UCSC Silicon Valley Extension

Advanced C Programming

Minimum Spanning Tree

Radhika Grover

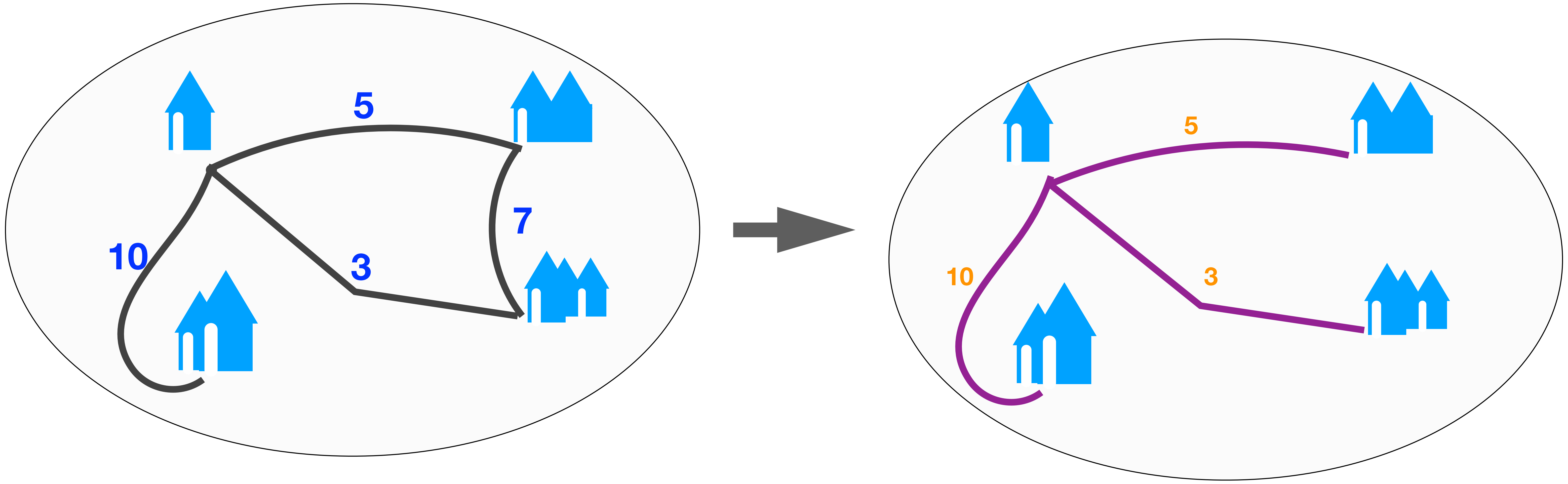
Minimum Spanning Tree -applications

- Minimum cost routing spanning trees
- Electronic circuits - use least amount of wire to connect pins
- Image segmentation

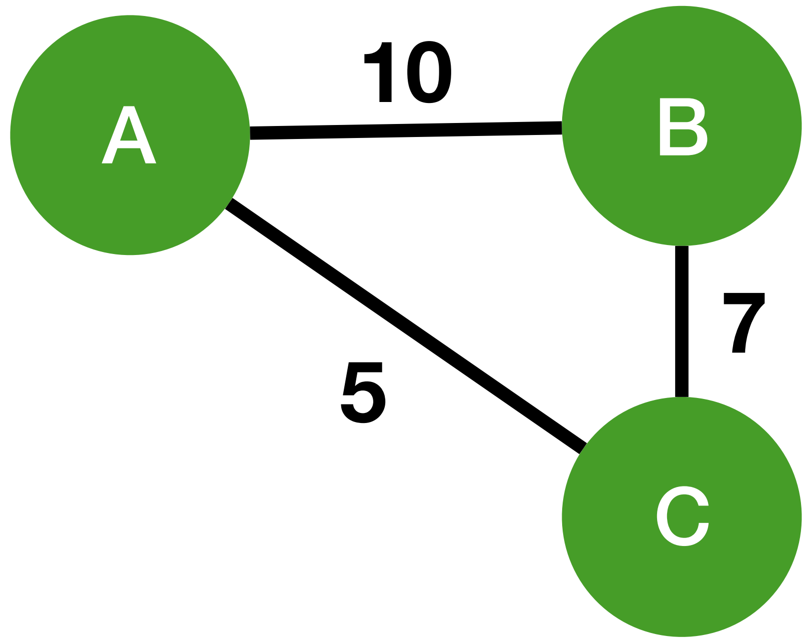
Minimum Spanning Tree -applications

- Image segmentation

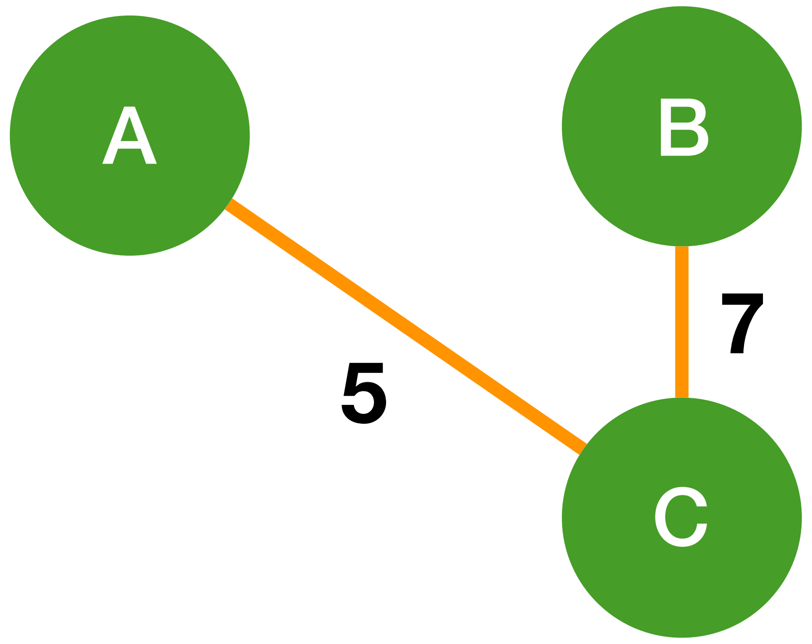
- Network design: Connecting all homes via cable



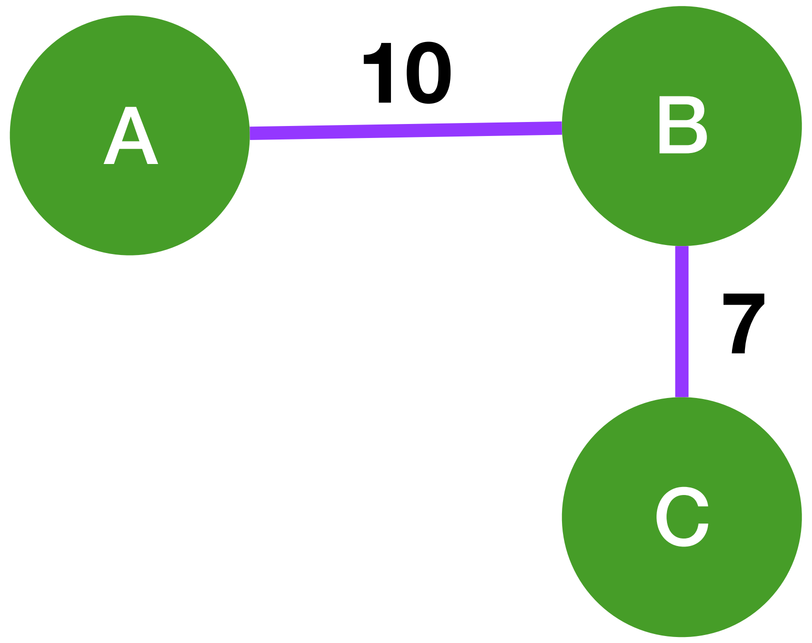
MST vs Shortest path



Graph



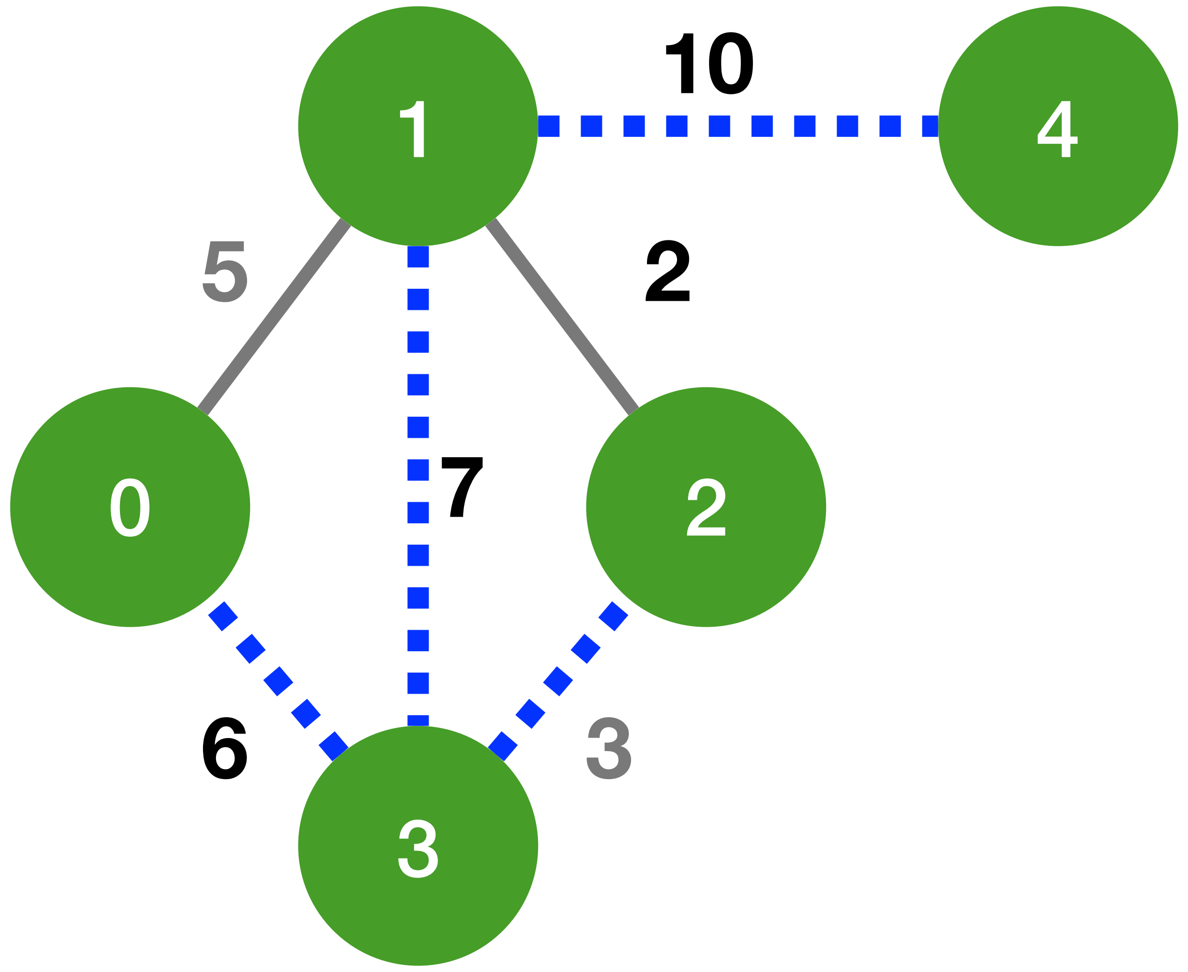
Minimum Spanning Tree



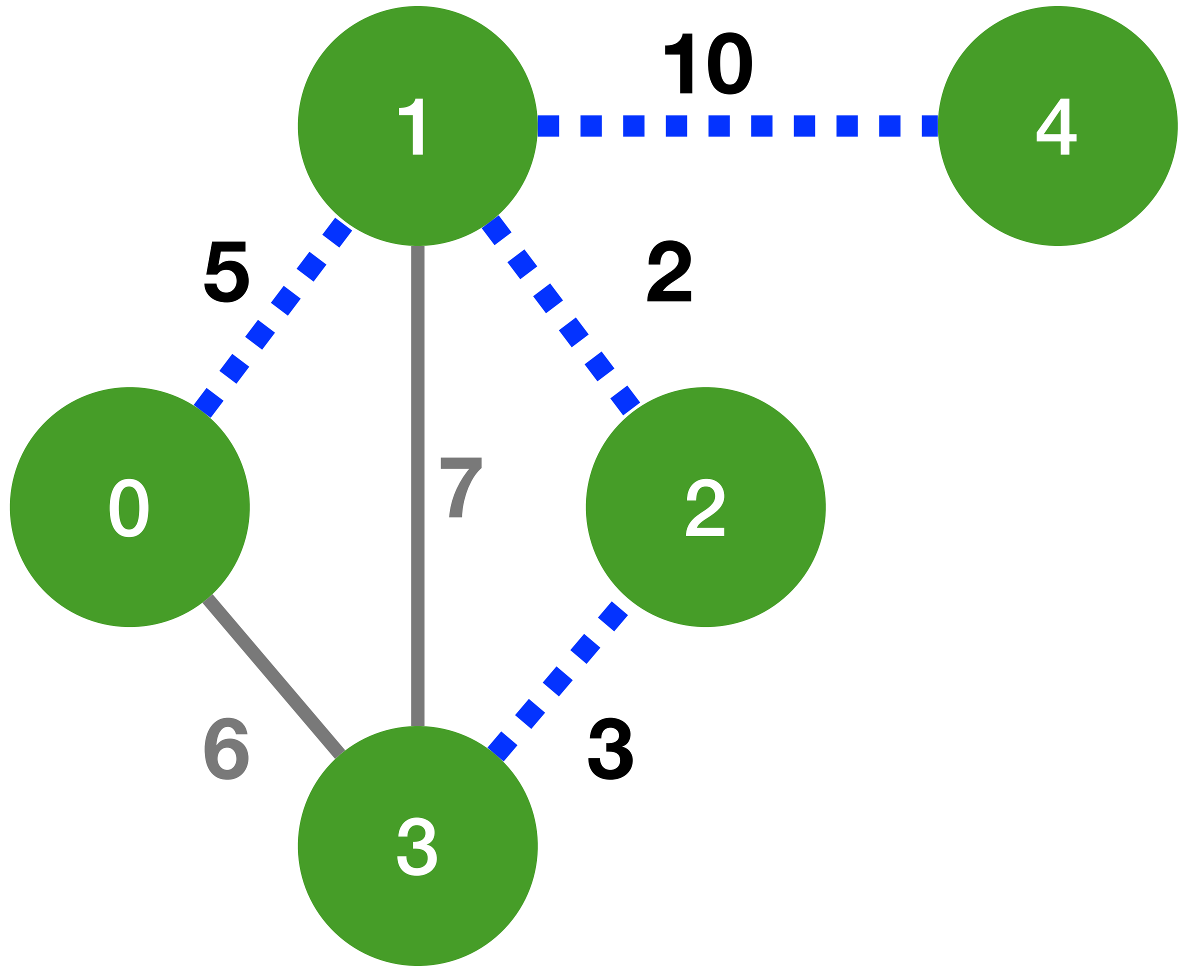
Shortest path from B

Minimum Spanning Tree

Spanning Tree: Tree that contains all vertices of graph G



Cost = 26

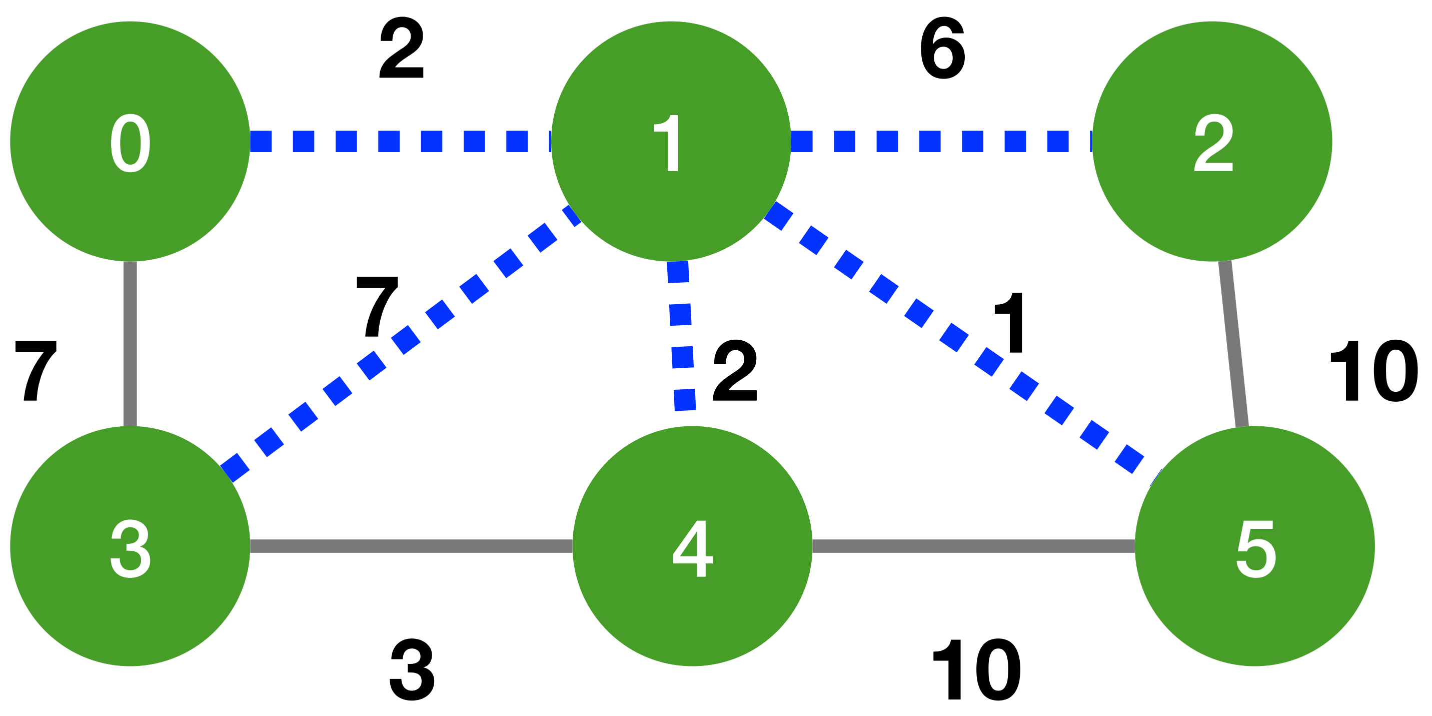


Cost = 20

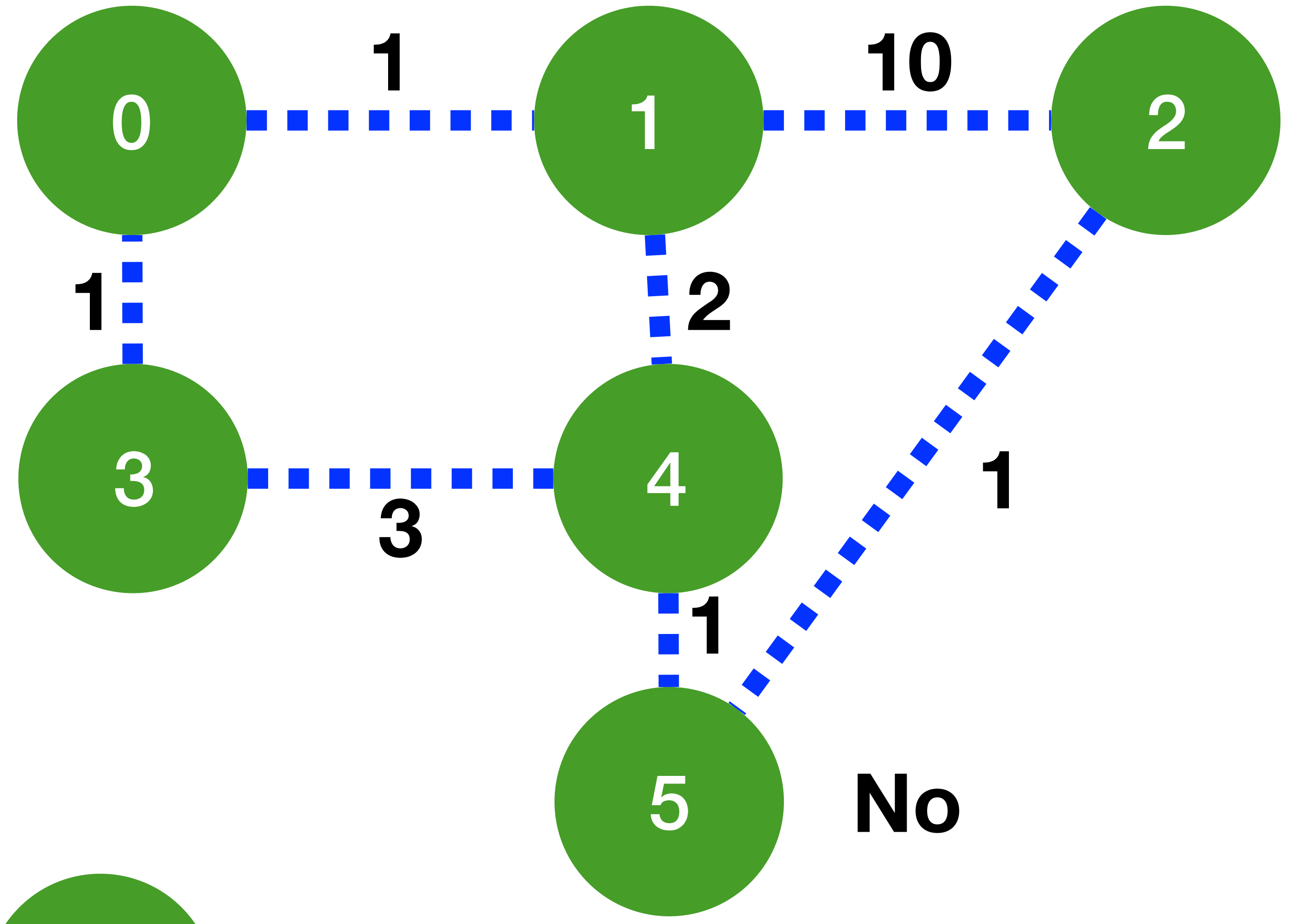
Does any other spanning tree have lower cost ?

Example

Which of the following are minimum spanning trees?

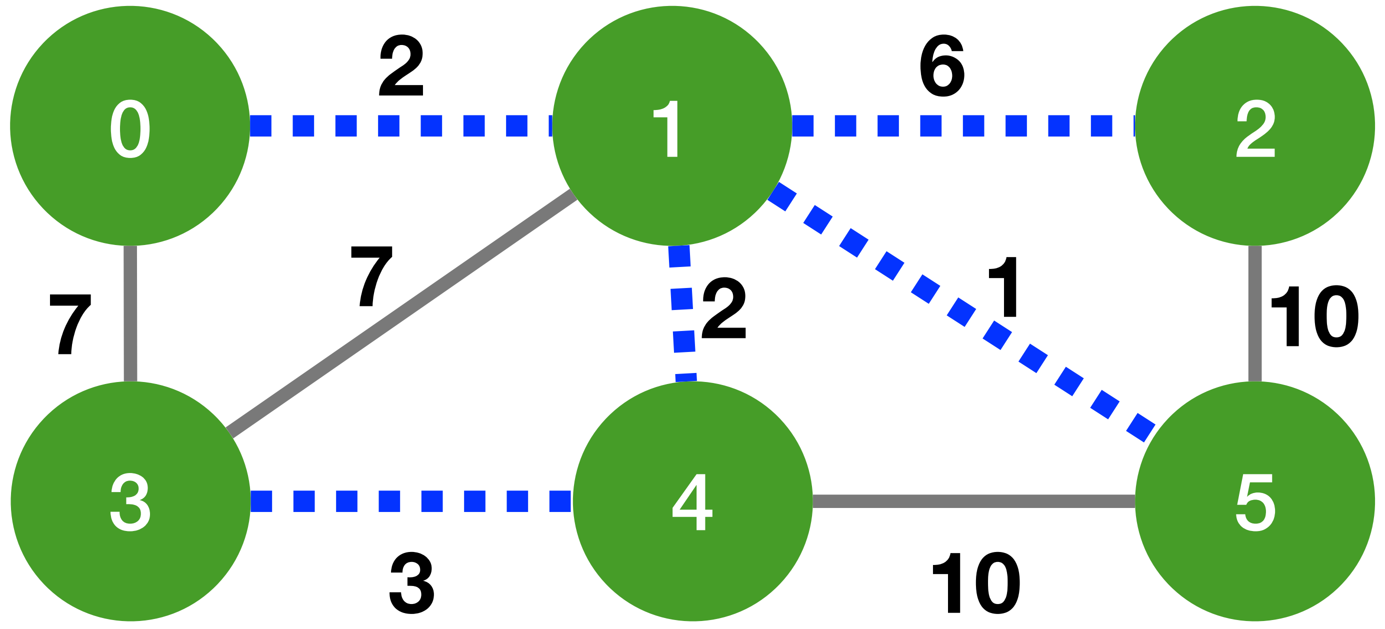


No



No

Yes



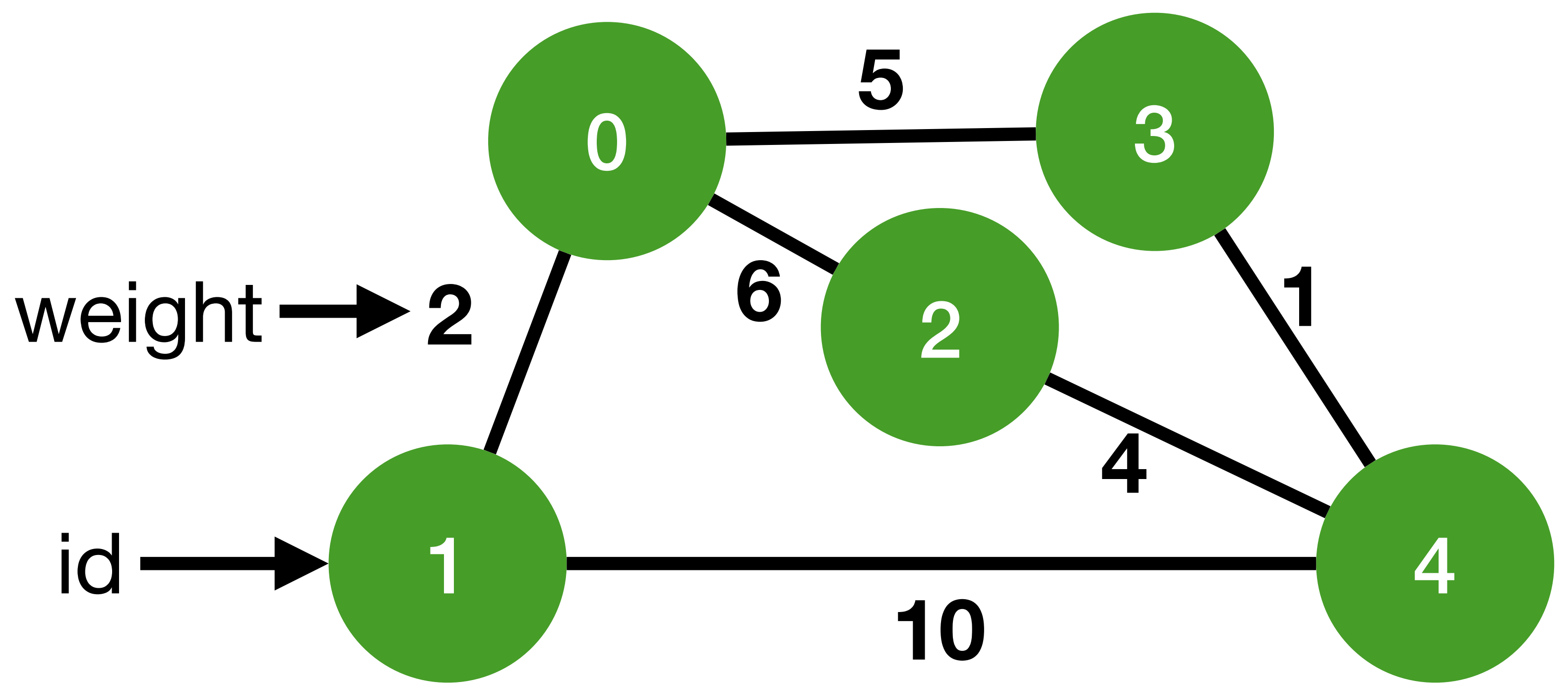
Weighted Graph

- Weight of an edge represented

    1. Cost: Amount of effort to travel from one place to another

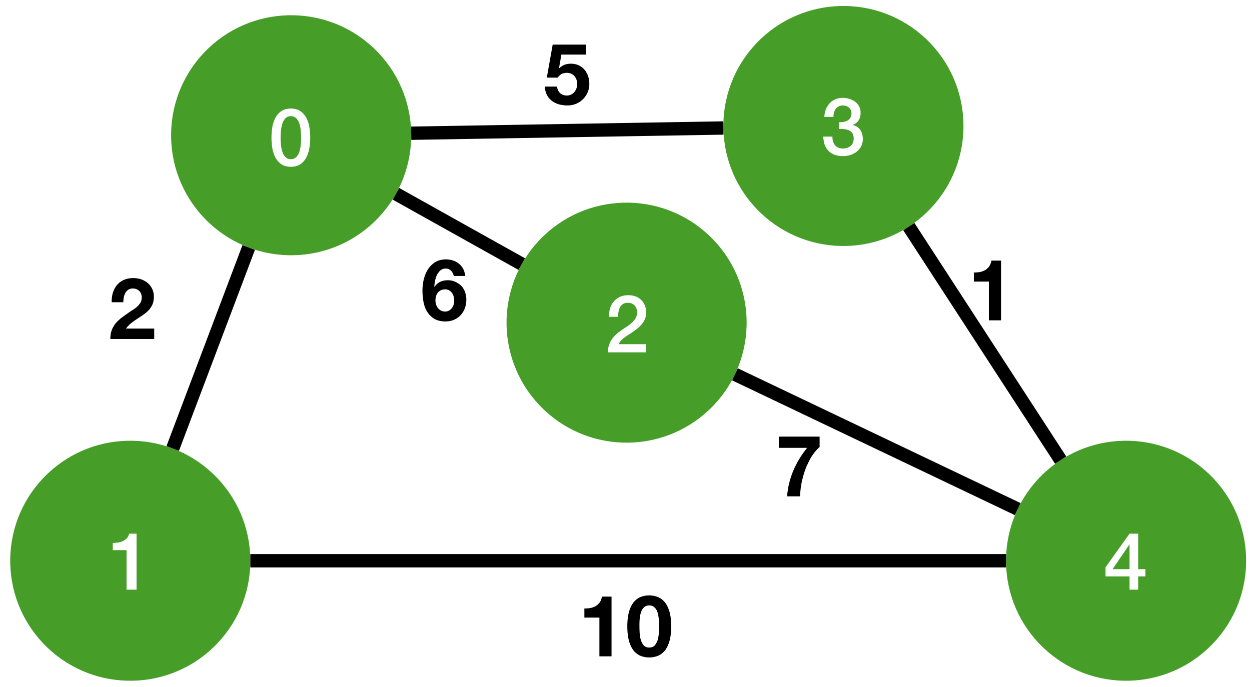
    2. Capacity: Maximum amount of flow that can be transported from one place to another

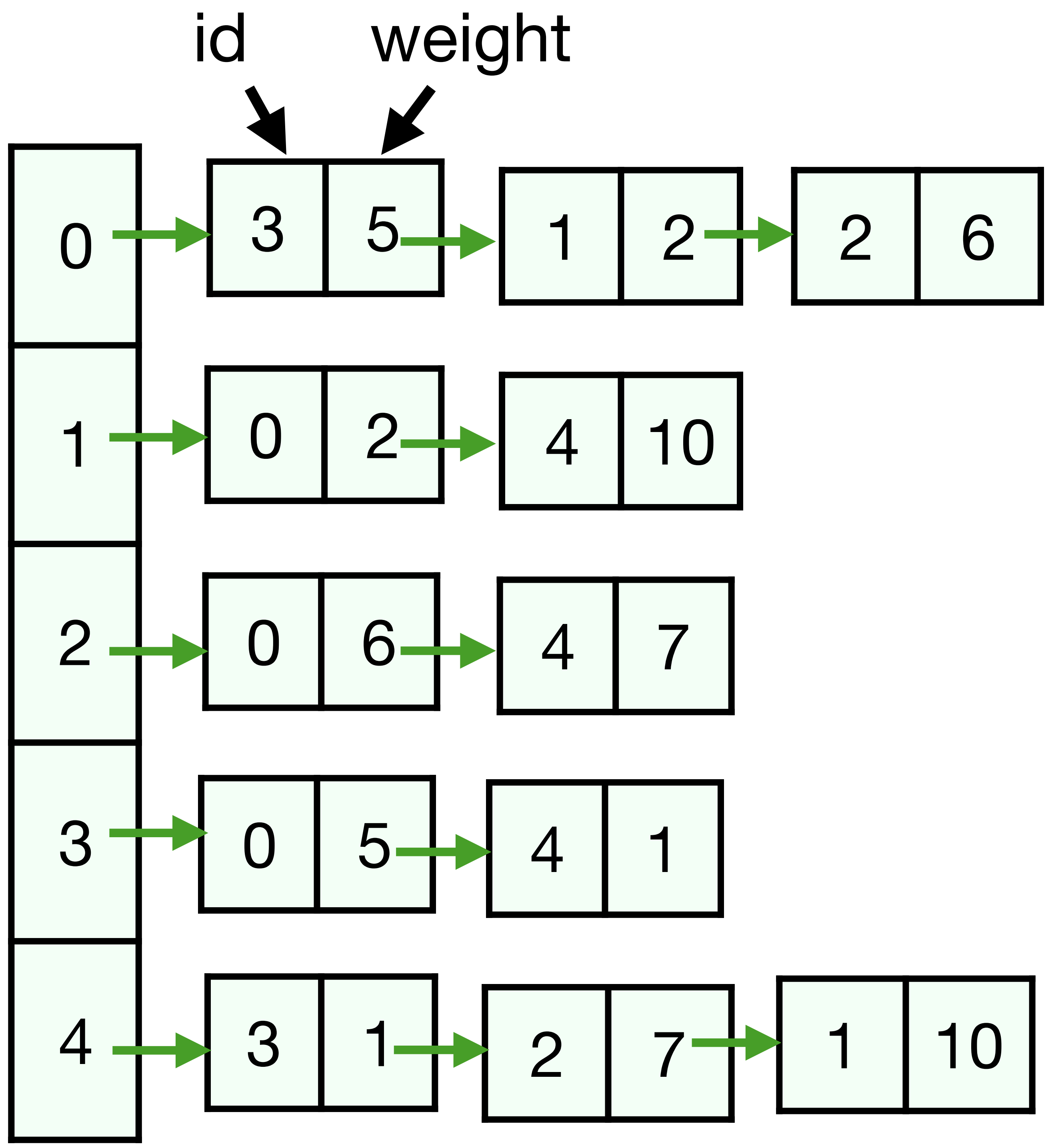
- Representation using adjacency list and array



Representation Weighted Graph

Adjacency list: Each node contains an additional field cost



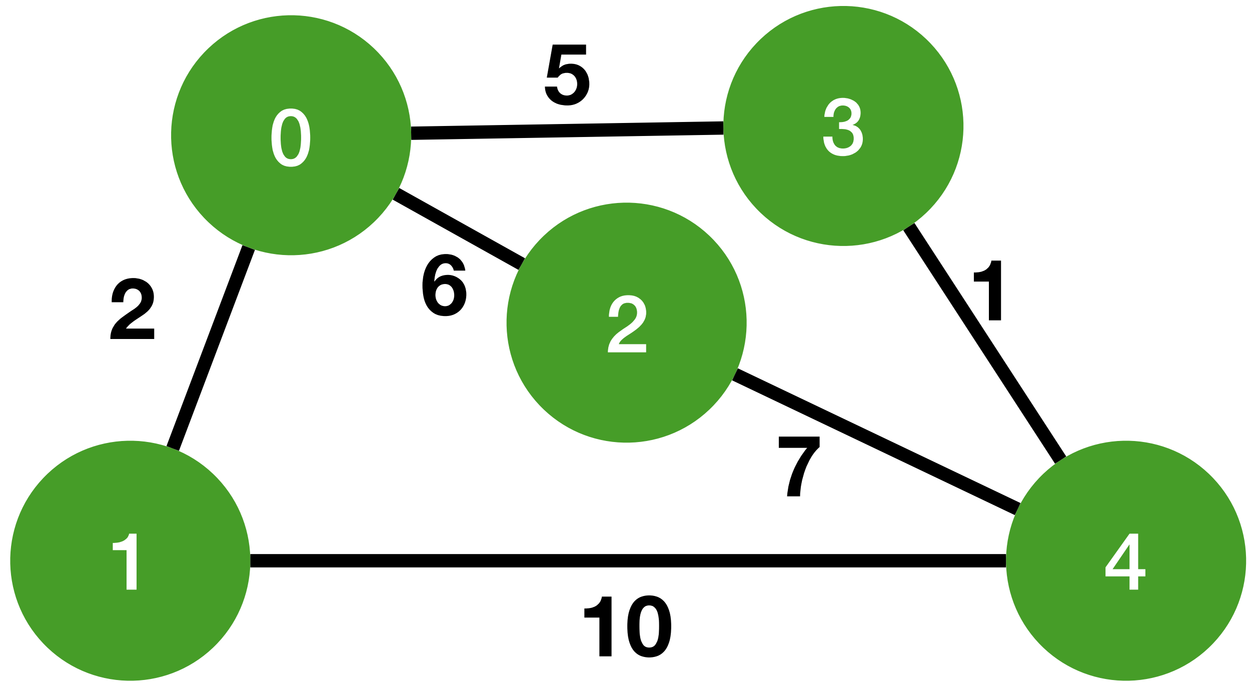


Adjacency List

Representation Weighted Graph

Adjacency array: Use a 2 - dimensional array a [i][j] to store weight

a [i] [j] = very large value if there is no edge between i and j



	0	1	2	3	4
0	X	2	6	5	X
1	2	X	X	X	10
2	6	X	X	X	7
3	5	X	X	X	1
4	X	10	7	1	X

X = very large value

Minimum Spanning Tree Exercise

- Given an edge weighted graph, find a subtree spanning all vertices whose weight is minimum.

- Greedy algorithms with optimal solution

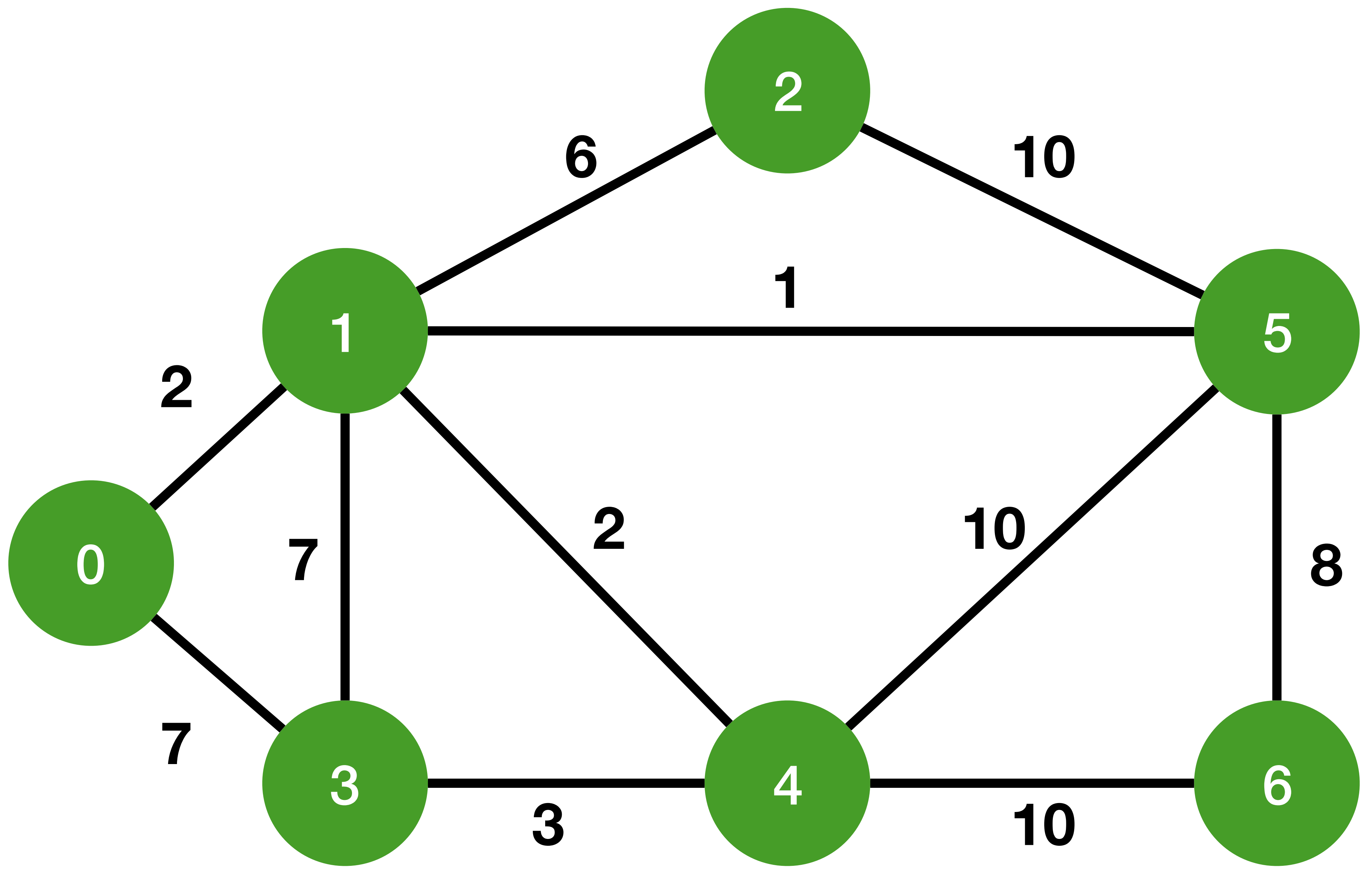
  1. Prim's algorithm

  2. Kruskal's algorithm

Prim’s algorithm (informal)

1. Pick an arbitrary starting vertex s and mark it as reached, set key = 0
2. Add all other vertices of G to a min - priority queue with key = ∞
3. Remove the vertex u with minimum key from queue and add to MST
4. Update keys and predecessor value of vertices (v) adjacent to u if weight (u, v) < key (v)
5. Repeat steps 3 and 4 until queue is empty
6. Return MST

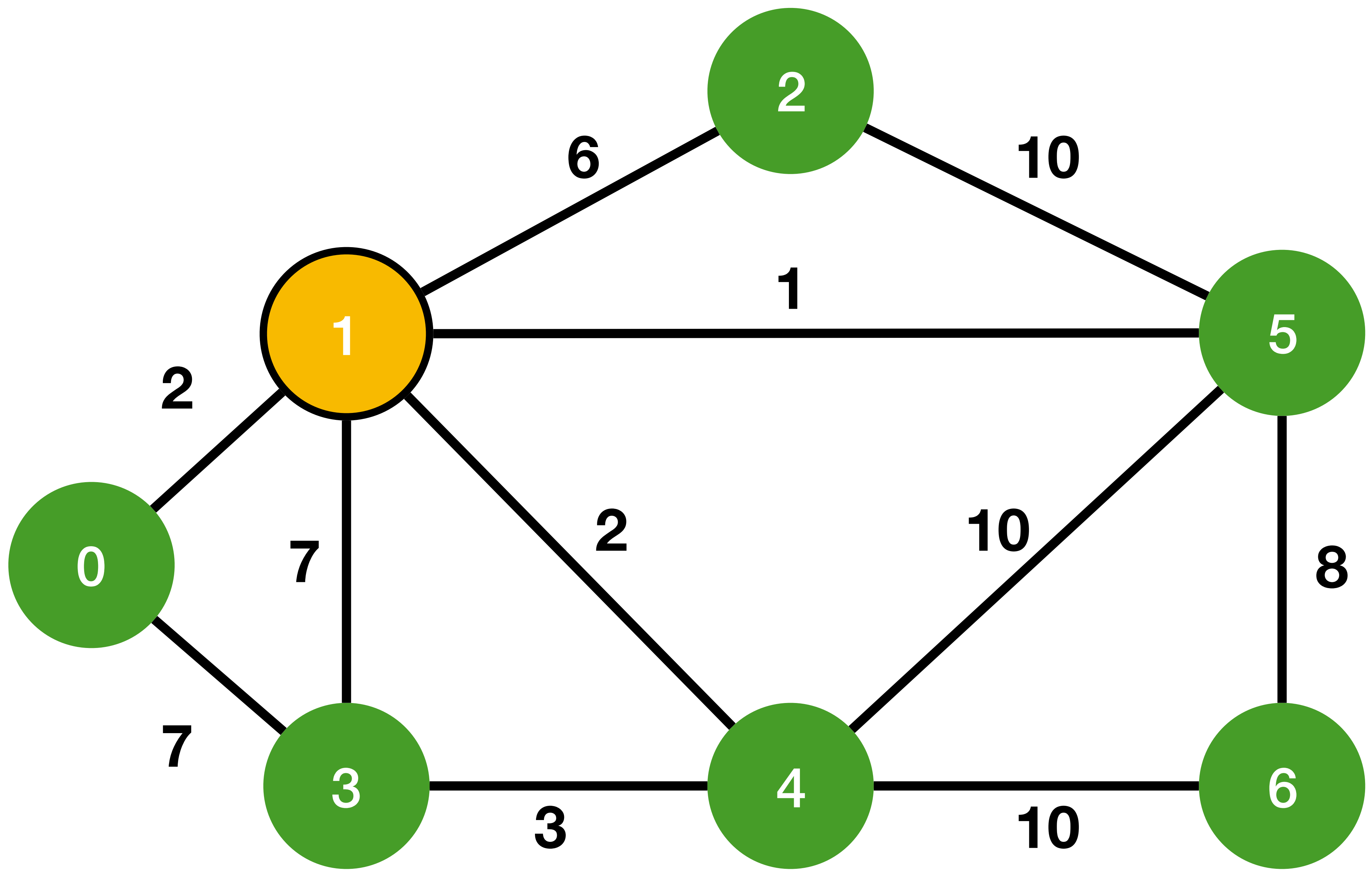
Example (Prim's)



Minimum Priority Queue

id	1	0	2	3	4	5	6
key							
P							

Example (Prim's)

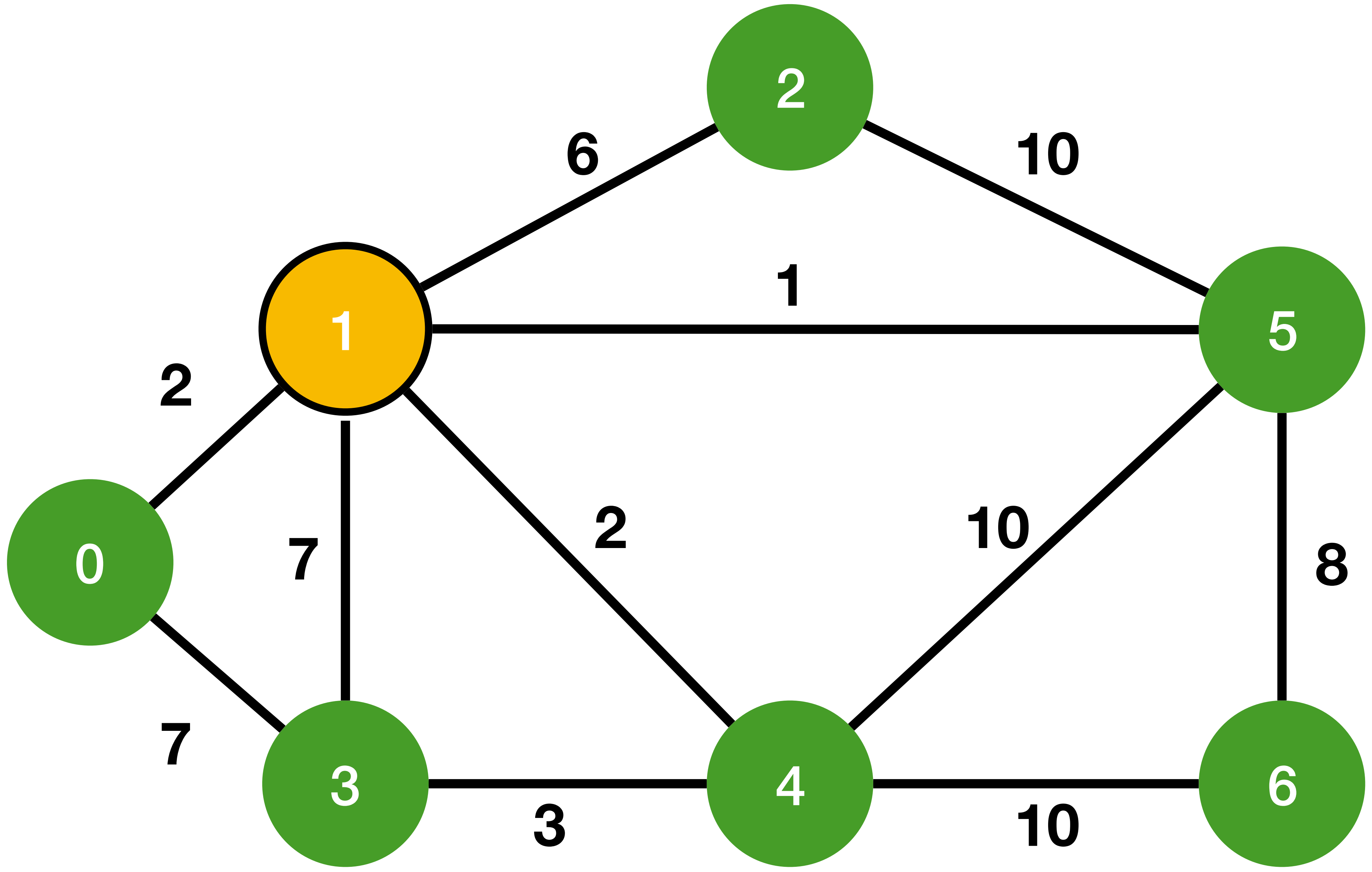


Minimum Priority Queue

id	1	0	2	3	4	5	6
key	0	∞	∞	∞	∞	∞	∞
P	-1	-1	-1	-1	-1	-1	-1

Step 1: Pick 1 as starting vertex, key = 0
MST = { 1}
P = {-1}

Example (Prim's)

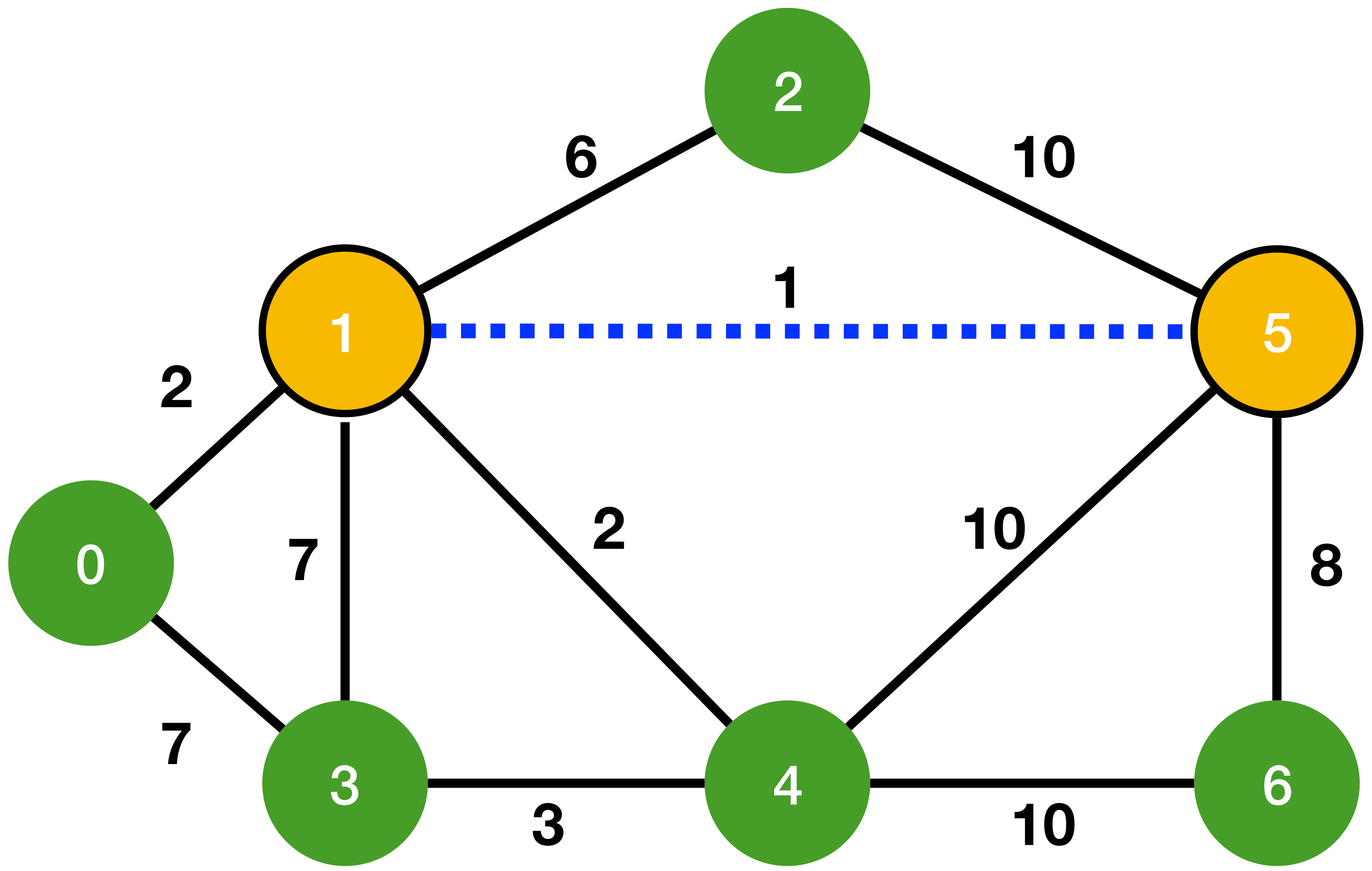


Minimum Priority Queue

id	0	2	3	4	5	6
key	2	6	7	2	1	∞
P	1	1	1	1	1	-1

Step 2: Update P-1 keys P of vertices adjacent to starting vertex

Example (Prim's)



Minimum Priority Queue

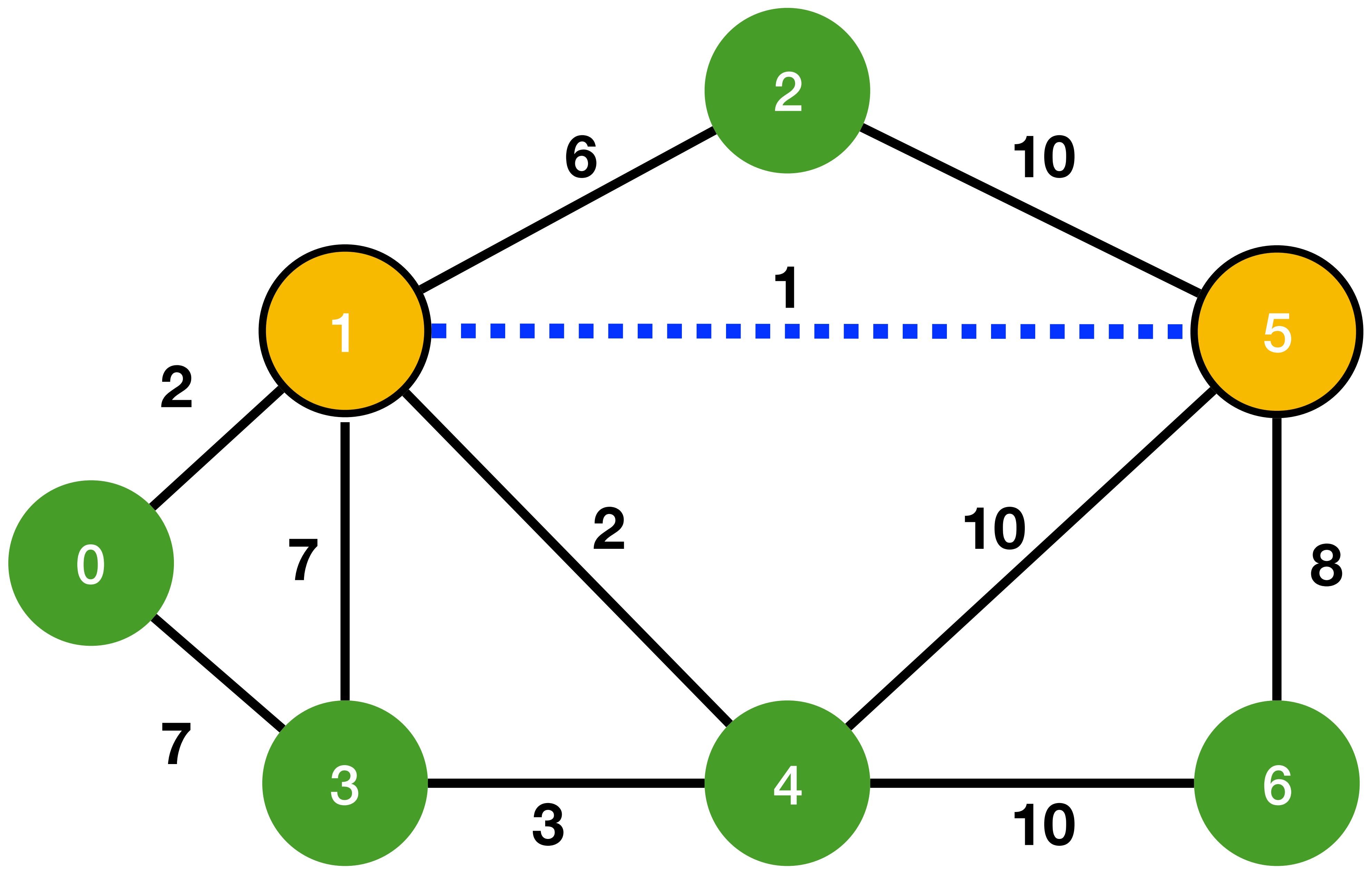
id	0	2	3	4	5	6
key	2	6	7	2	1	∞
P	1	1	1	1	1	-1

Step 3: Move vertex with minimum key into MST

MST = { 1, 5}

P = {-1, 1}

Example (Prim's)

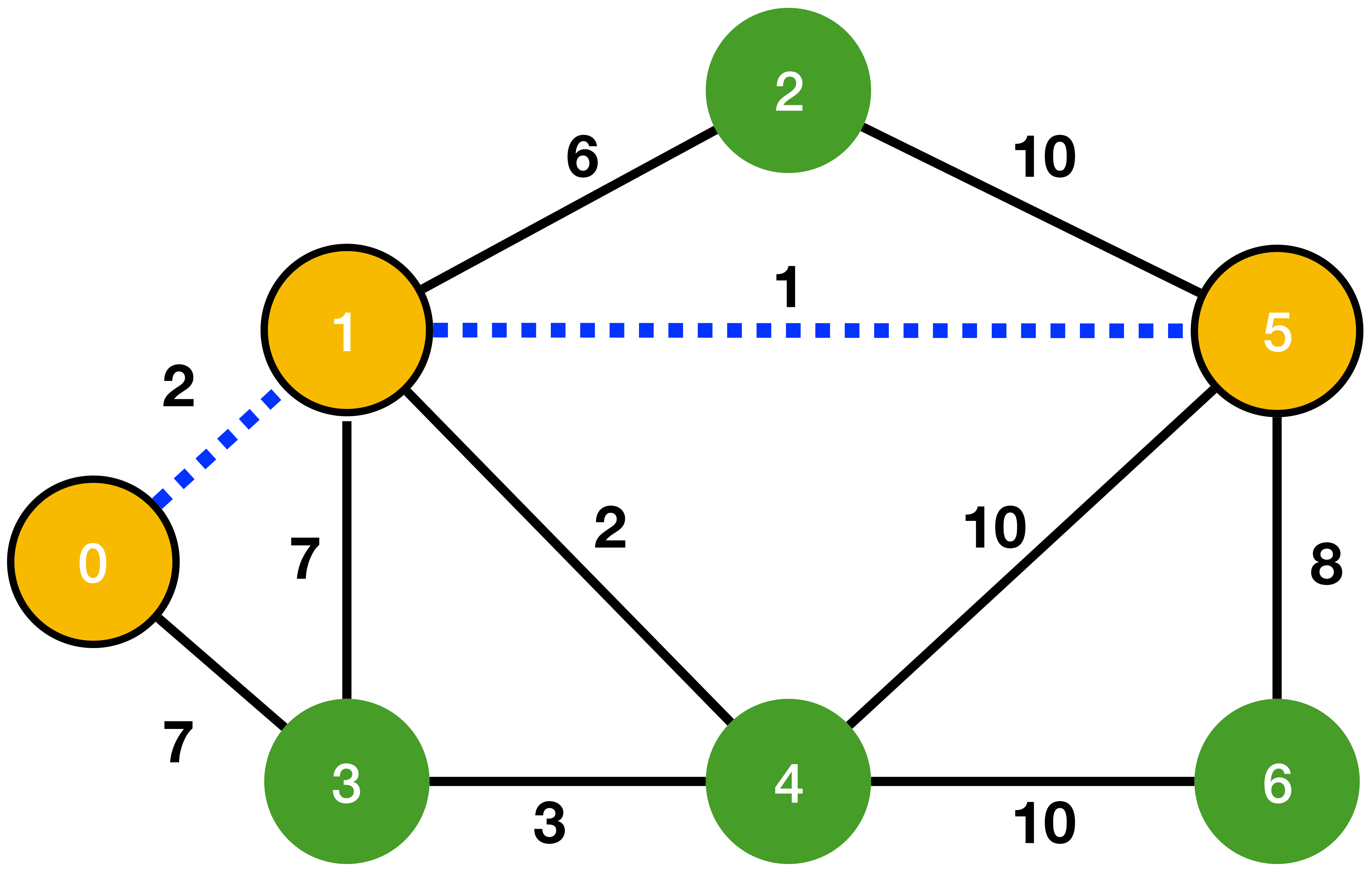


Minimum Priority Queue

id	0	2	3	4	**6**
key	2	6	7	2	**8**
P	1	1	1	1	**5**

Step 4: Update keys and P of vertices(v) adjacent to 5(u) if weight (5, v) < key (v)

Example (Prim's)

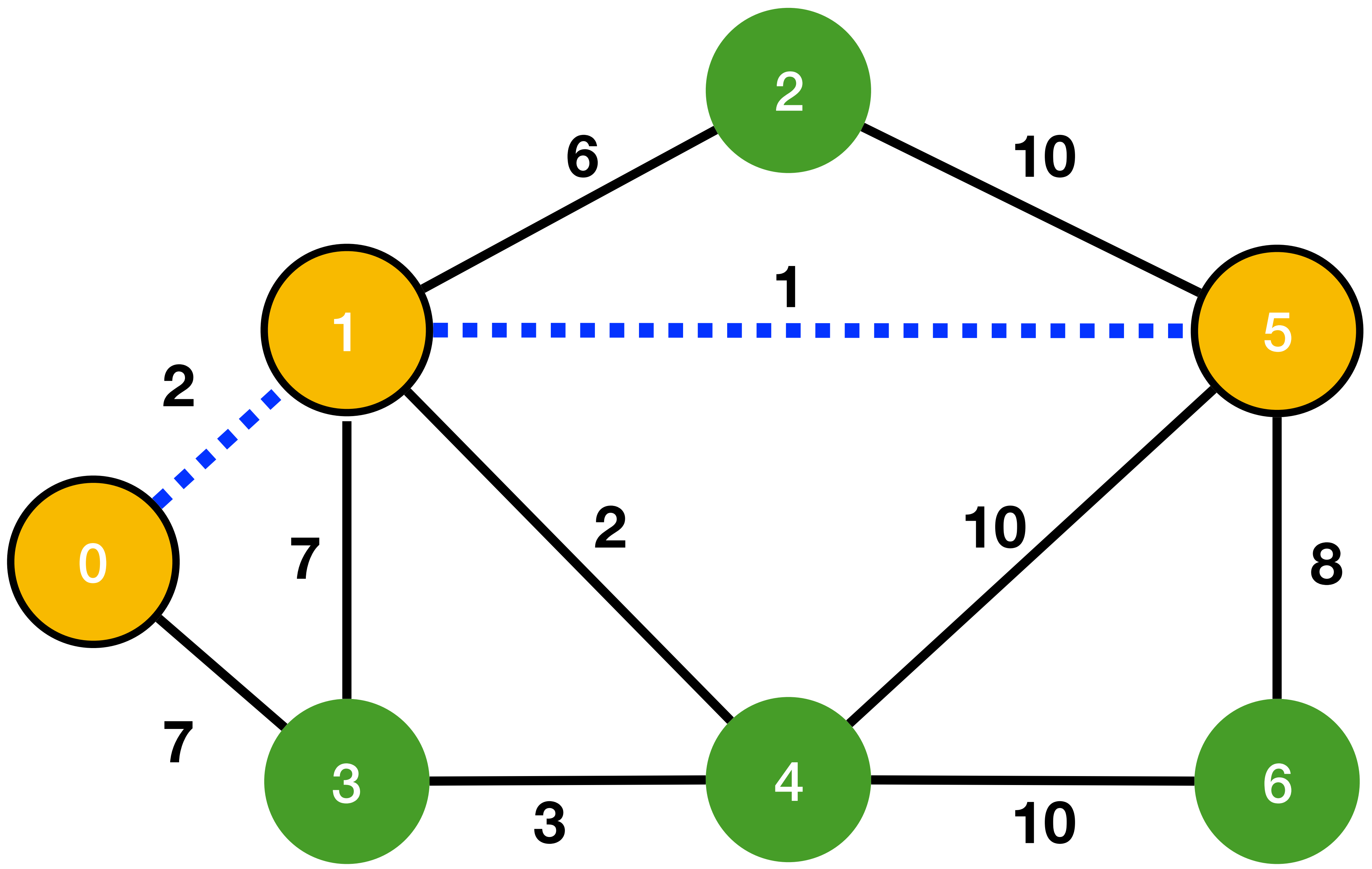


Minimum Priority Queue

id	2	3	4	6
key	6	7	2	8
P	1	1	1	5

Step 5: Move vertex with minimum key into MST
MST = {1, 5, 0}
P = {-1, 1, 1}

Example (Prim's)

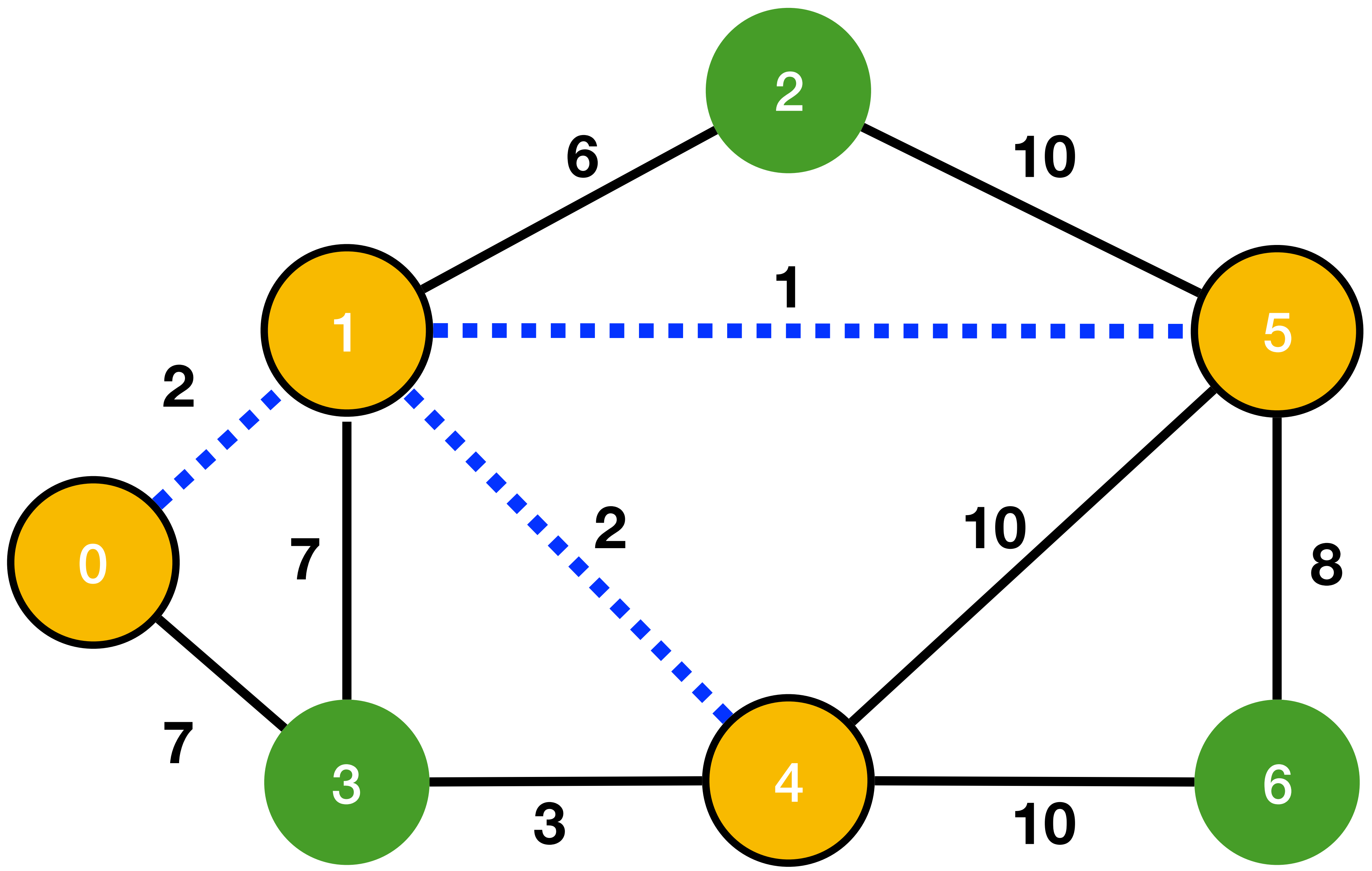


Minimum Priority Queue

id	2	3	4	6
key	6	7	2	8
P	1	1	1	5

Step 6: Update keys and P of vertices(v) adjacent to 0(u) if weight (0, v) < key (v)

Example (Prim's)

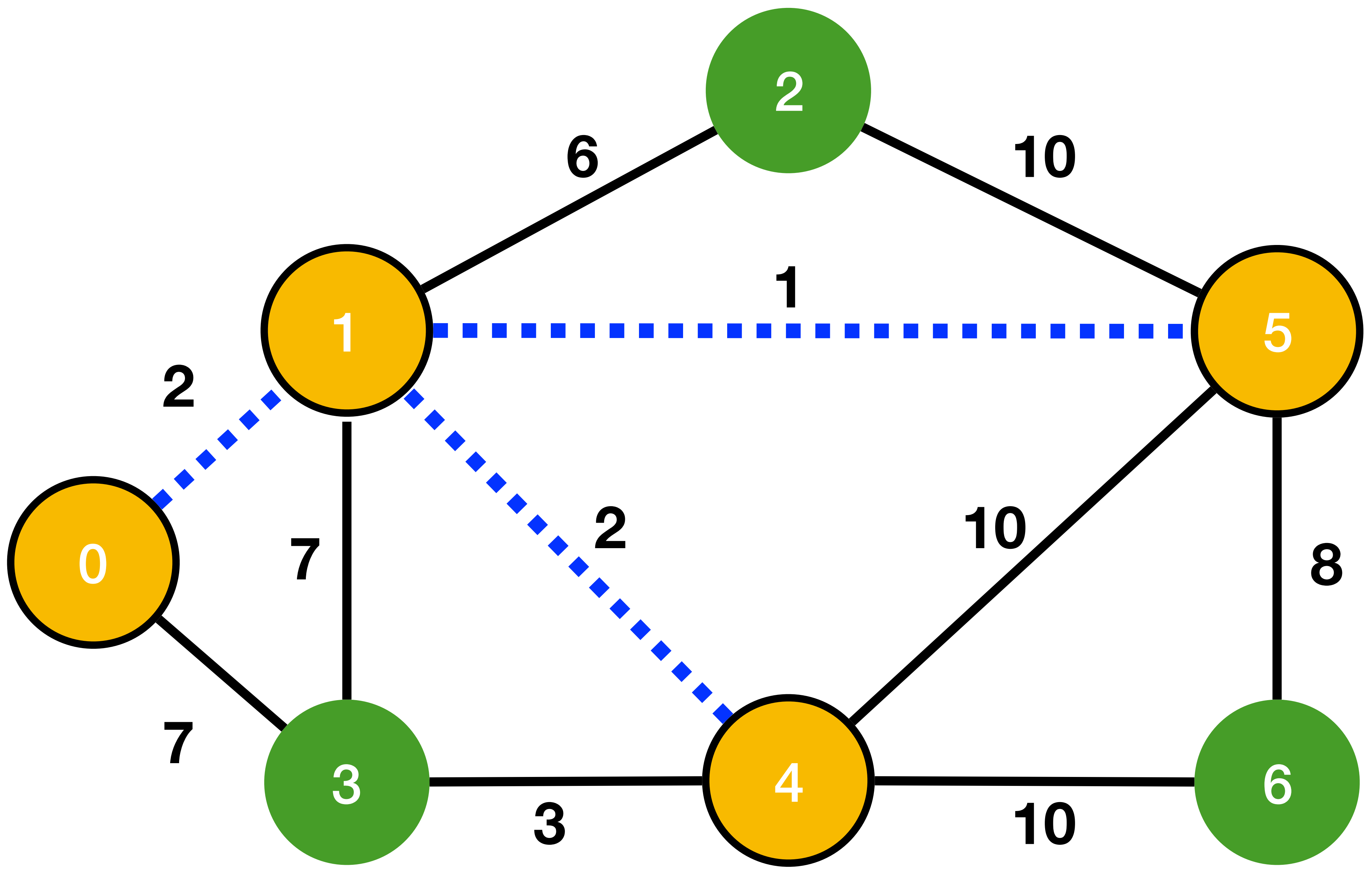


Minimum Priority Queue

id	2	3	6
key	6	7	8
P	1	1	5

Step 7: Move vertex with minimum key into MST
MST = {1, 5, 0, 4}
P = {-1, 1, 1, 1}

Example (Prim's)

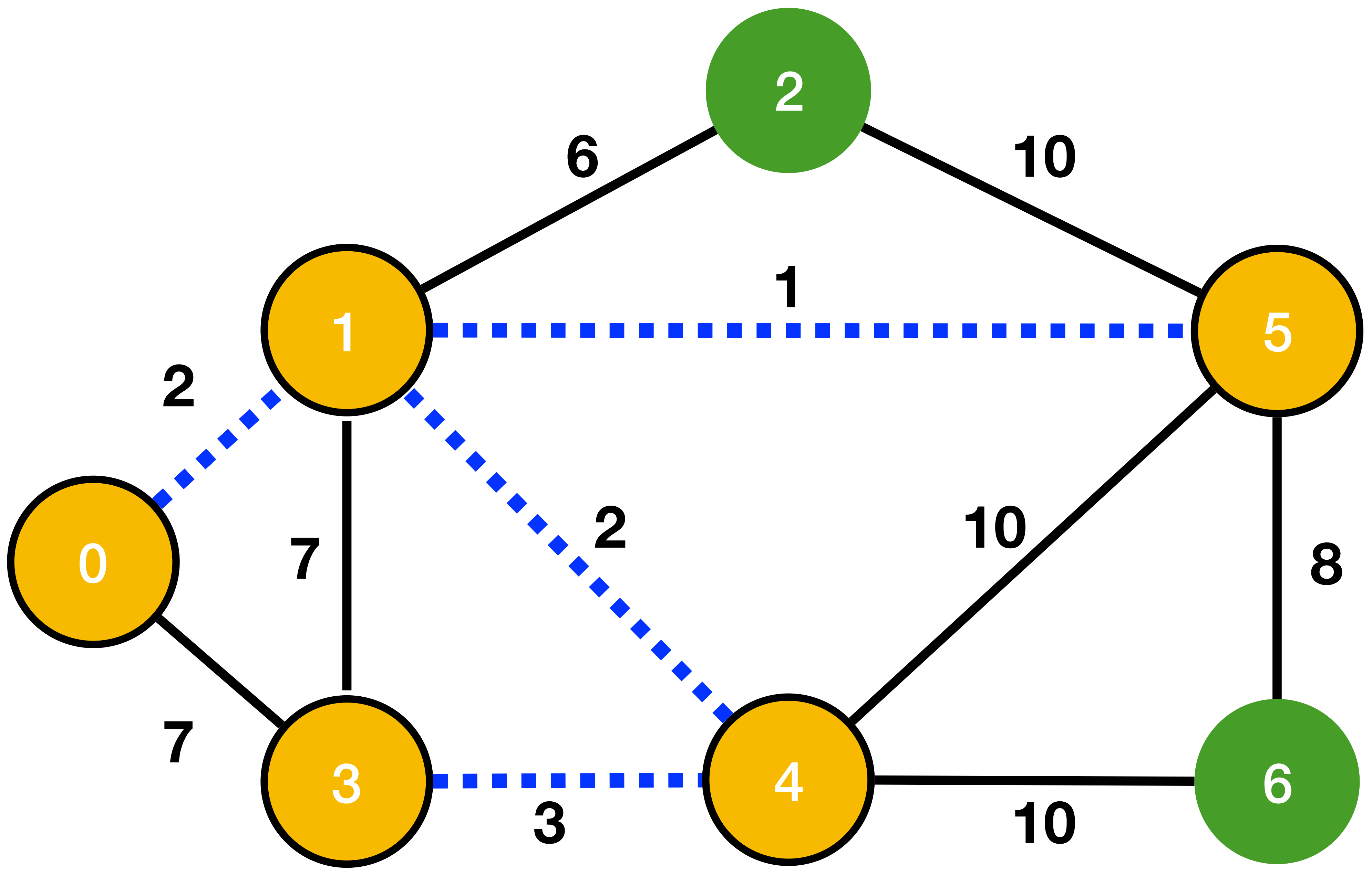


Minimum Priority Queue

id	2	**3**	6
key	6	**3**	8
P	1	**4**	5

Step 8: Update keys and P of vertices(v) adjacent to 4(u) if weight (4, v) < key (v)

Example (Prim's)



Minimum Priority Queue

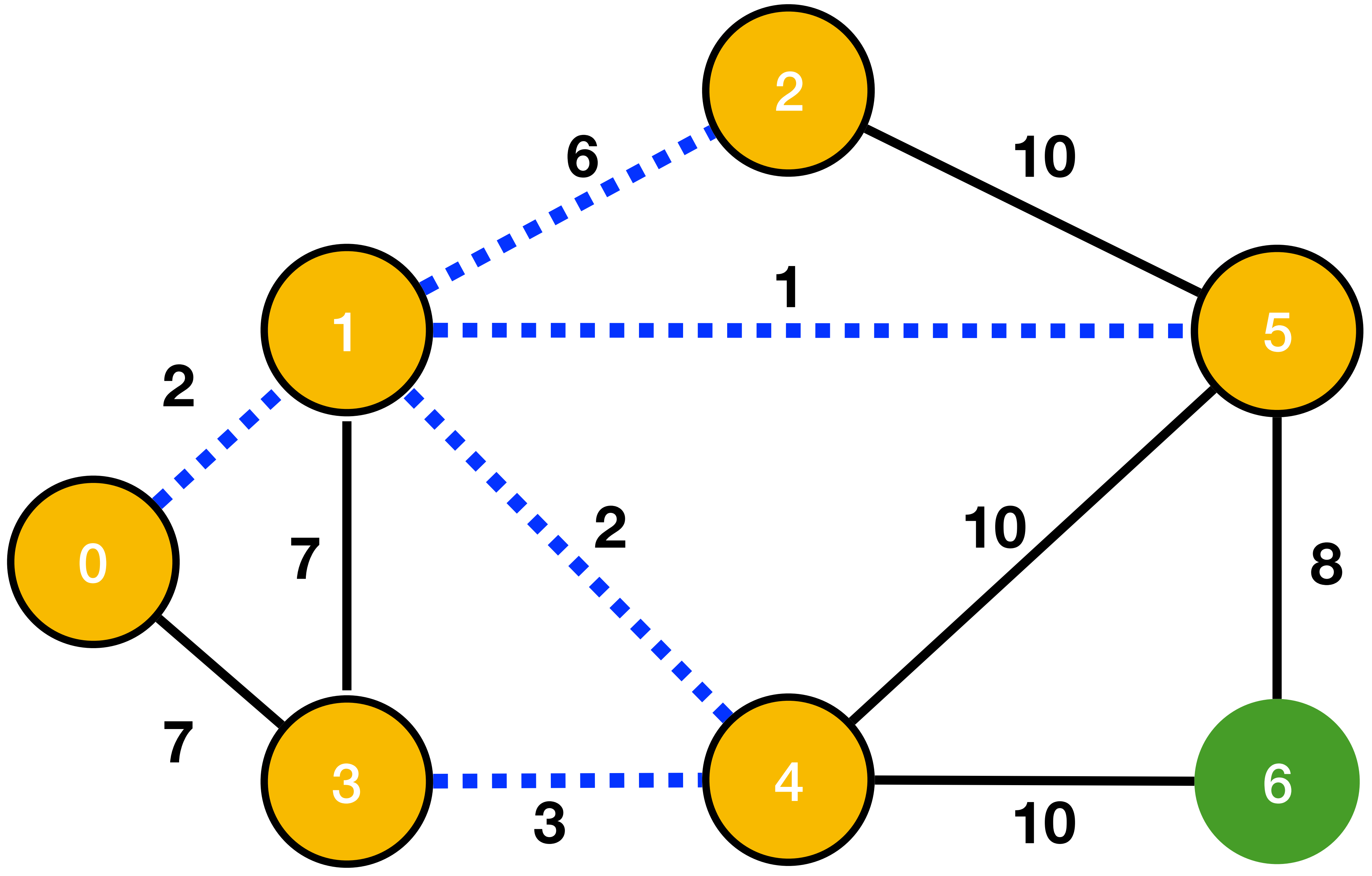
id	2	6
key	6	8
P	1	5

Step 9: Move vertex with minimum key into MST

MST = {1, 5, 0, 4, 3}

P = {-1, 1, 1, 1, 4}

Example (Prim's)

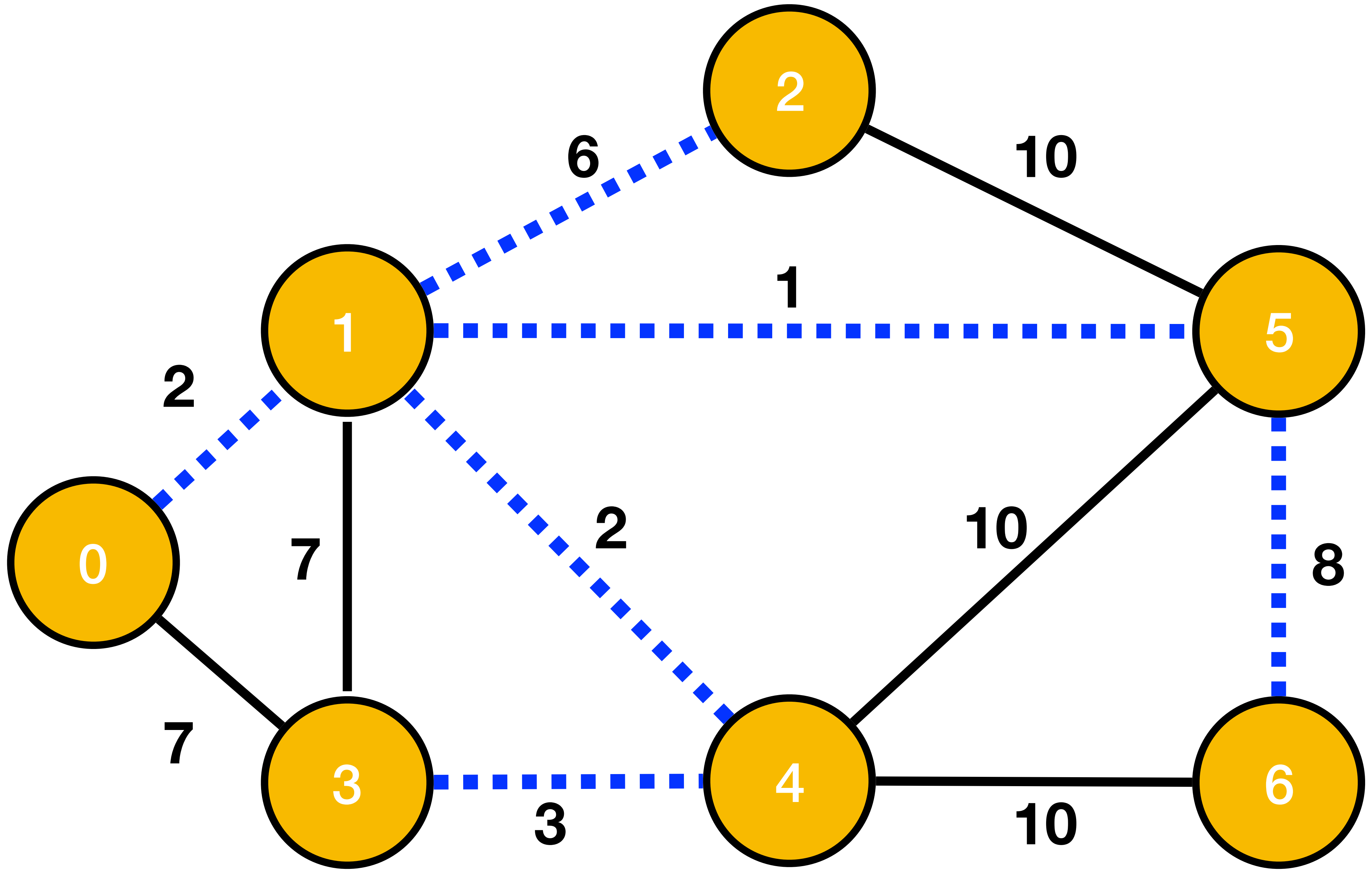


Minimum Priority Queue

id	6
key	8
P	5

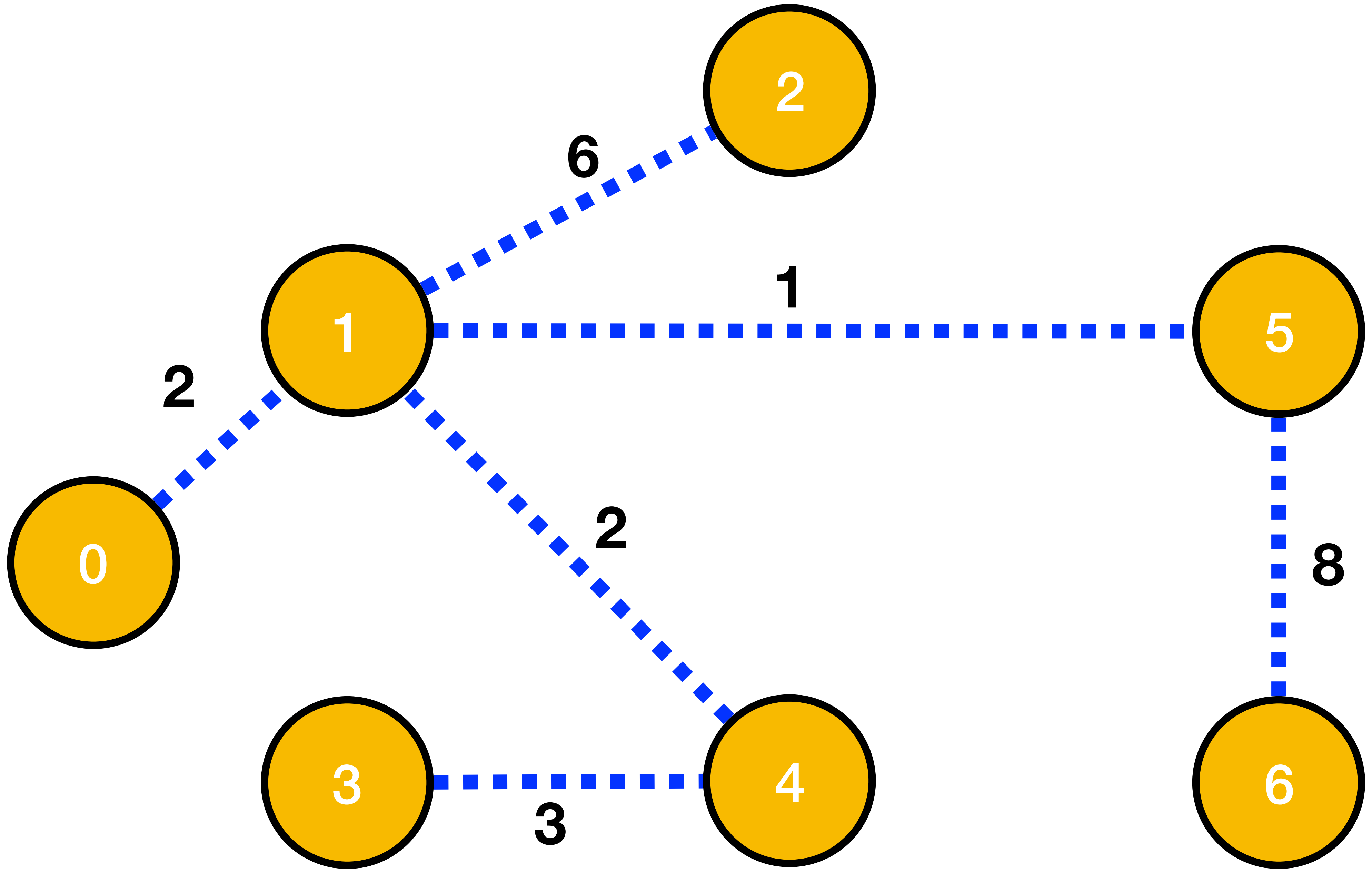
Step 10: Move vertex with minimum key into MST
MST = {1, 5, 0, 4, 3, 2}
P = {-1, 1, 1, 1, 4, 1}

Example (Prim's)



Step 11: Move vertex with minimum key into MST
MST = {1, 5, 0, 4, 3, 2, 6}
P = {-1, 1, 1, 1, 4, 1, 5}

Example (Prim's) - Solution



Minimum Cost = 22

Performance of Prim's algorithm

- With binary min-heap

  Step 1: Build min heap O(V)

  Step 2: Extract - min O(log V) - Repeat | V | times

  Step 3: Decrease - key O(log V) - Repeat O(E) times

Total time: O(V log V) + O(E log V) = O(E log V)

- With Fibonacci heap

  Step 1: Build min O(V)

  Step 2: Extract min O(log V) - Repeat | V | times

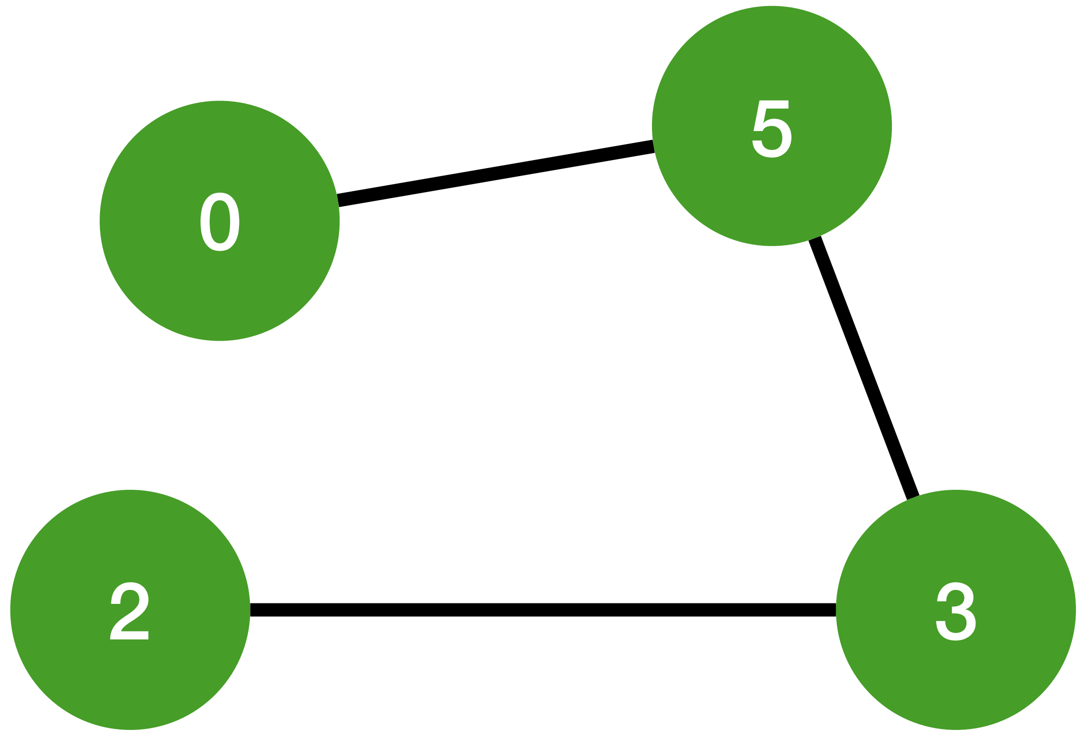
  Step 3: Decrease - key O(1) - Repeat O(E) times

Total time: O(V log V + E)

Kruskal's algorithm

Idea

- Sort the edges into nondecreasing order by weight
- Add lowest - cost edge to MST, if it does not create a cycle



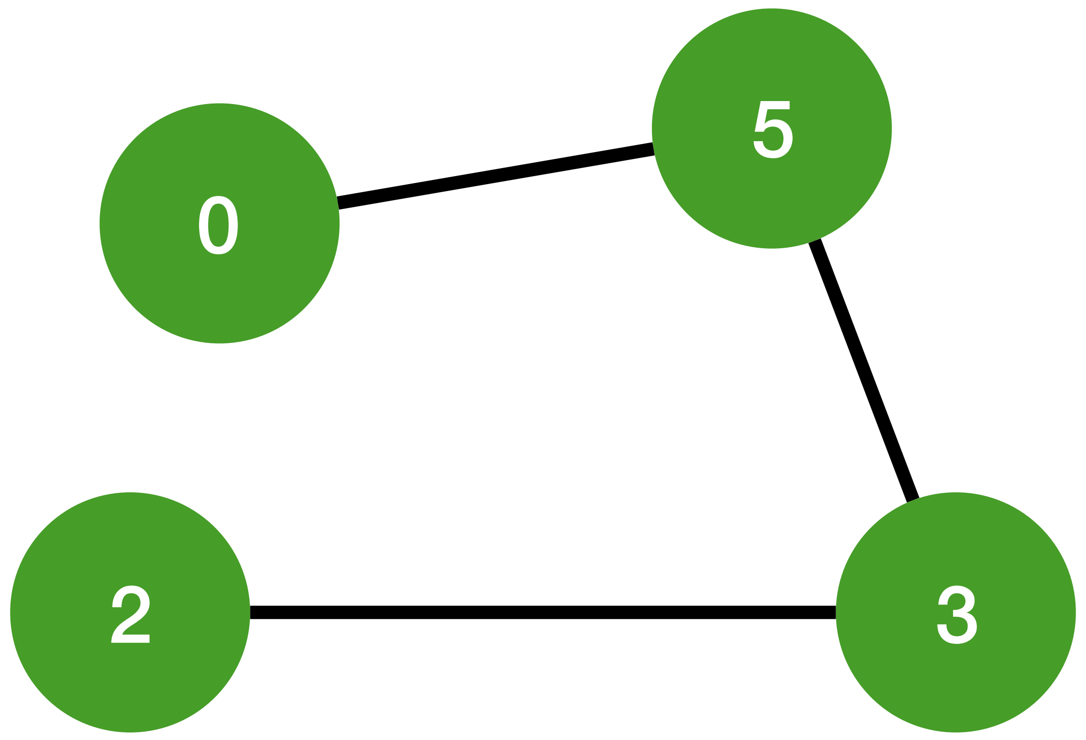
Adding edge (0,2) to MST will create a cycle

How to detect a cycle ?

Kruskal's algorithm

Detecting a cycle :

Union - Find algorithm - Check whether an undirected graph has a cycle



For each edge, create a subset using vertices

Add (0, 5)

0
5

Add (2, 3)

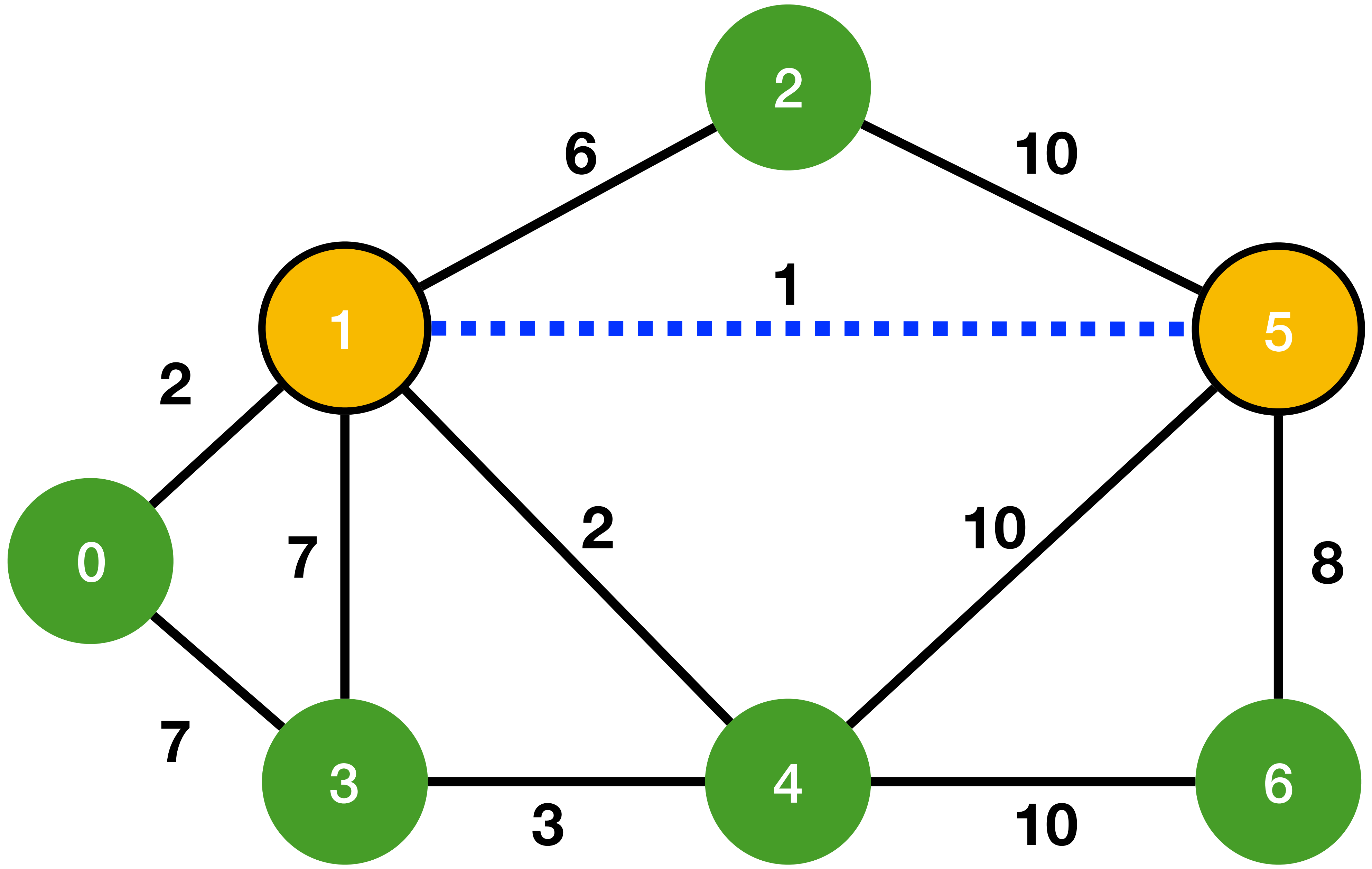
0	2
5	3

Add (3, 5) Merge the two sets

0	2	5	3

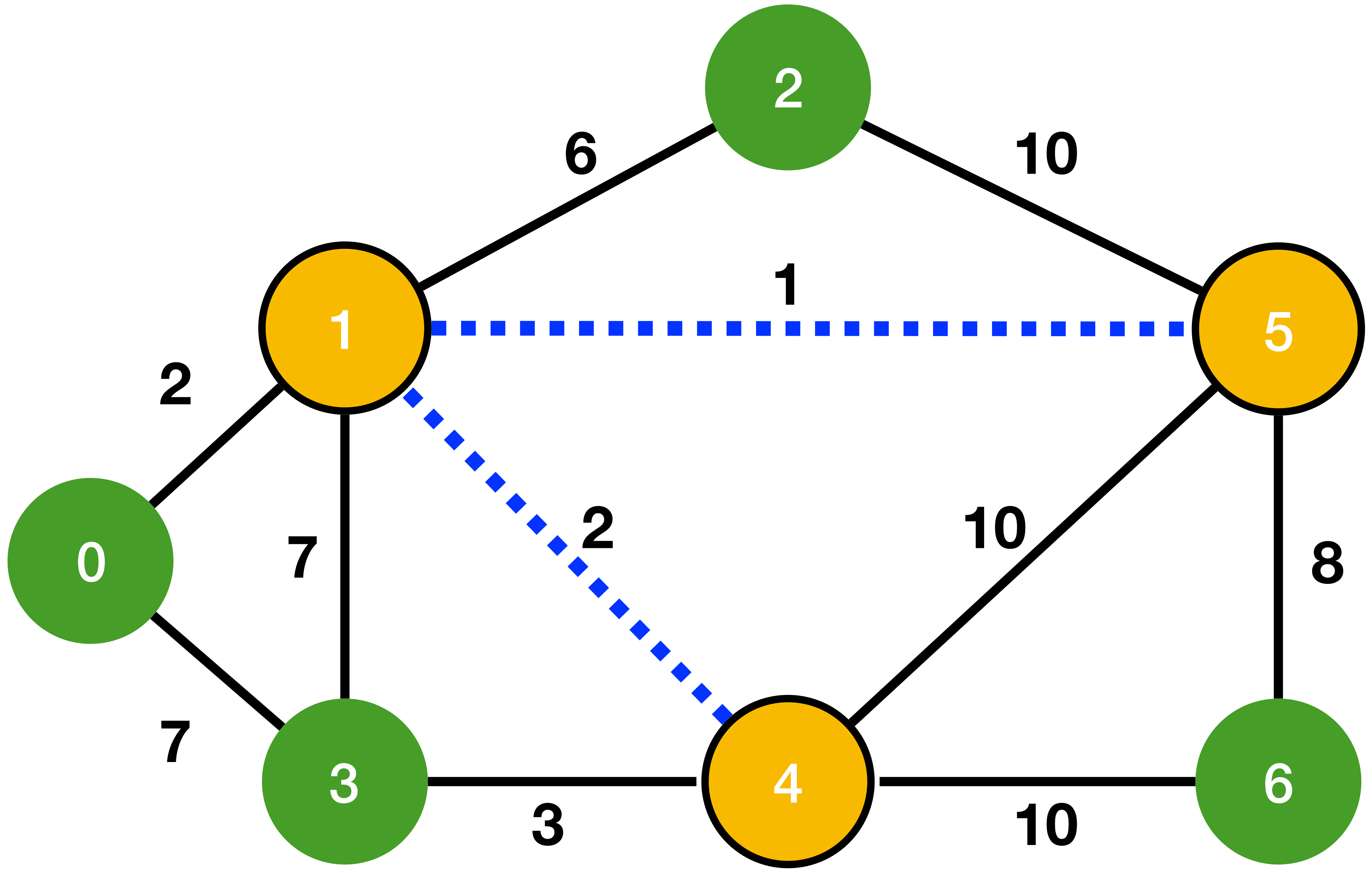
Add (0, 2) - in the same set => **Cycle**

Example (Kruskal's)



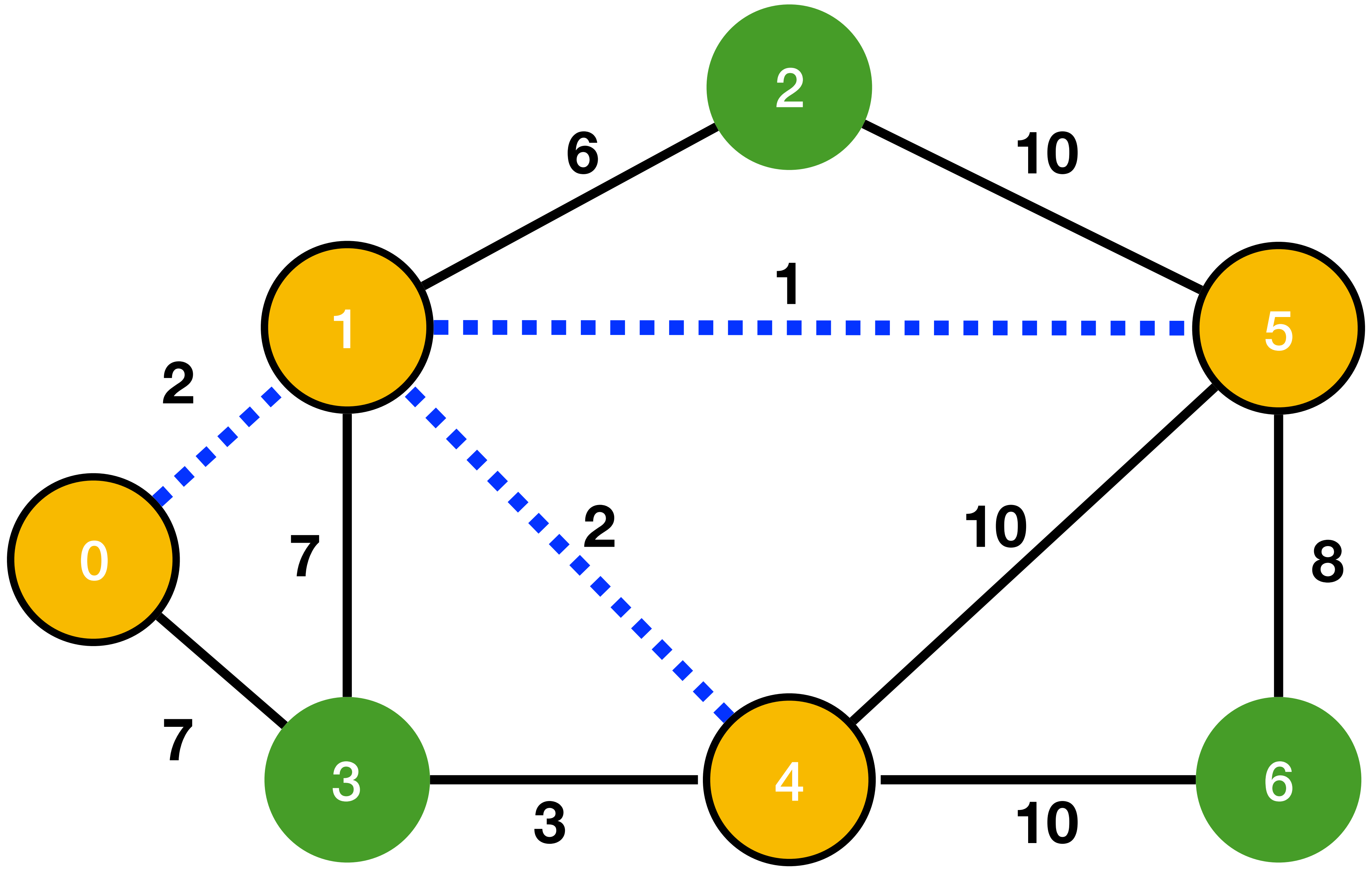
Select (1, 5) with lower weight

Example (Kruskal's)



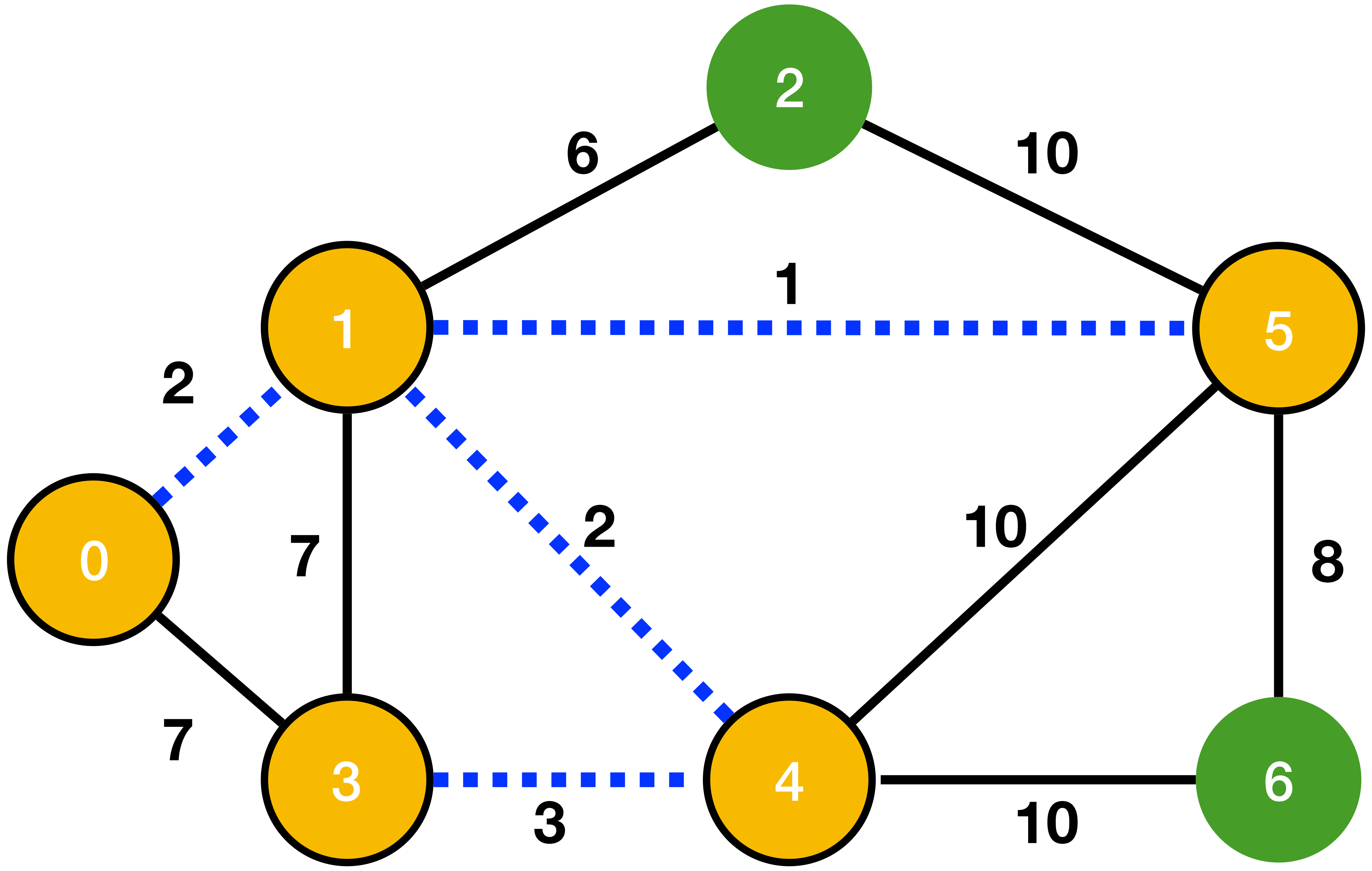
Select (1, 4) with 2nd lowest weight

Example (Kruskal's)



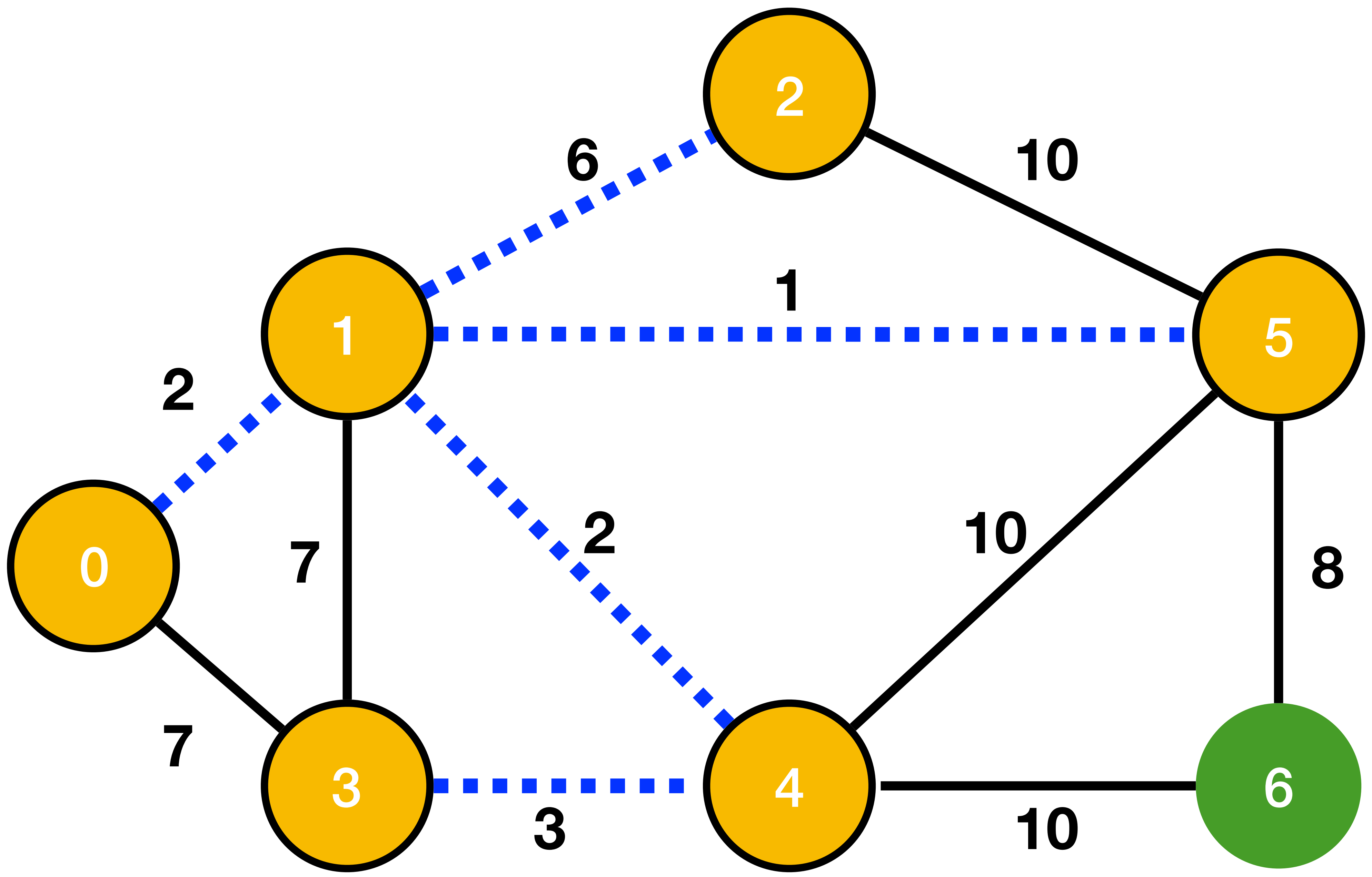
Select (1, 4) with 3rd lowest weight

Example (Kruskal's)



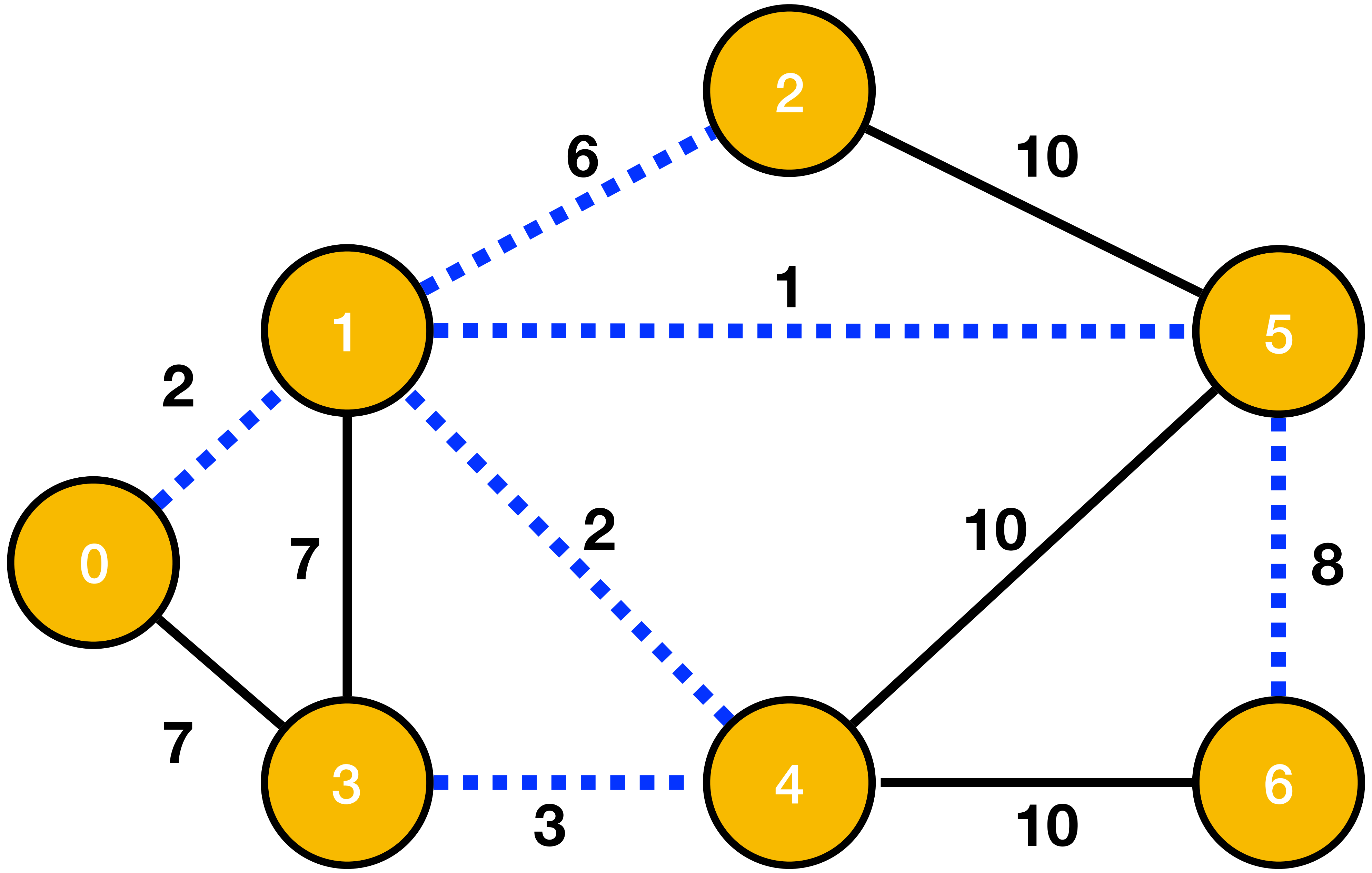
Select (1, 4) with 4th lowest weight

Example (Kruskal's)



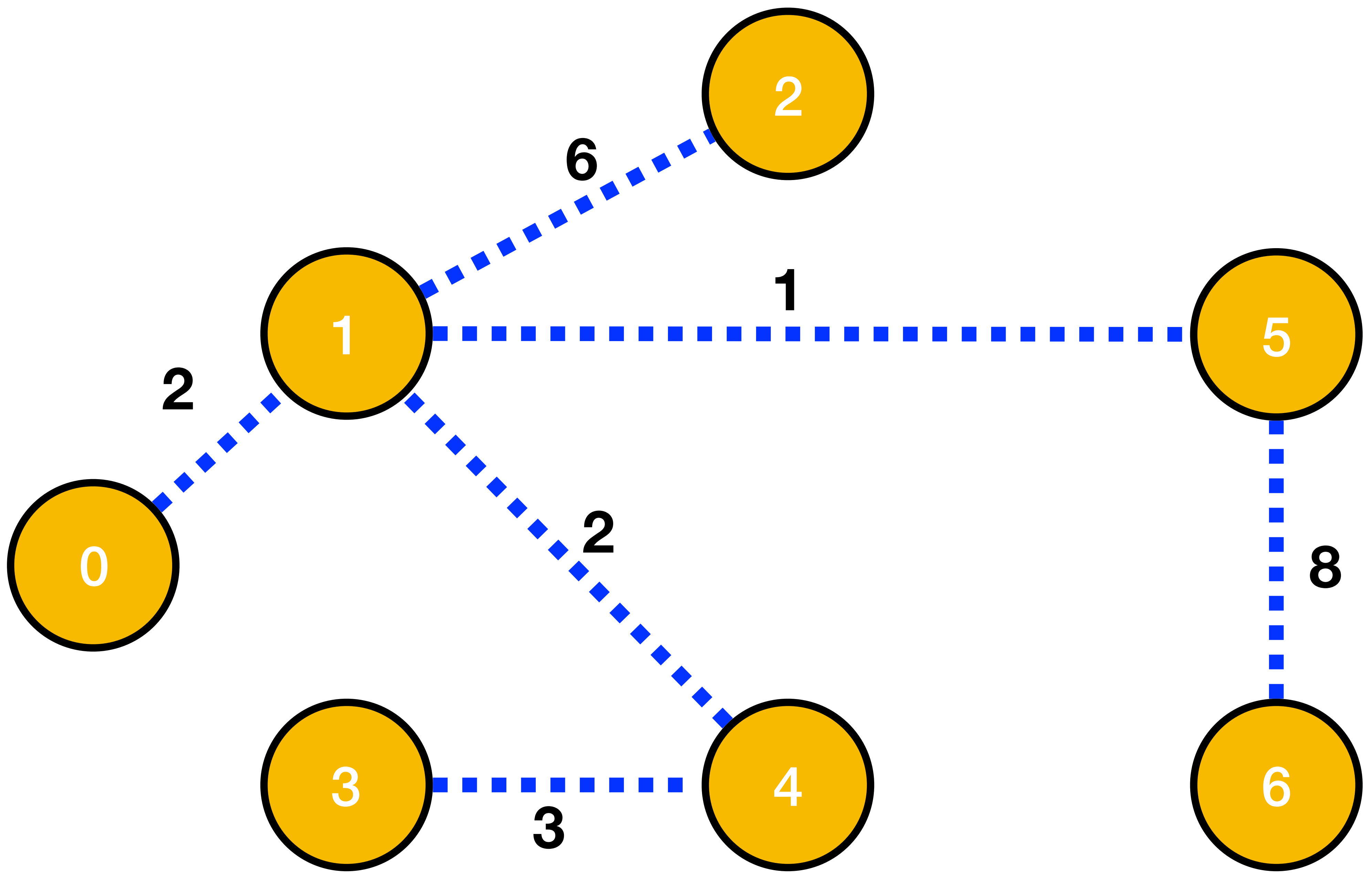
Select (1, 2)
Cannot select (1, 3) or (0,3) Why ?

Example (Kruskal’s)



Select (5, 6)

Example (Kruskal's)



Minimum Cost = 22

Kruskal's algorithm (informal)

1. Add each vertex of the graph to a separate set to create disjoint set T
2. Sort the edges of E into non decreasing order by weight w
3. While T is not spanning:
    - Remove an edge with minimum weight e_{min}
    - if e_{min} connects two different sets (T_1 and T_2)

      add it to T and combine the set T_1 and T_2 into a single set ;
    - else

      discard e_{min} ;
4. return T

When the algorithm terminates, the forest T will contain only one connected component.