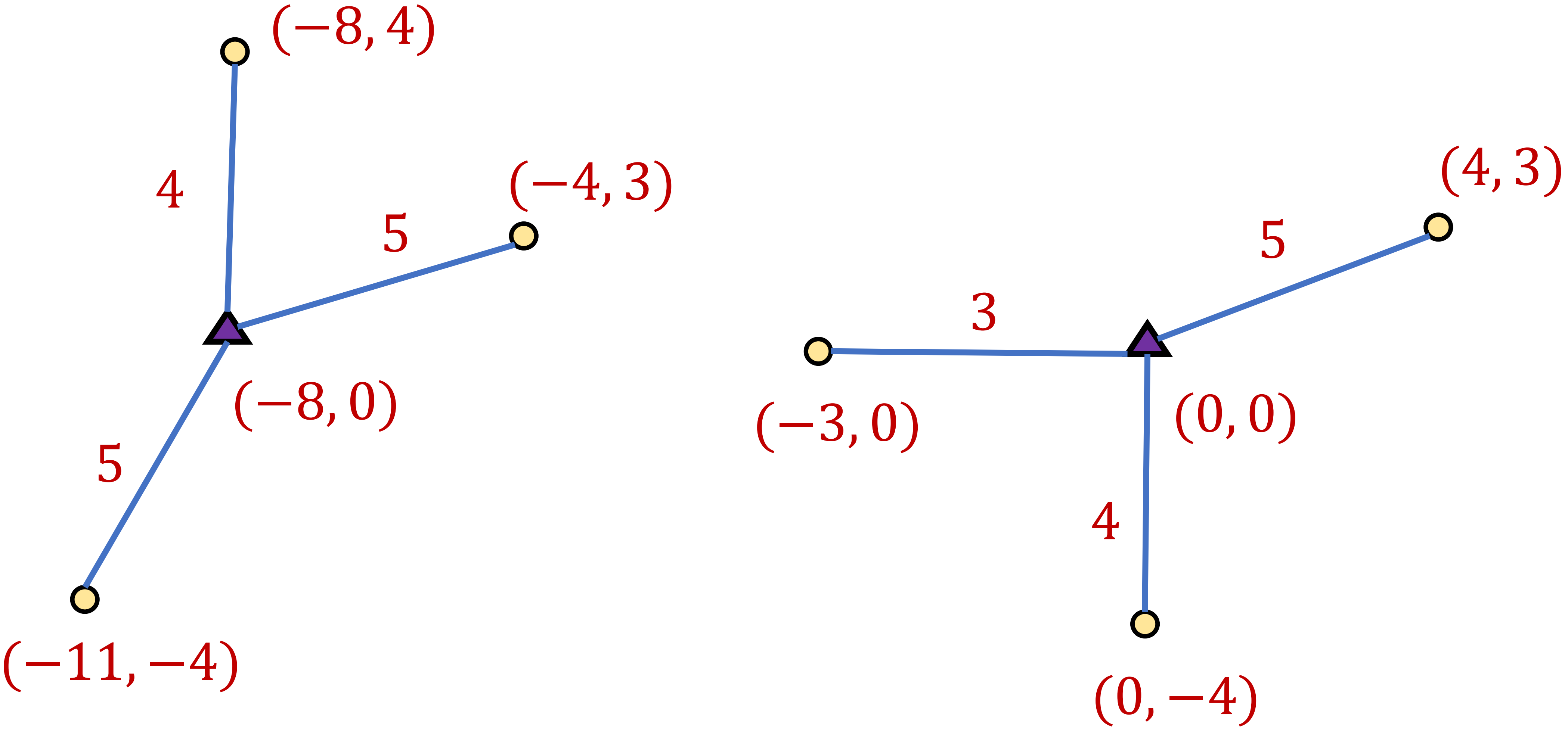


$(-8,4)$
4
$(-4,3)$
5
$(-8,0)$
5
$(-11,-4)$
$(4,3)$
5
3
$(-3,0)$
$(0,0)$
4
$(0,-4)$

k-center: $\mathrm{Cost}(X, C) = \max_{x \in X} \mathrm{dist}(x, C) = 5$

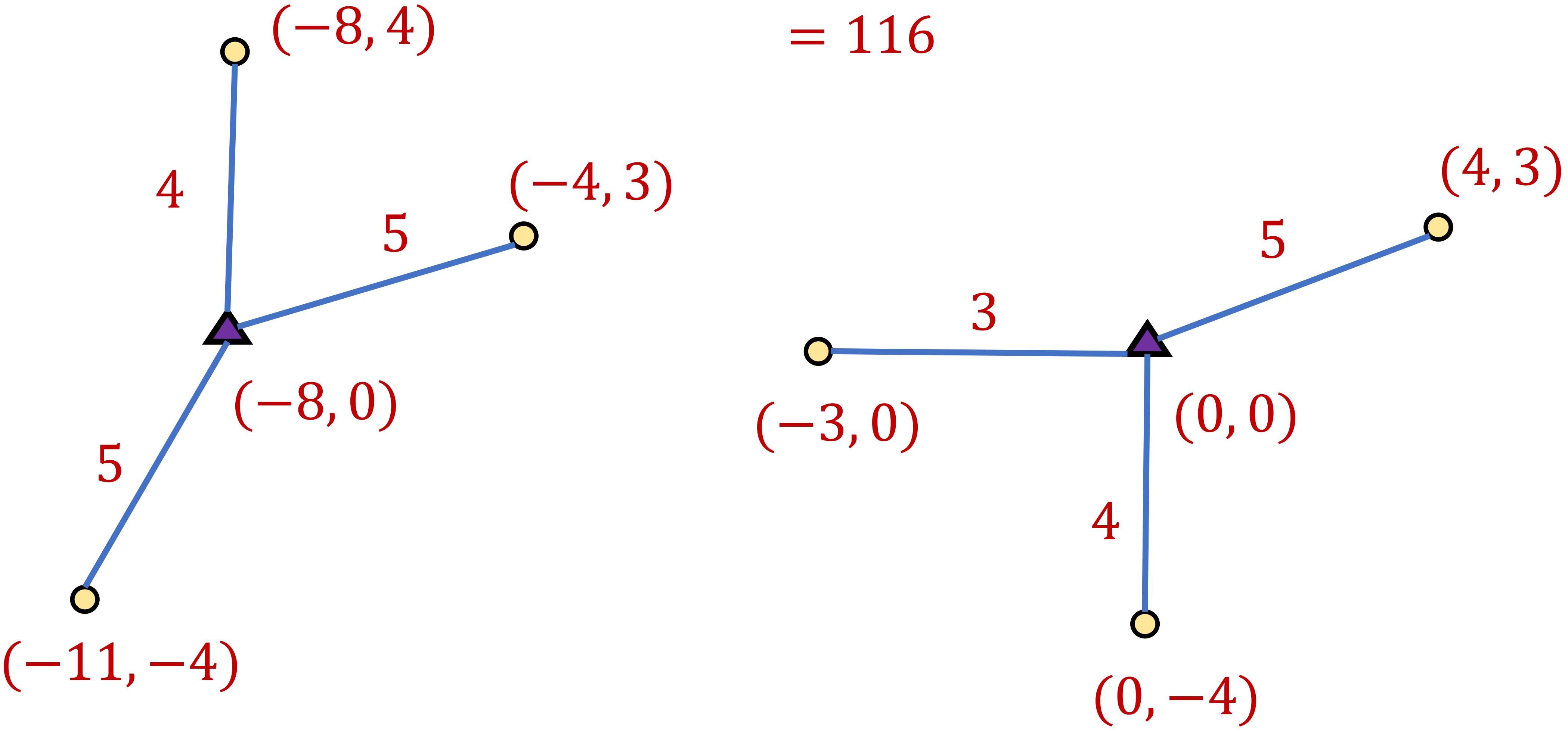


k-median: $\mathrm{Cost}(X, C) = \sum_{x \in X} \mathrm{dist}(x, C) = 4 + 5 + 5 + 3 + 4 + 5 = 26$



k-means: $\mathrm{Cost}(X, C) = \sum_{x \in X} \left(\mathrm{dist}(x, C)\right)^2 = 16 + 25 + 25 + 9 + 16 + 25$

$= 116$



Coreset

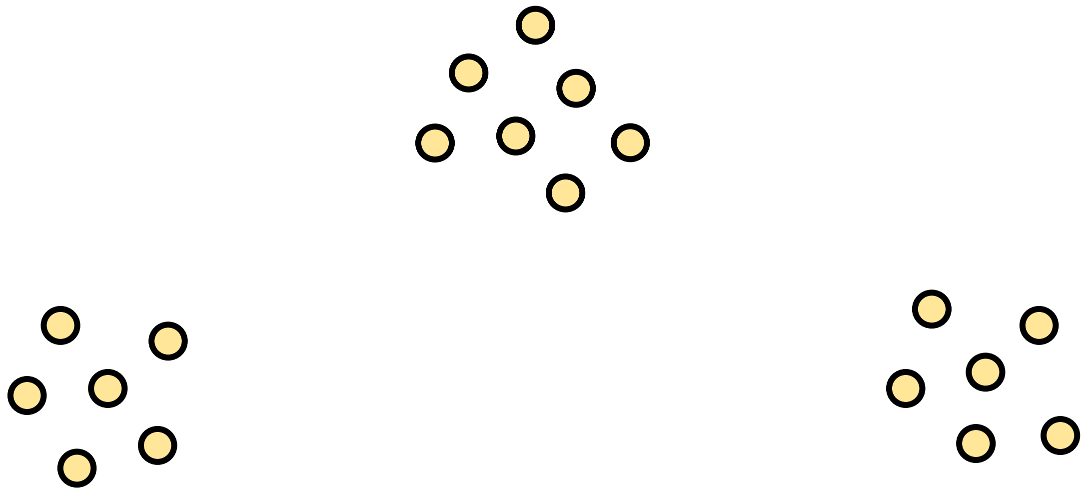
- Subset X' of representative points of X for a specific clustering objective

- $\mathrm{Cost}(X, C) \approx \mathrm{Cost}(X', C)$
  for all sets C with $|C| = k$

Coreset

- Subset X' of representative points of X for a specific clustering objective

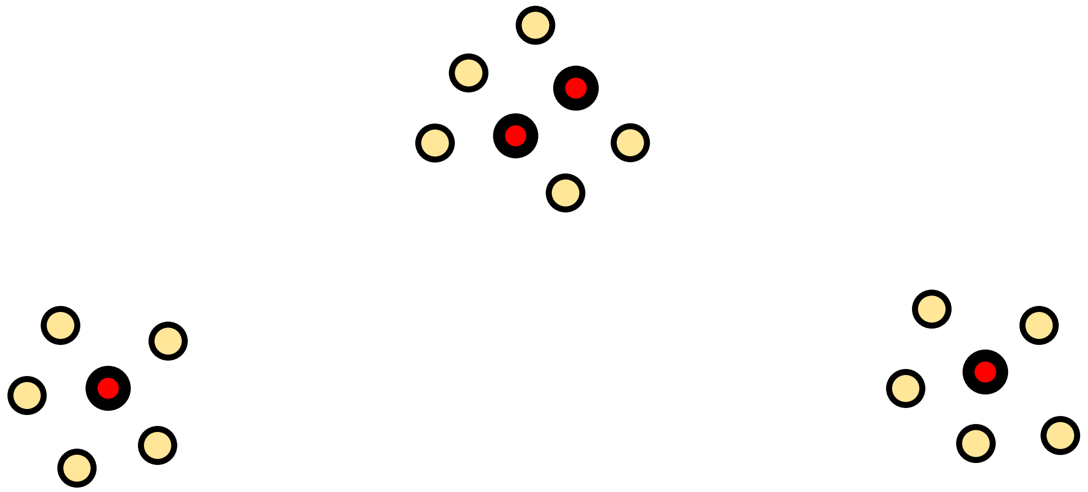
- $\text{Cost}(X, C) \approx \text{Cost}(X', C)$ for all sets C with $|C| = k$



Coreset

- Subset X' of representative points of X for a specific clustering objective

- $\text{Cost}(X, C) \approx \text{Cost}(X', C)$ for all sets C with $|C| = k$



Coreset (Formal Definition)

- Given a set X and an accuracy parameter $\varepsilon > 0$, we say a set X' with weight function w is an $(1+\varepsilon)$-*multiplicative coreset* for a cost function Cost, if for all queries C with $|C| \leq k$, we have

$$(1-\varepsilon)\mathrm{Cost}(X,C) \leq \mathrm{Cost}(X',C,w) \leq (1+\varepsilon)\mathrm{Cost}(X,C)$$

(k,z)-clustering: $\mathrm{Cost}(X',C,w) = \sum_{x\in X'} w(x)\cdot \left(\mathrm{dist}(x,C)\right)^z$

(k, z)-Clustering in the Streaming Model

- Merge-and-reduce framework

- Suppose there exists a $(1+\varepsilon)$-coreset construction for (k, z)-clustering that uses $f\left(k, \frac{1}{\varepsilon}\right)$ weighted input points

$\boxed{\tilde{O}\left(\frac{k^2}{\varepsilon^2}\right)}$

- Partition the stream into blocks containing $f\left(k, \frac{\log n}{\varepsilon}\right)$ points

(k, z)-Clustering in the Streaming Model

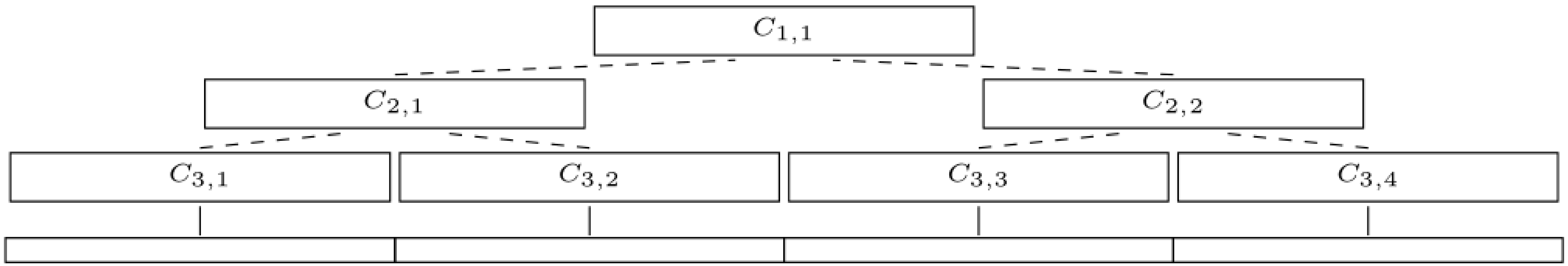
- Partition the stream into blocks containing $f\left(k, \frac{\log n}{\varepsilon}\right)$ points
- Create a $\left(1 + \frac{\varepsilon}{\log n}\right)$-coreset for each block
- Create a $\left(1 + \frac{\varepsilon}{\log n}\right)$-coreset for the set of points formed by the union of two coresets for each block

Reduce

Merge

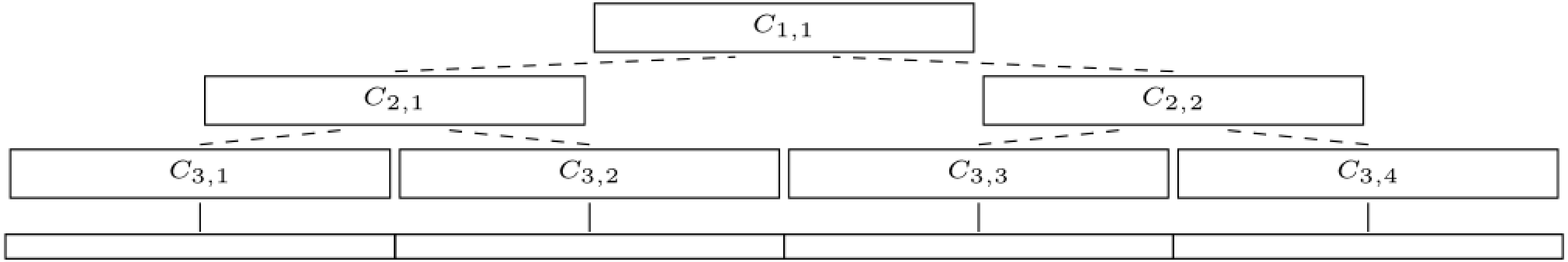
(k, z)-Clustering in the Streaming Model

- Partition the stream into blocks containing $f\left(k, \frac{\log n}{\varepsilon}\right)$ points
- Create a $\left(1 + \frac{\varepsilon}{\log n}\right)$-coreset for each block
- Create a $\left(1 + \frac{\varepsilon}{\log n}\right)$-coreset for the set of points formed by the union of two coresets for each block



(k, z)-Clustering in the Streaming Model

- There are $O(\log n)$ levels
- Each coreset is a $\left(1 + \frac{\varepsilon}{\log n}\right)$-coreset of two coresets
- Total approximation is $\left(1 + \frac{\varepsilon}{\log n}\right)^{\log n} = (1 + O(\varepsilon))$



(k,z)-Clustering in the Streaming Model

- Suppose there exists a $(1+\varepsilon)$-coreset construction for (k,z)-clustering that uses $f\left(k,\frac{1}{\varepsilon}\right)$ weighted input points
- Partition the stream into blocks containing $f\left(k,\frac{\log n}{\varepsilon}\right)$ points
- Total space is $f\left(k,\frac{\log n}{\varepsilon}\right)\cdot O(\log n)$ points

For k-means clustering, this is $\tilde{O}\left(\frac{k^2}{\varepsilon^2}\cdot\log^3 n\right)$ points