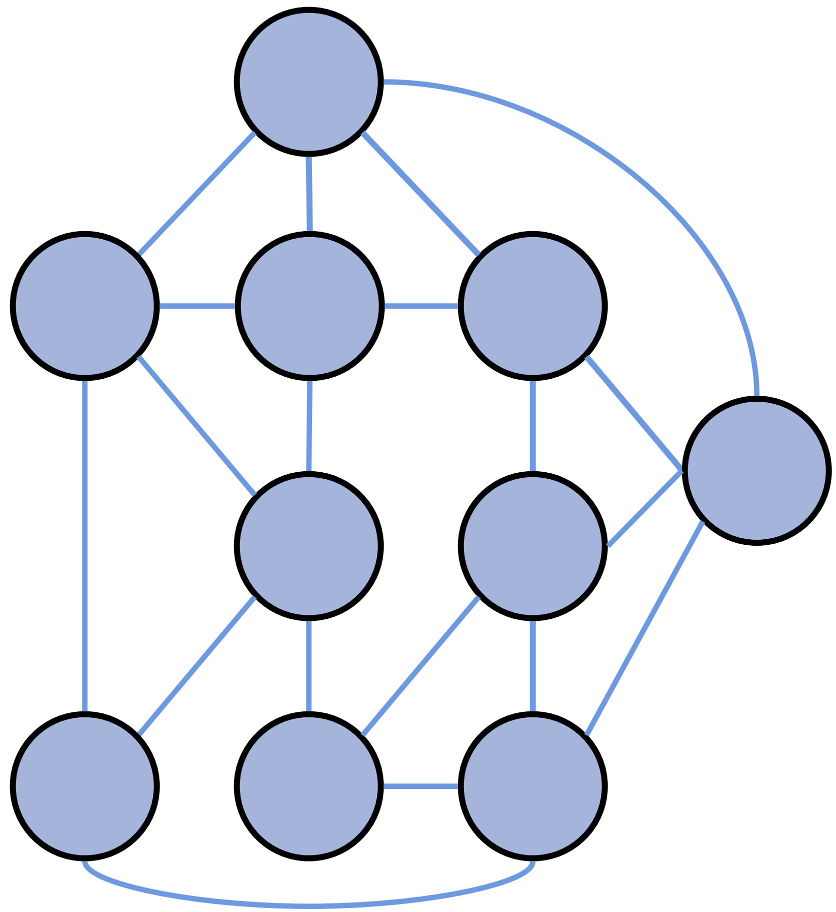
MST
(Minimum Spanning Tree)

Kruskal’s Algorithm
Guan’s Algorithm

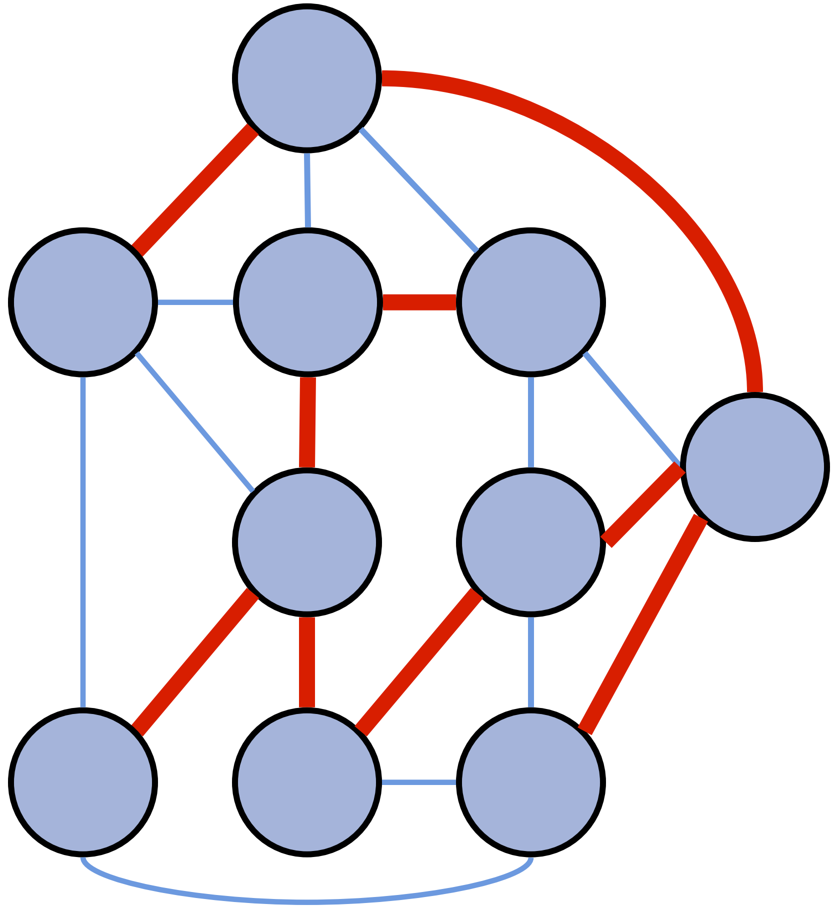
Modified and adapted from Ooi Wei Tsang’s notes

Input Graph

Want "minimal" graph that is still *fully connected*



Spanning Tree



Minimum Spanning Tree

Problem:

- Input: A graph G, with costs on the edges.
- ToDo: Find a spanning tree where total cost is minimum.

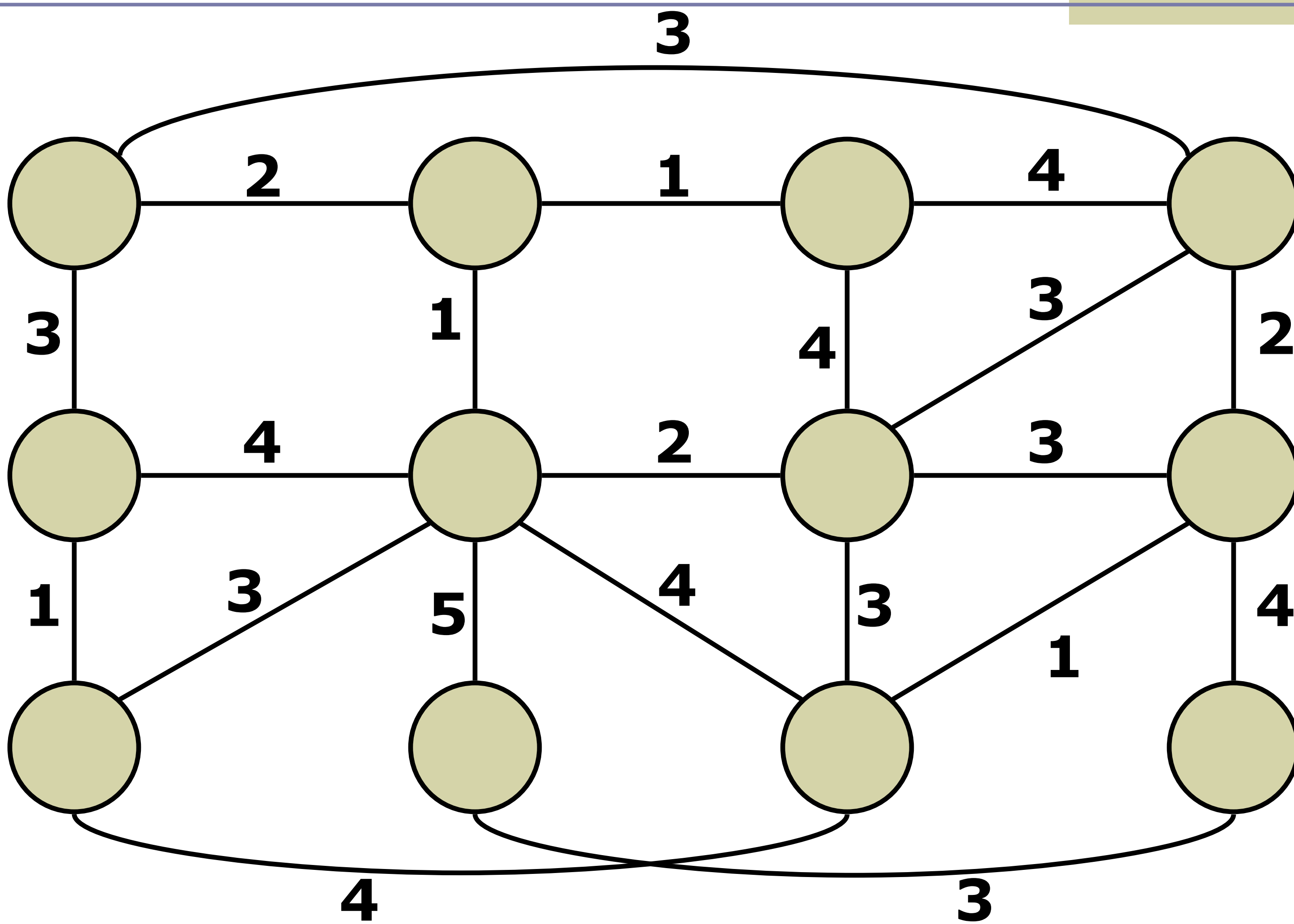
Kruskal's MST Algorithm

- Joseph Kruskal, 1956

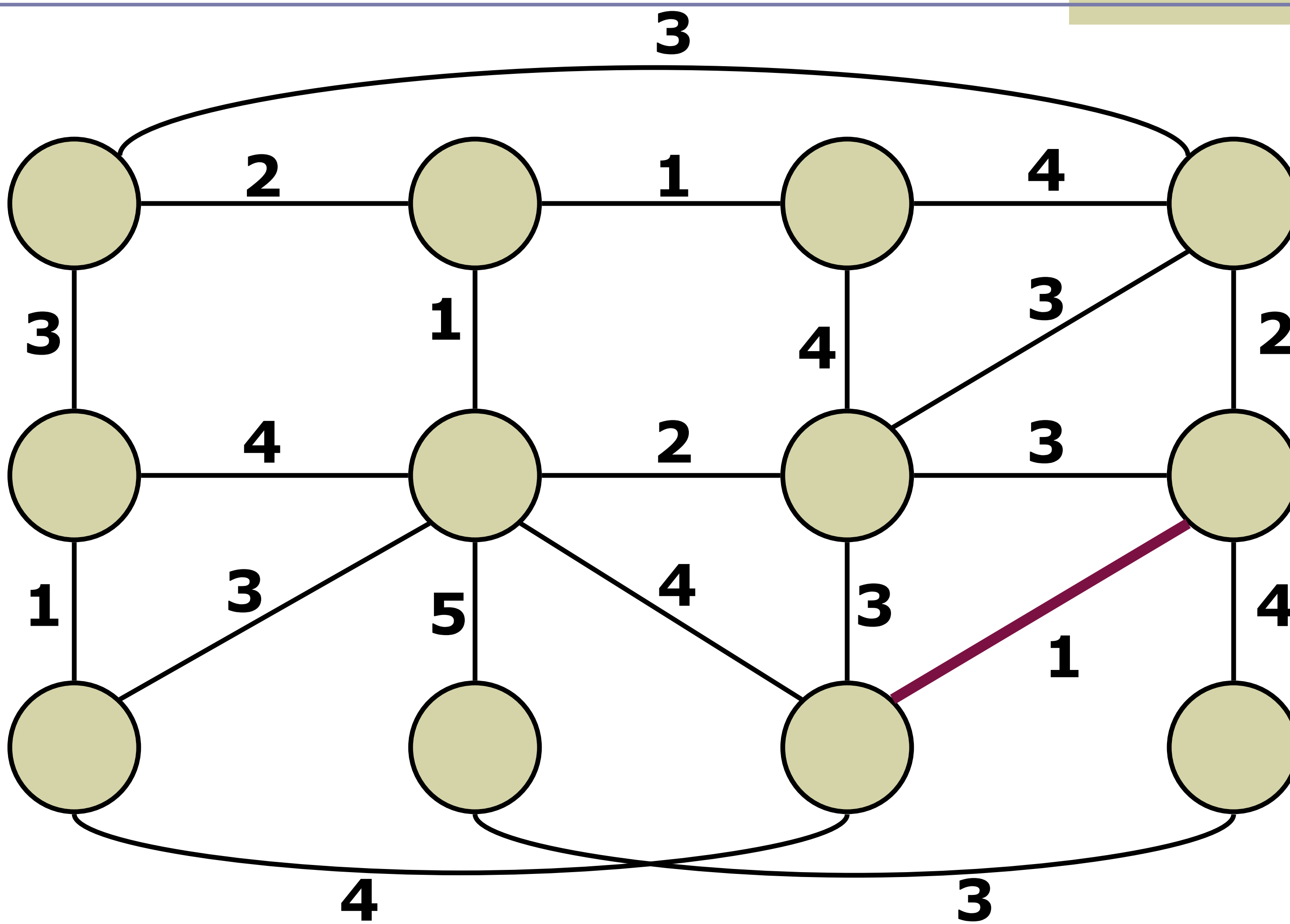
2. ^ Kruskal, J. B. (1956). "On the shortest spanning subtree of a graph and the traveling salesman problem". *Proceedings of the American Mathematical Society* **7**: 48–50. doi:10.1090/S0002-9939-1956-0078686-7. JSTOR 2033241.

Idea:
"Repeatedly,
add shortest edge whenever possible"

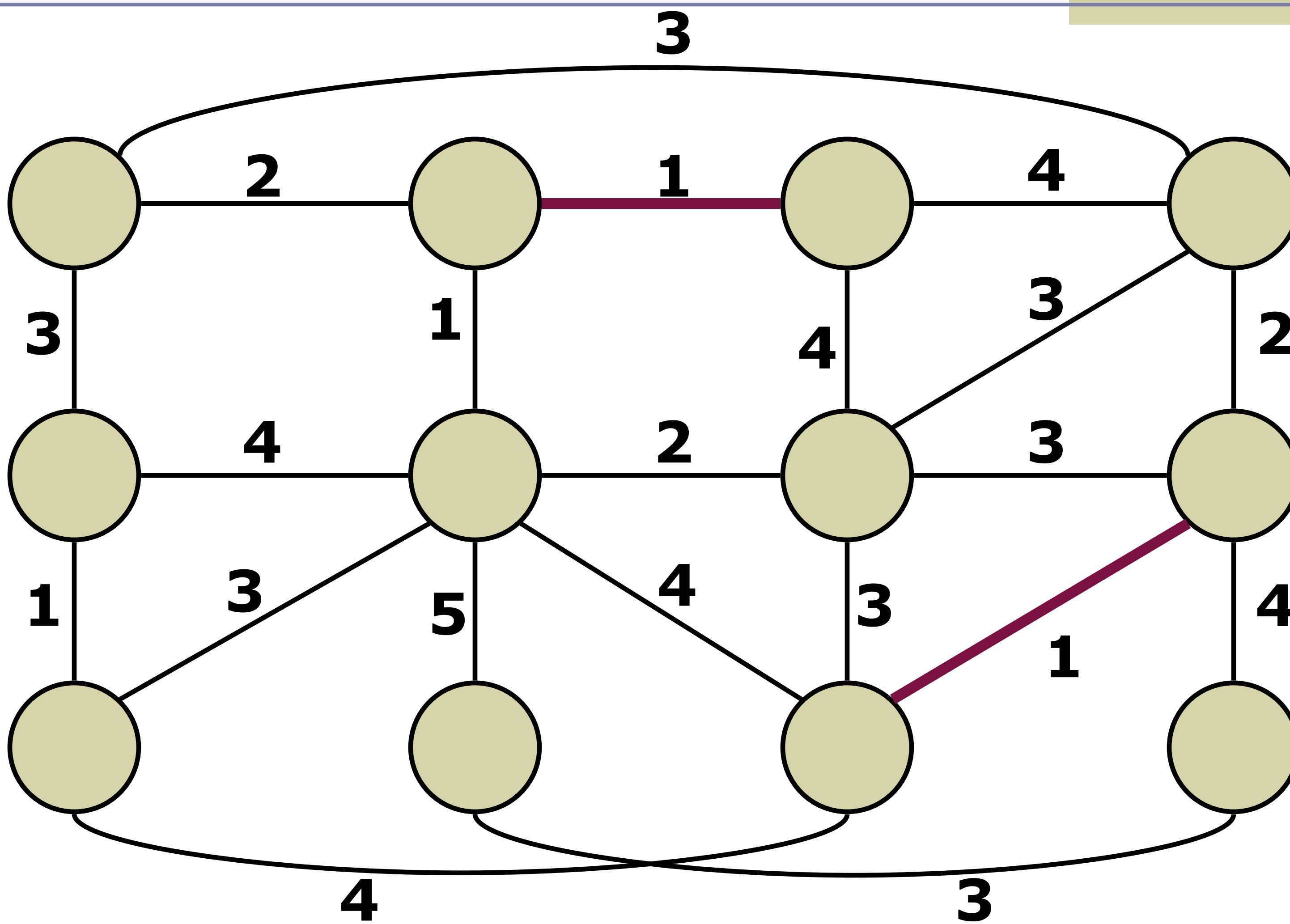
Kruskal's Algorithm



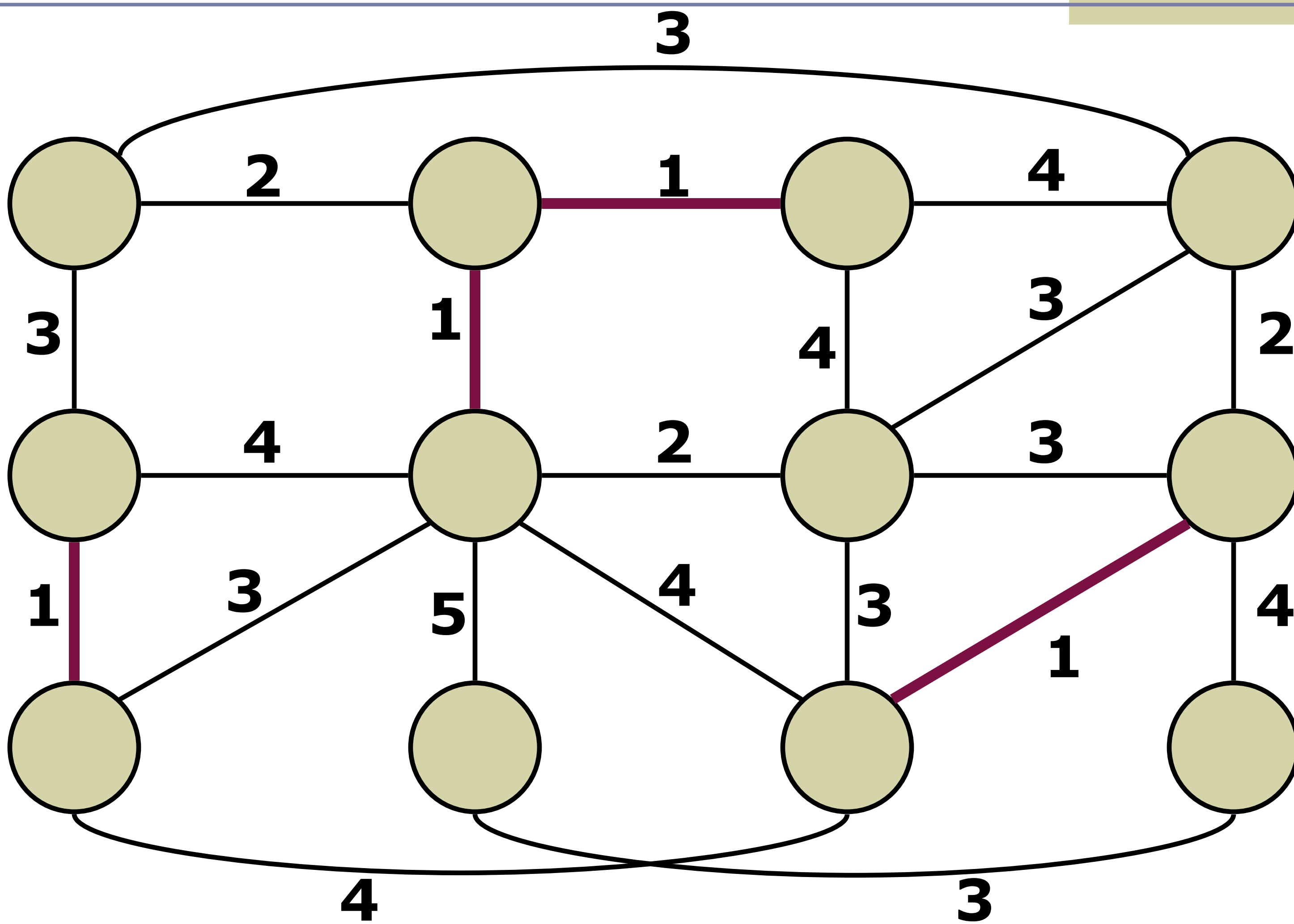
Kruskal's Algorithm



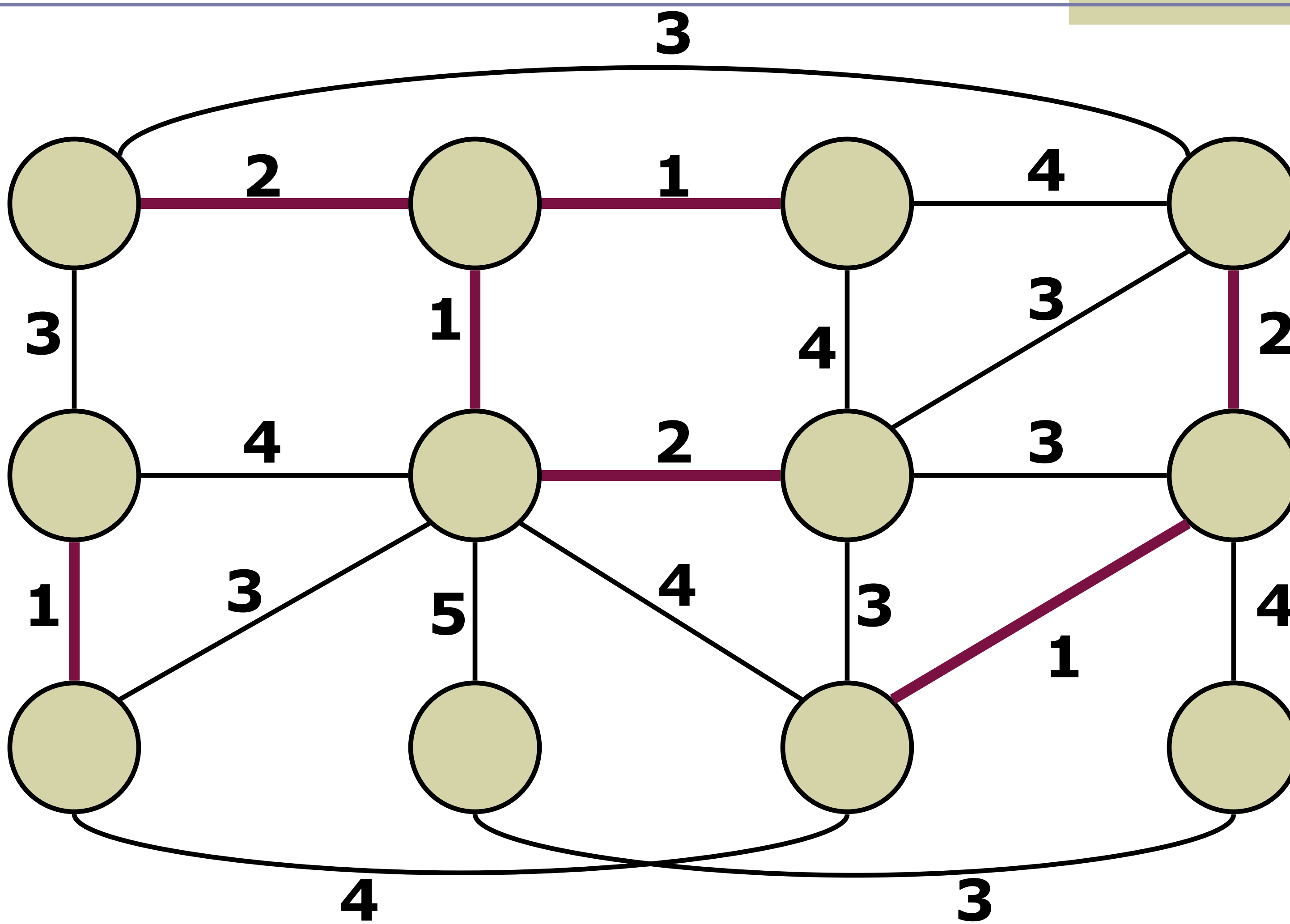
Kruskal's Algorithm



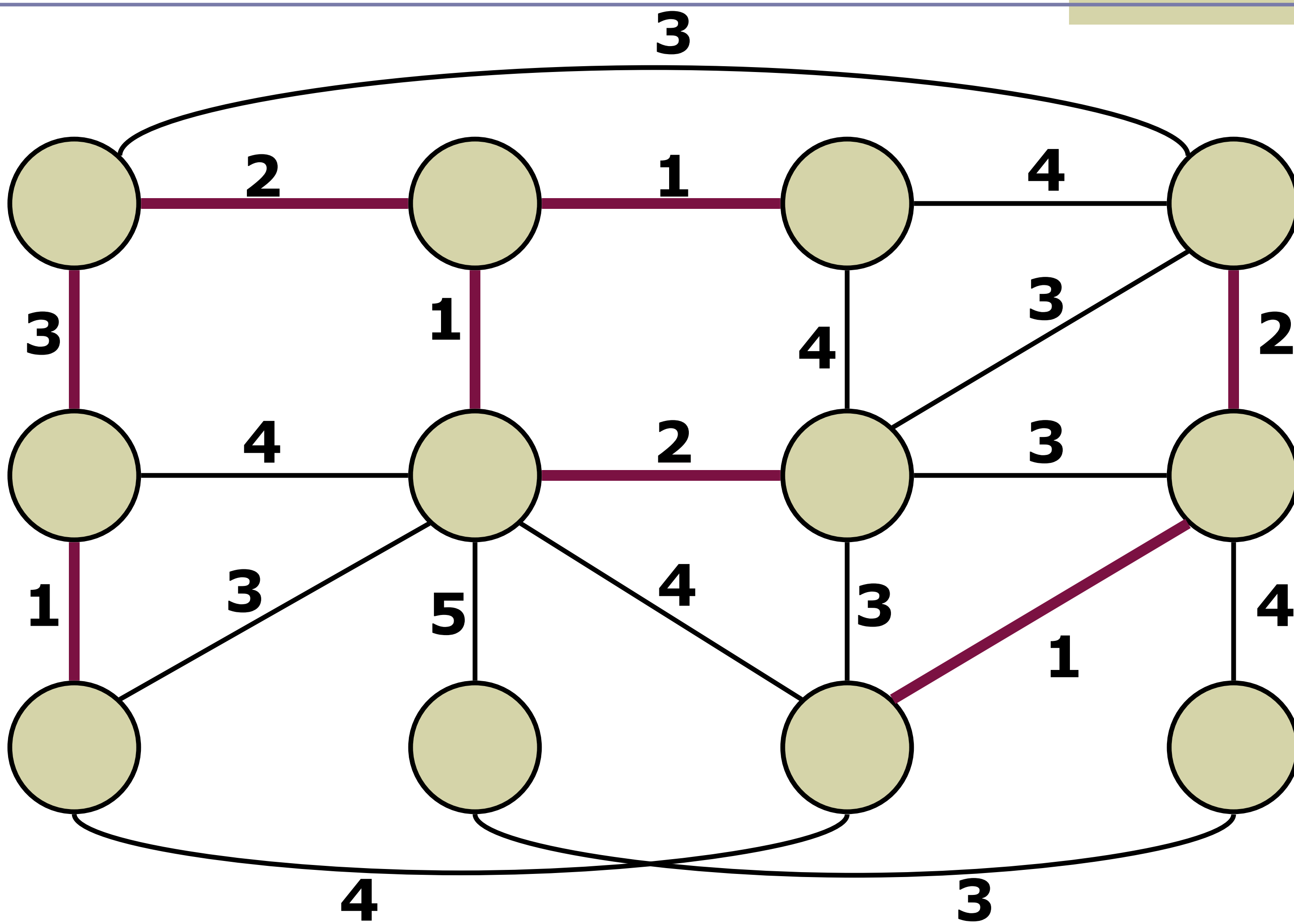
Kruskal's Algorithm



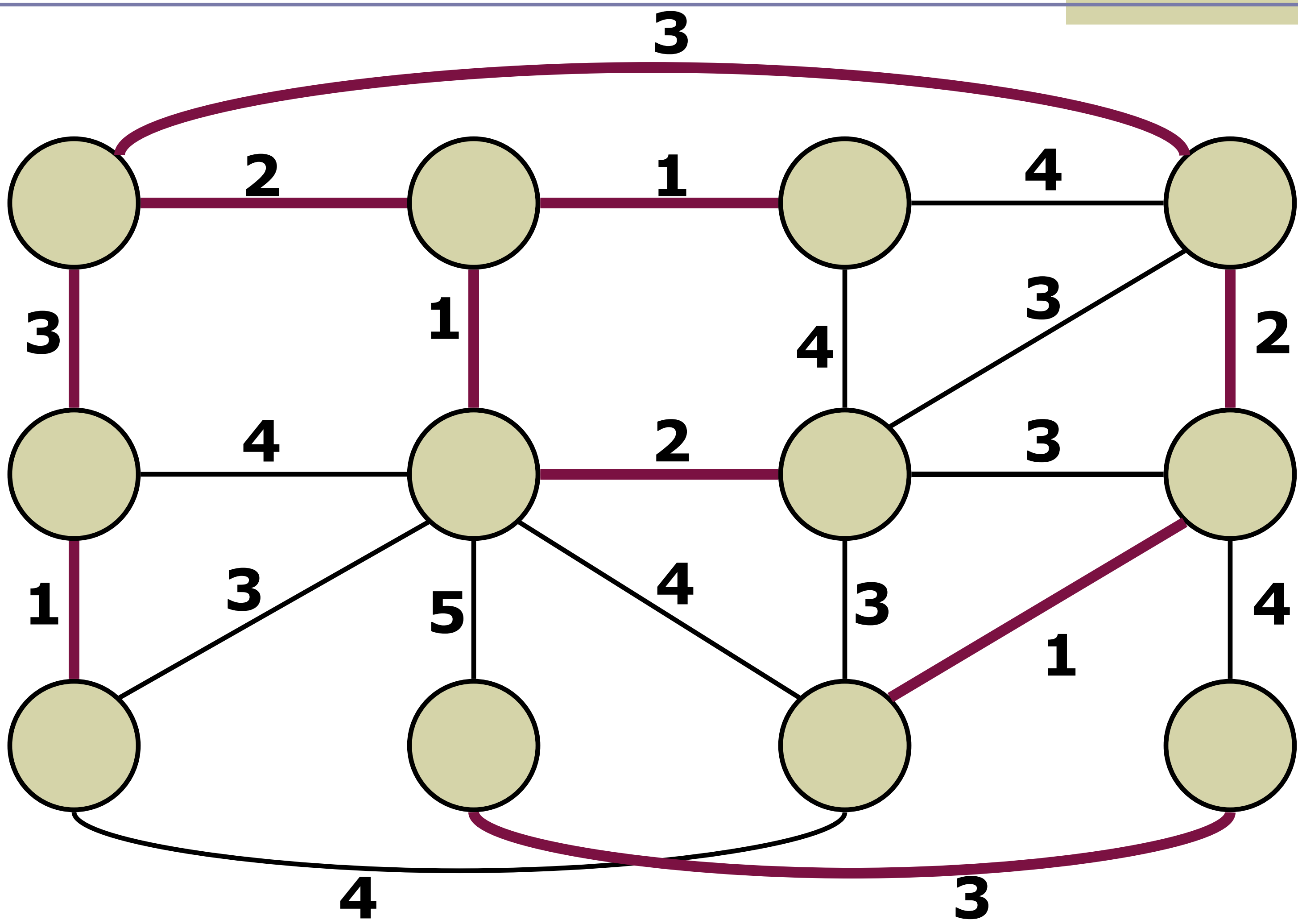
Kruskal's Algorithm



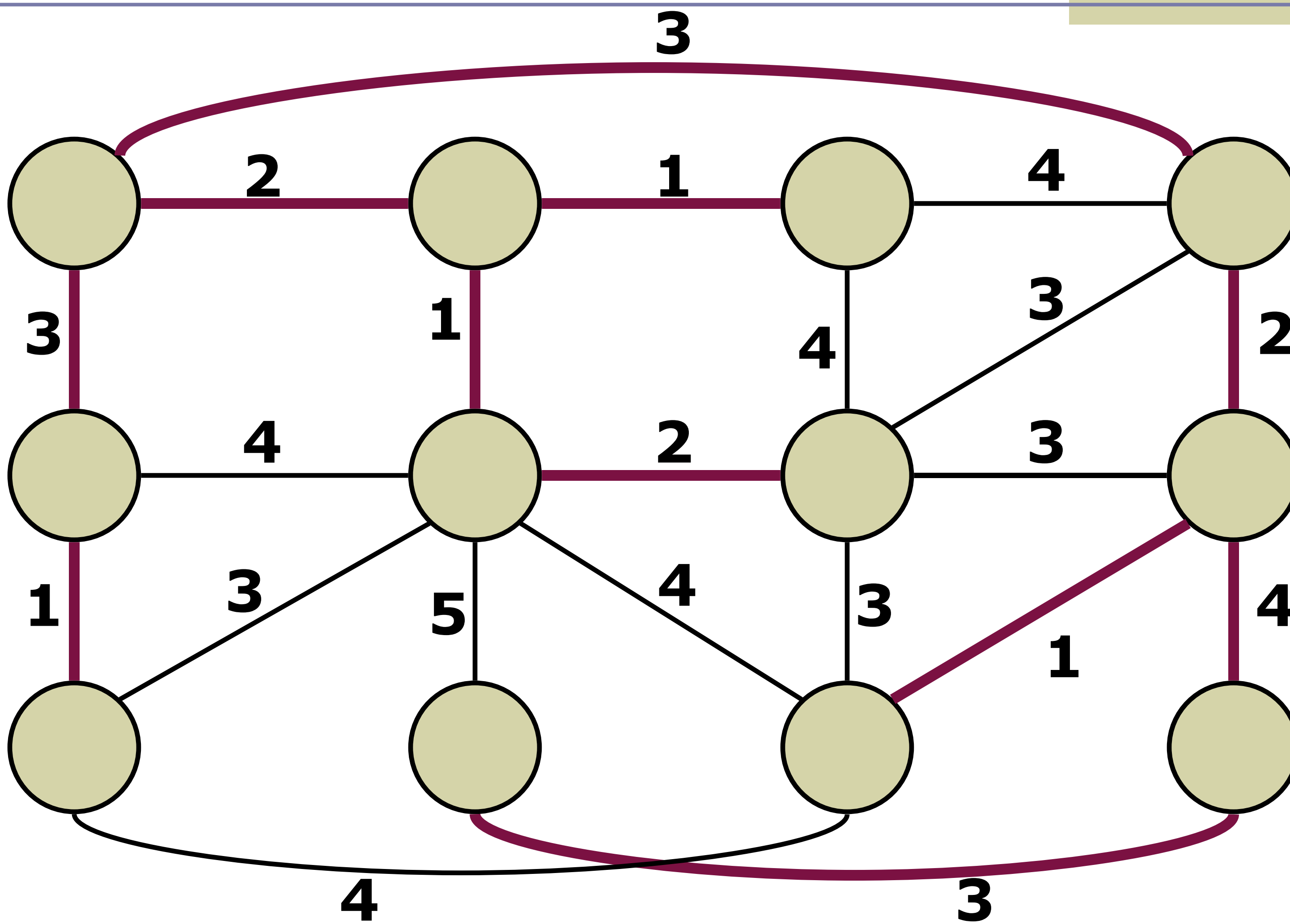
Kruskal's Algorithm



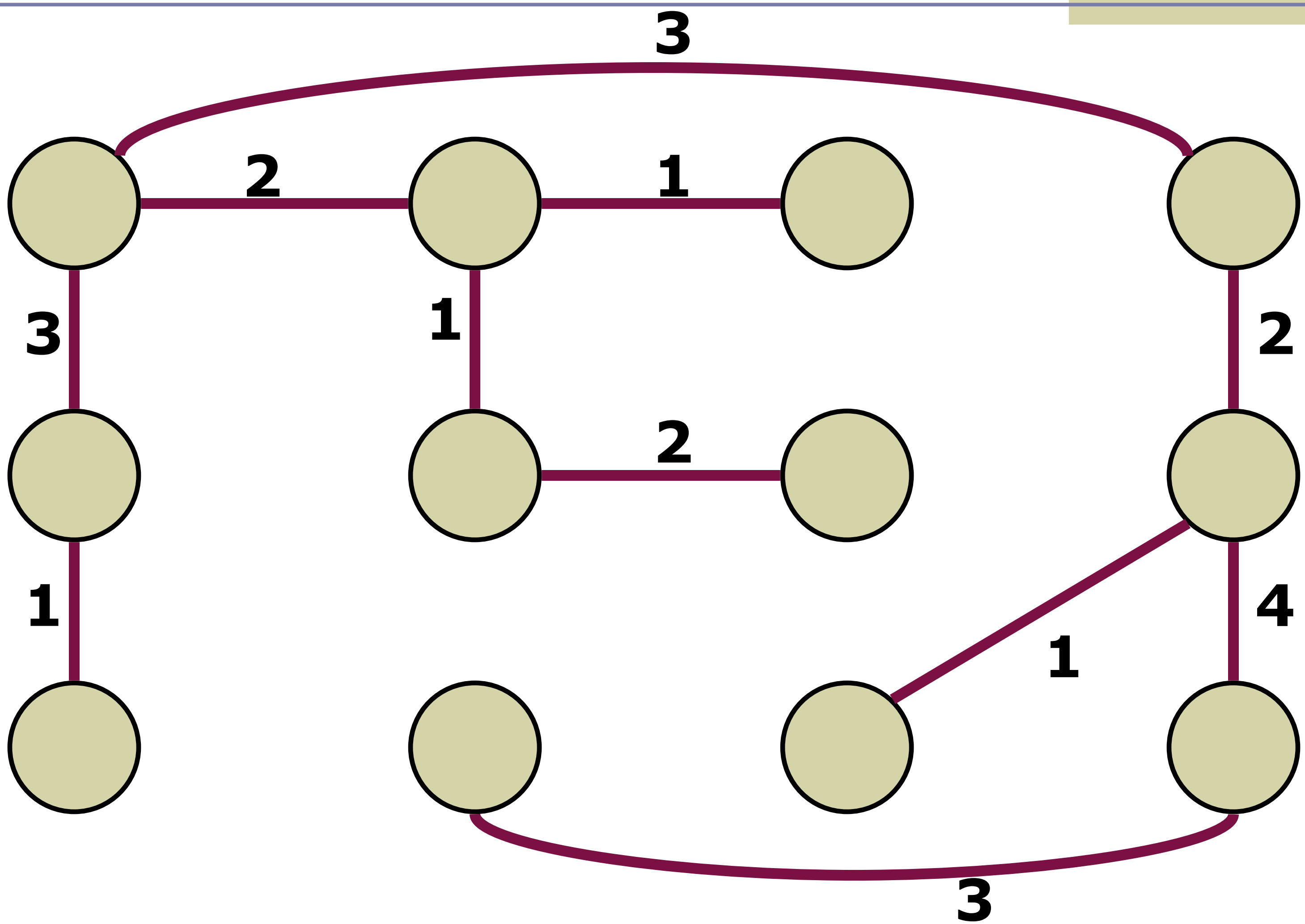
Kruskal's Algorithm



Kruskal's Algorithm



Kruskal's Algorithm



Kruskal's Algorithm

while there are unprocessed edges left
    pick an edge e with *minimum* cost
    if adding e to MST does not form a cycle
        add e to MST
    else
        throw e away

Data Structures

- How to pick edge with minimum cost?
  - Use a Priority Queue
- How to check if adding an edge can form a cycle?
  - Use a Disjoint Set

Data Structures needed

Guan's MST Algorithm

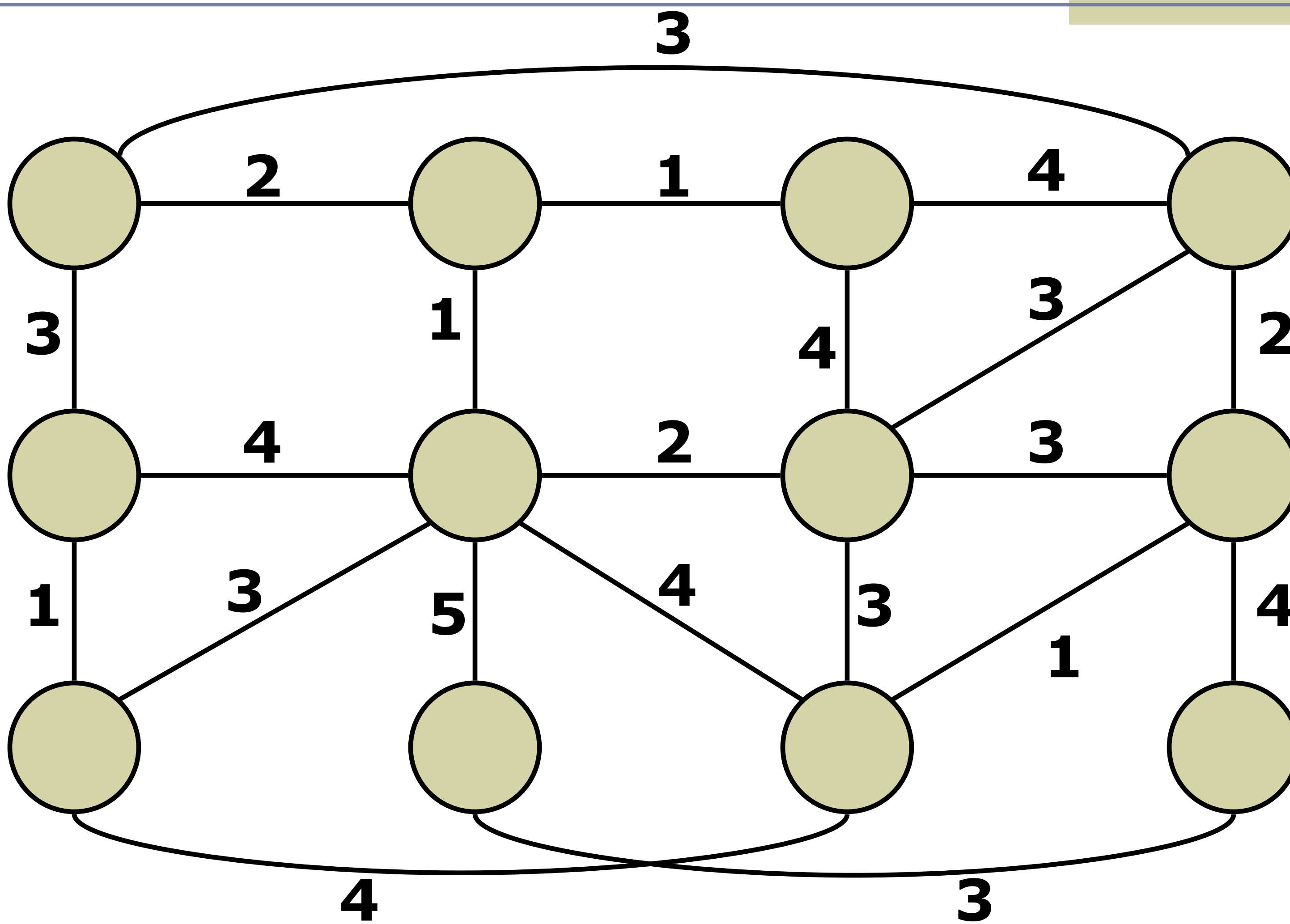
- Guan Meigu (管梅谷), 1975
  Shandong Normal University

[Gua75] Guan Mei-Gu. The method of eliminating cycles for finding minimum spanning trees (in Chinese). *Shuxue de Shijian yu Renshi* 4 (1975).

Idea:
"In any cycle in the graph,
remove the longest edge."

LeongHW met Prof Guan Meigu in 1979 in a mathematics conference in Nantah (Nanyang University, 南洋大学)

Guan's Algorithm



DIY time…

Review…

- MST – minimal graph that is fully connected
- Kruskal’s algorithm – keep adding short edges (whenever possible)
- Guan’s algorithm – keep removing long edges in a cycle (while keeping it fully connected)

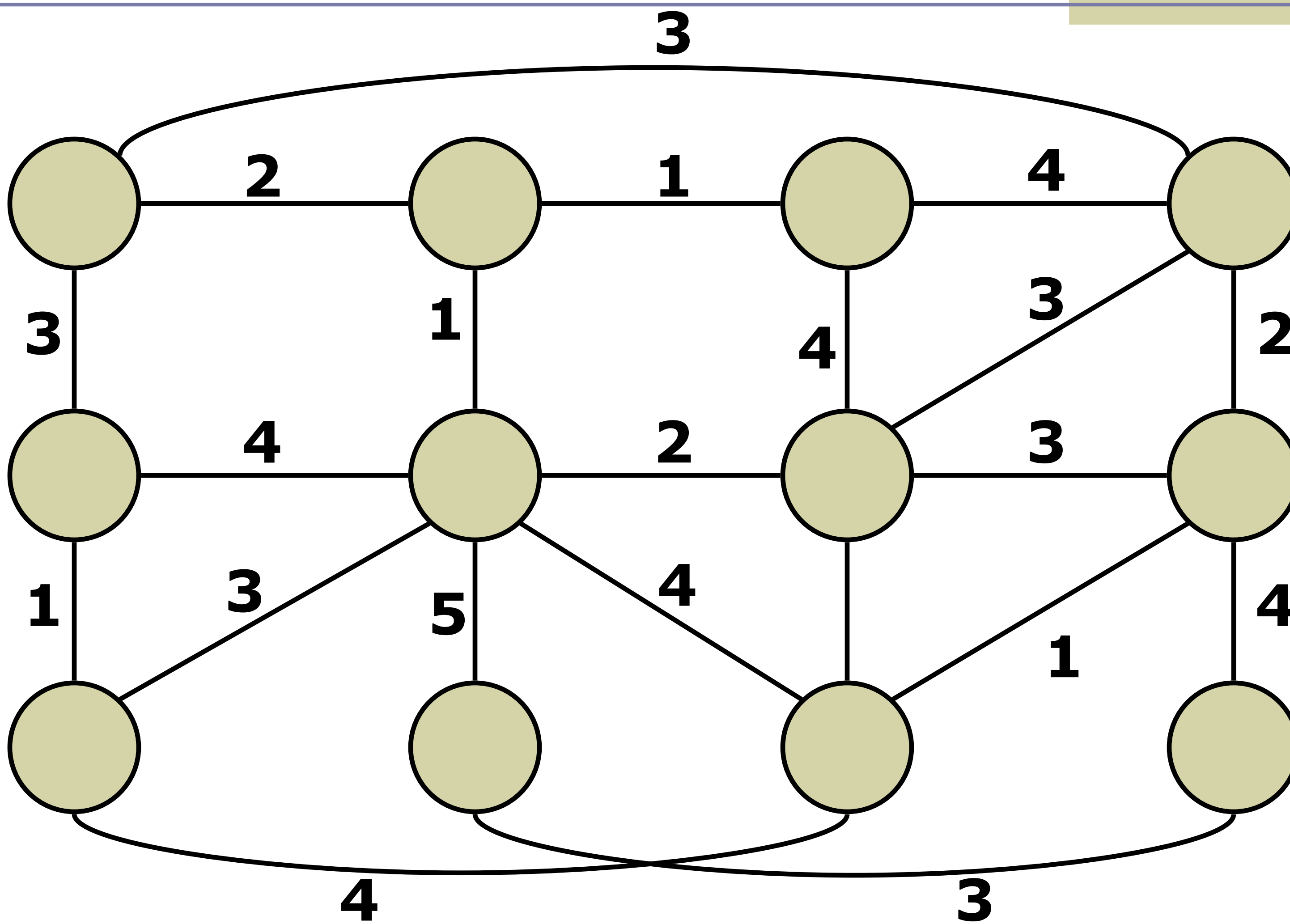
Prim-Dijkstra MST Algorithm

- R. C. Prim, "Shortest Connection Networks and some Generalizations, Bell System Tech. J, 36, (1957), pp. 1389-1401.
- E. W. Dijkstra, "A note on two problems in connections with graphs," Numerical Math, 1, (1959), pp. 269-271.

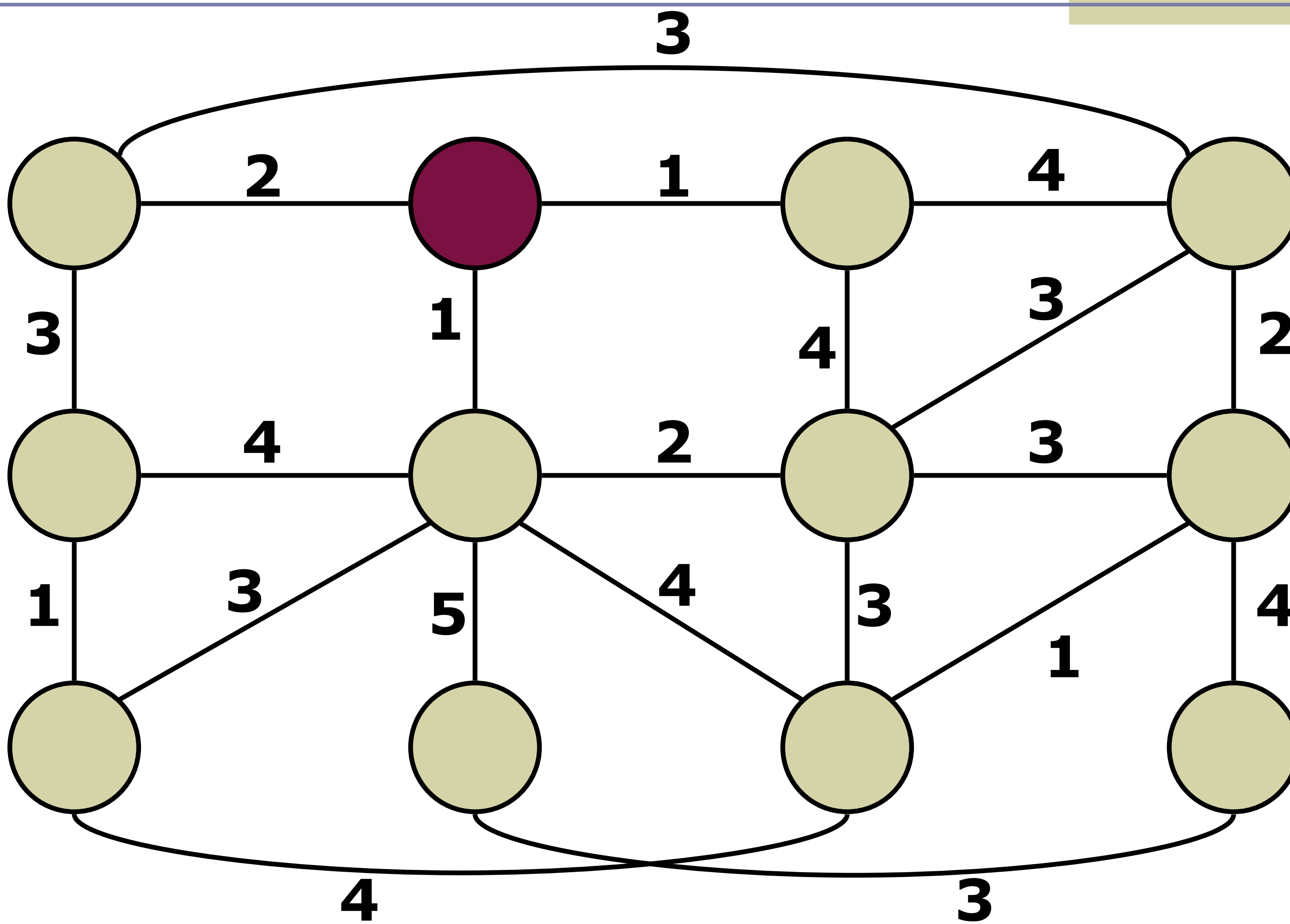
Idea:

"Repeatedly, add shortest edge connecting a red vertex (in A) with a yellow vertices (in (V-A))"

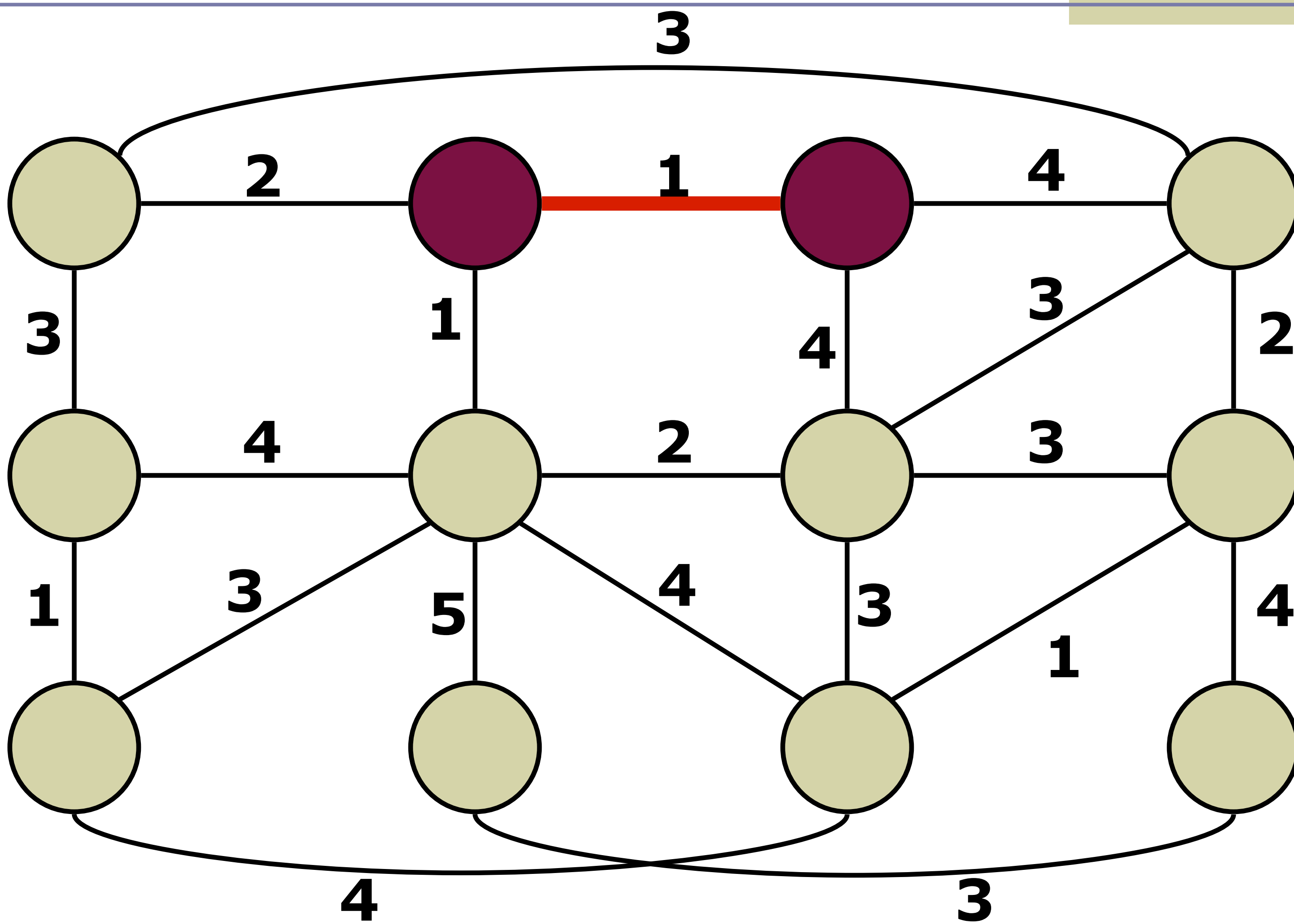
Prim-Dijkstra Algorithm



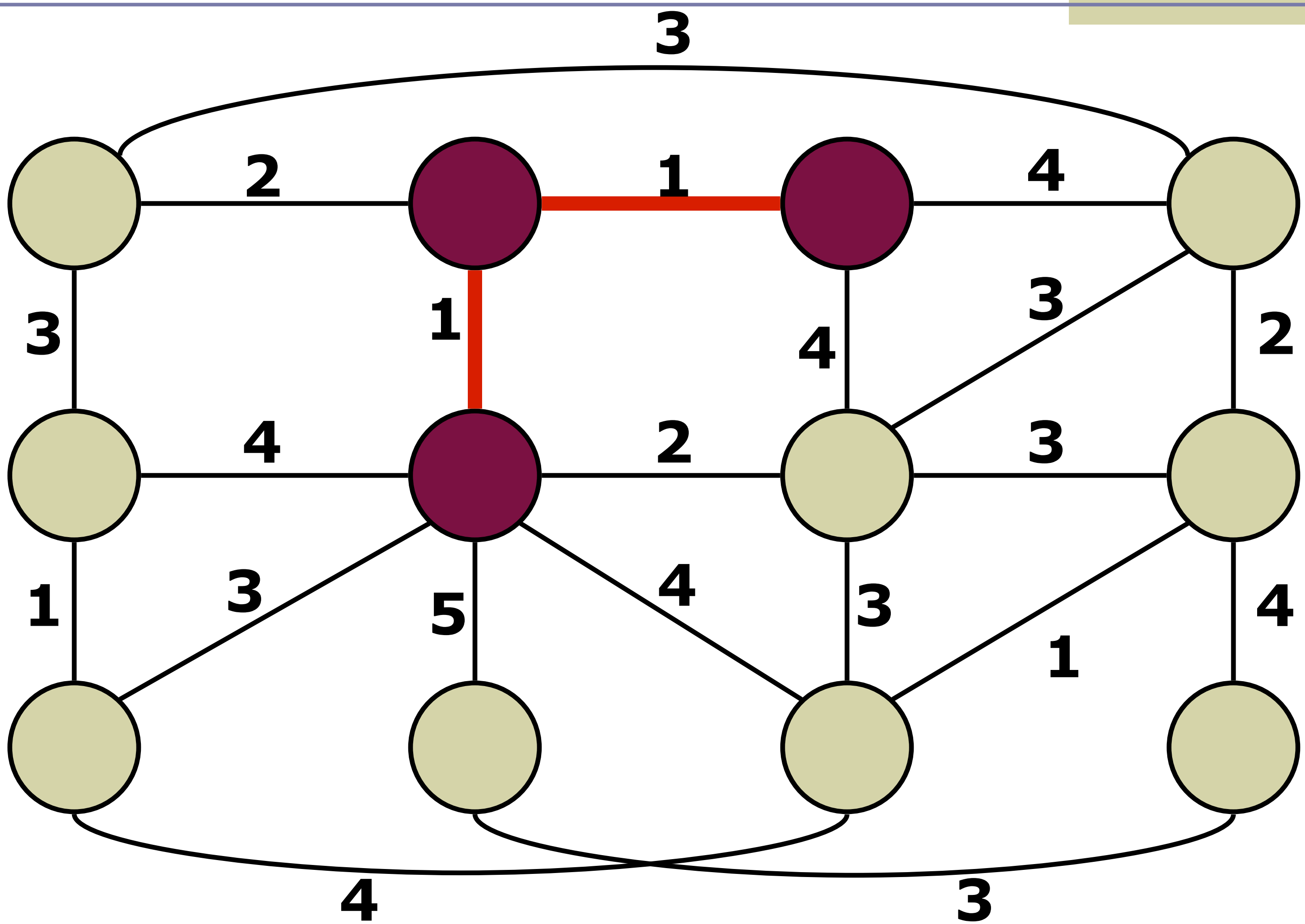
Prim's Algorithm



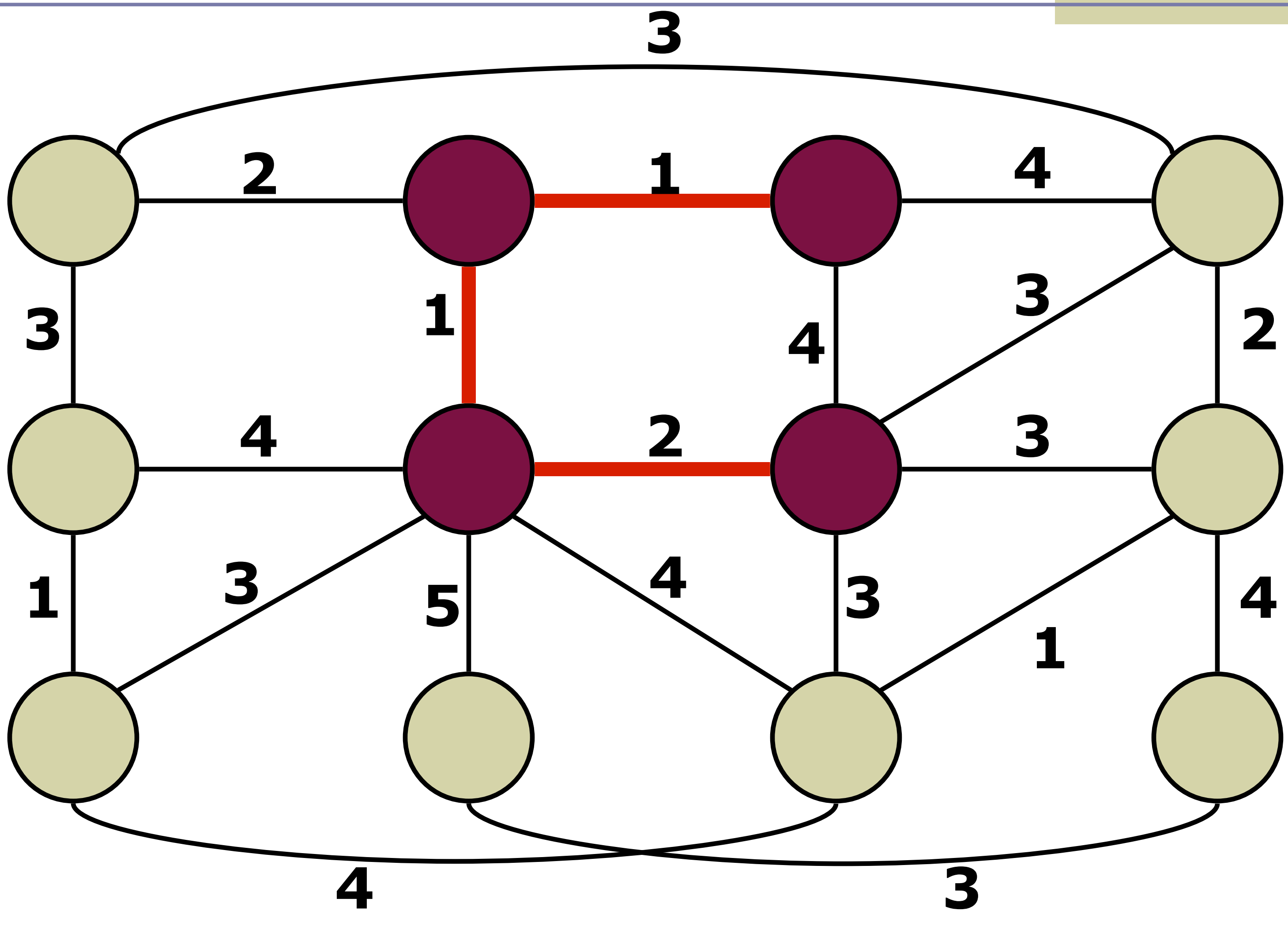
Prim's Algorithm



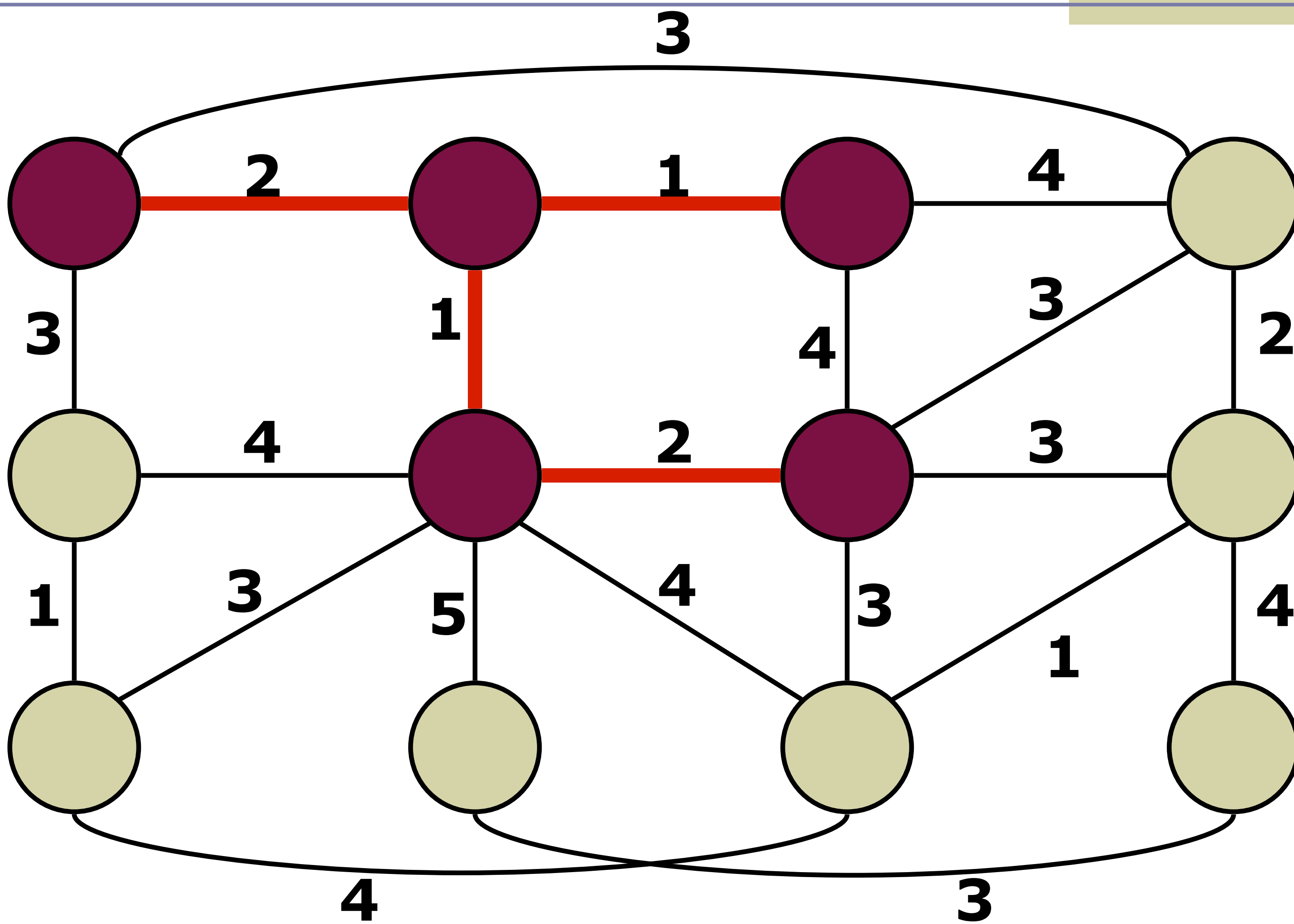
Prim's Algorithm



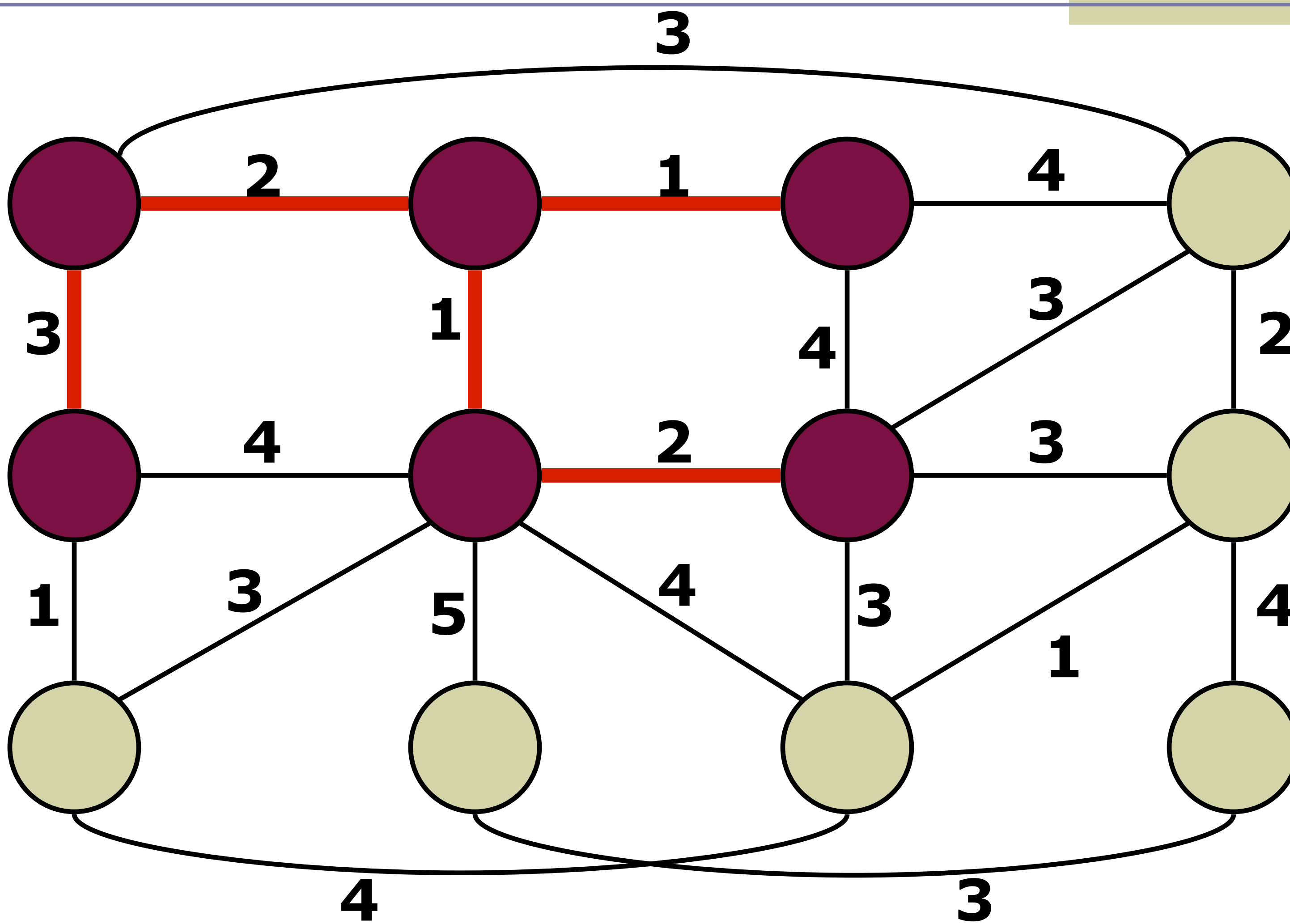
Prim's Algorithm



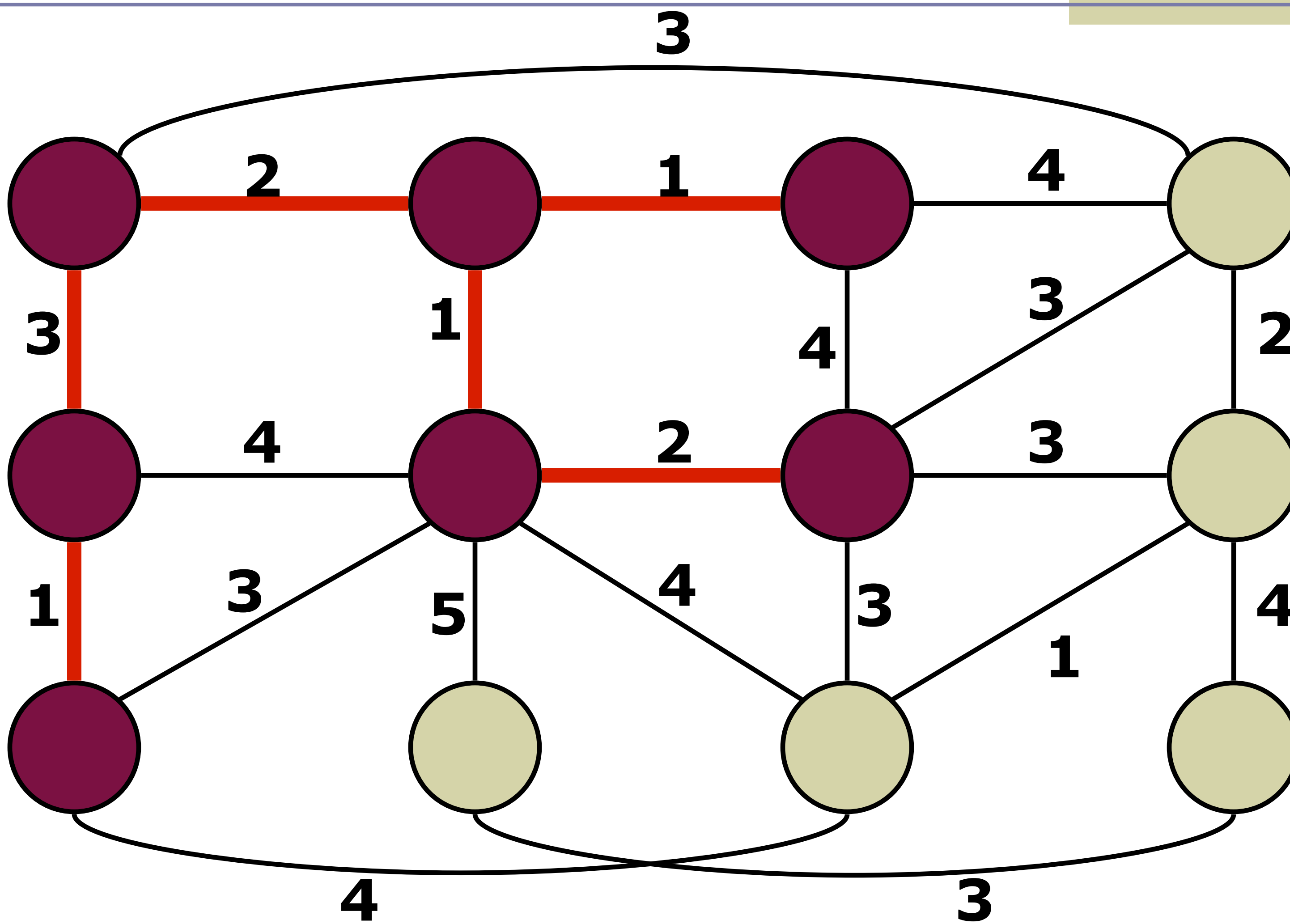
Prim's Algorithm



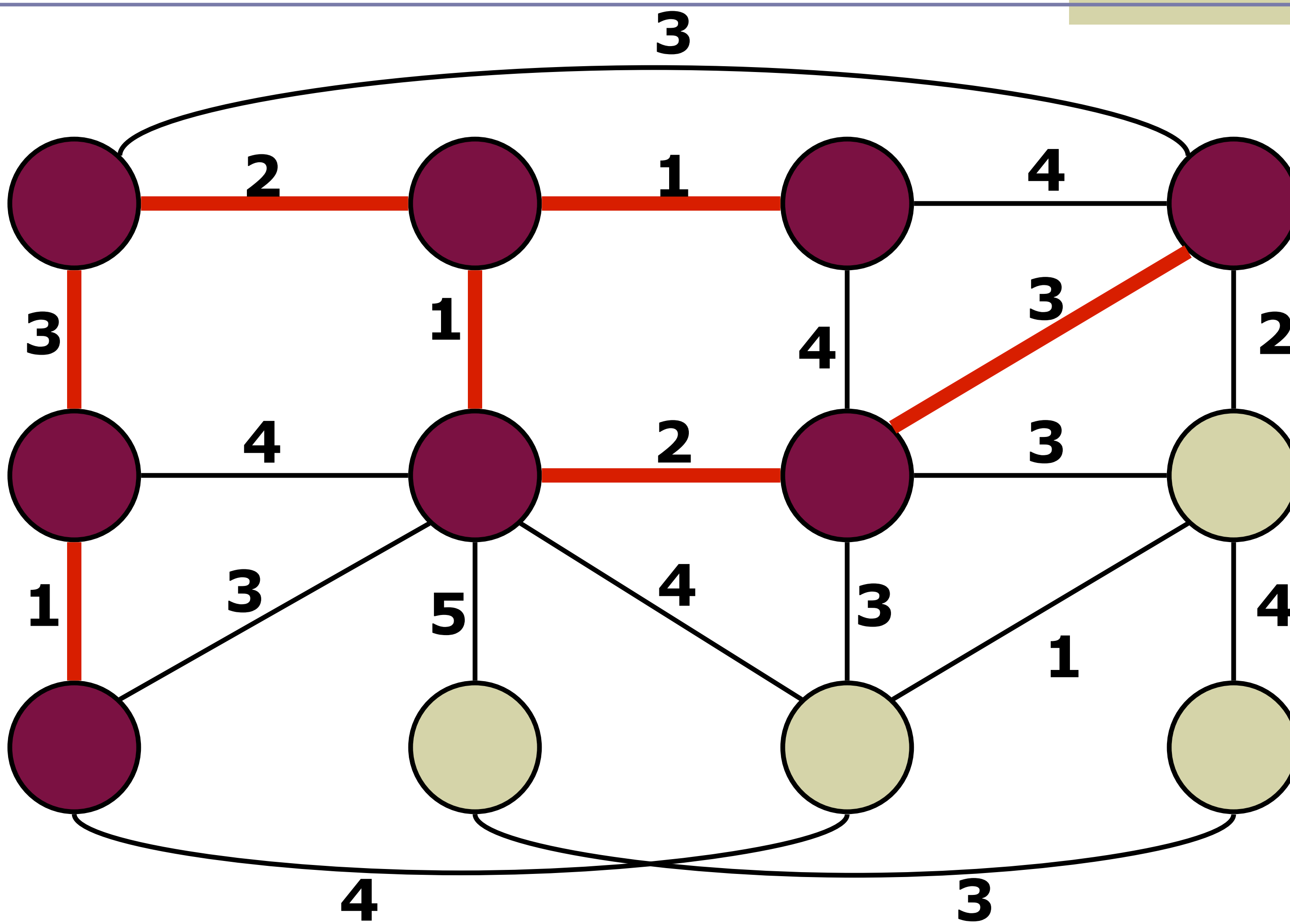
Prim's Algorithm



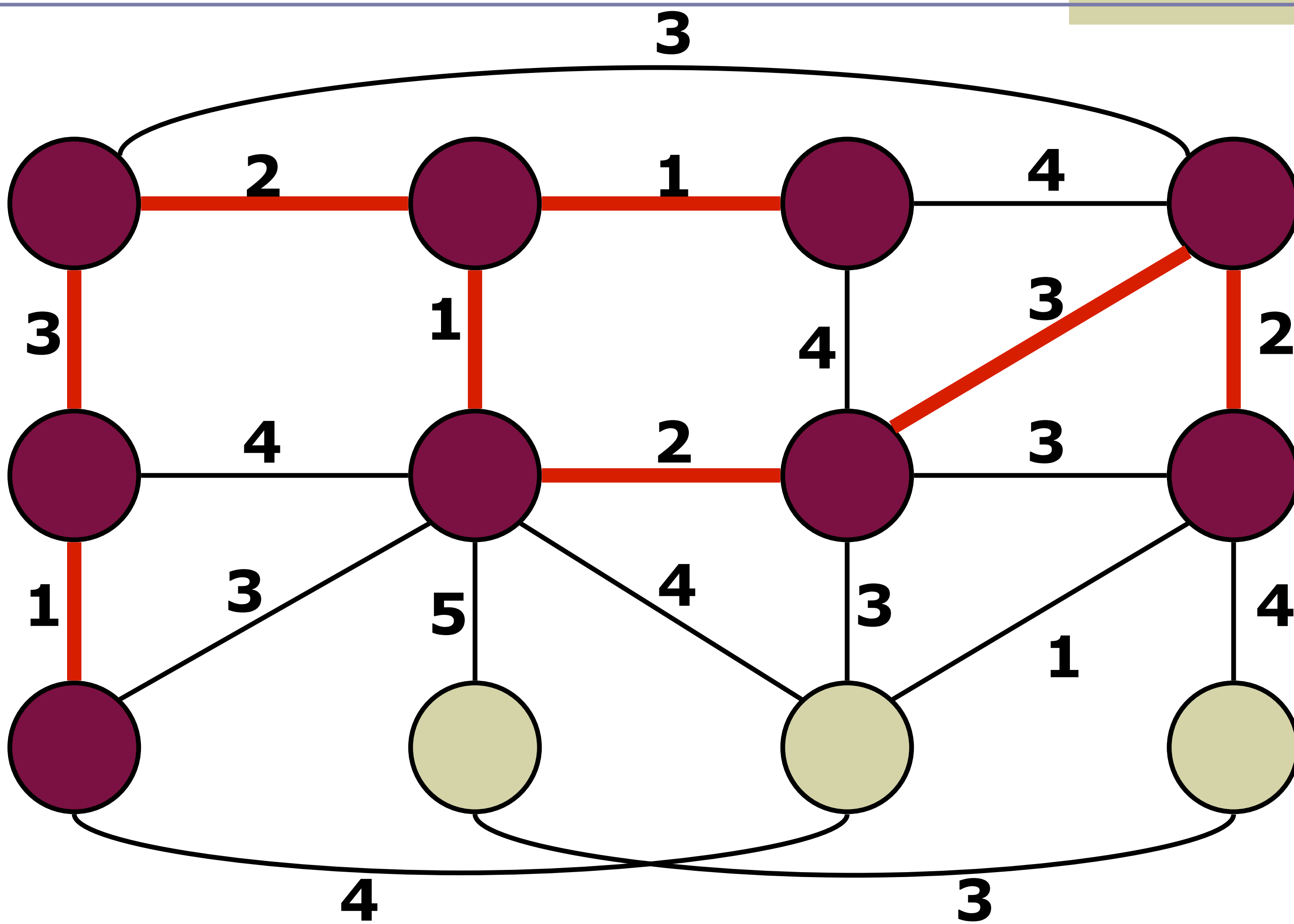
Prim's Algorithm



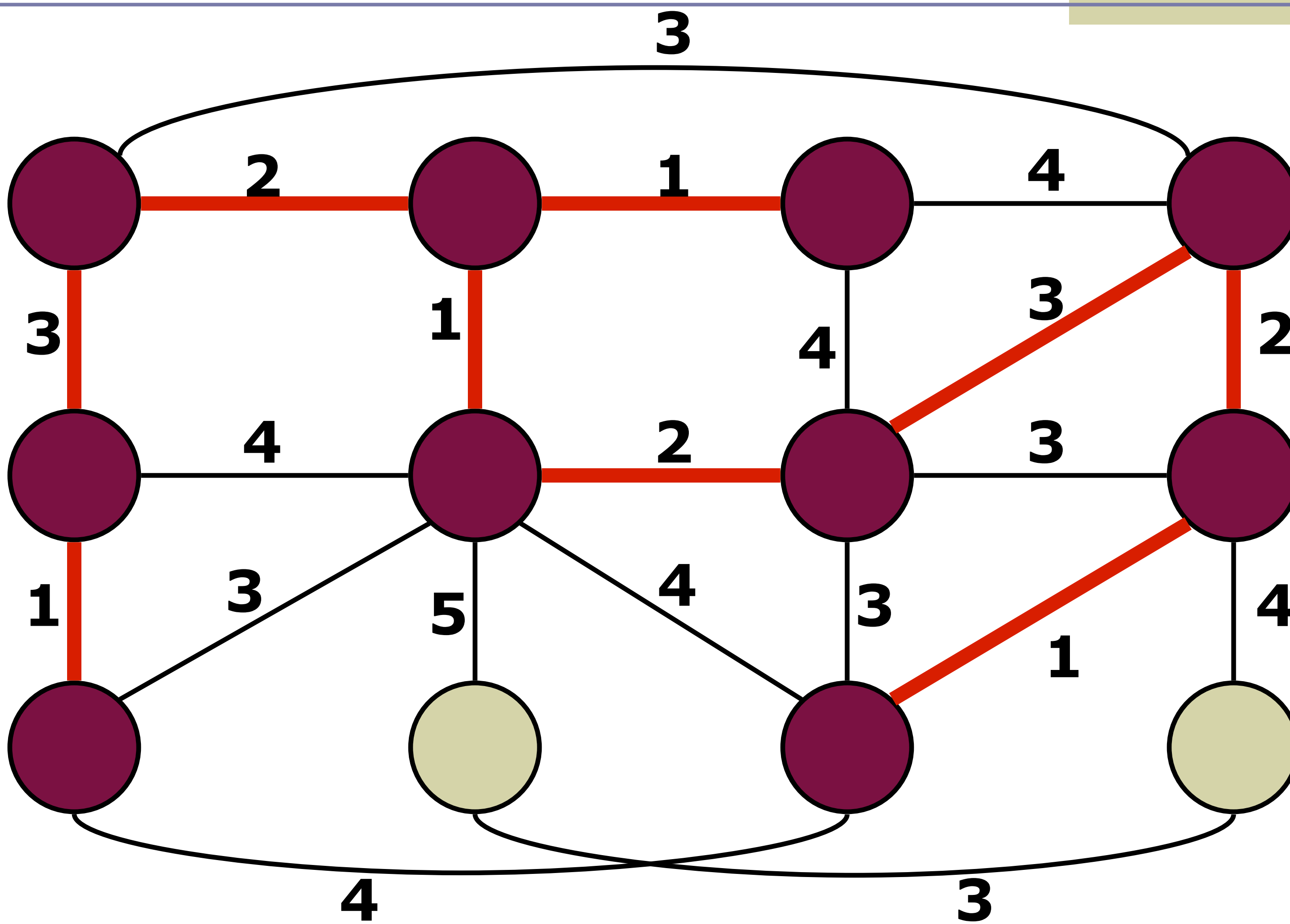
Prim's Algorithm



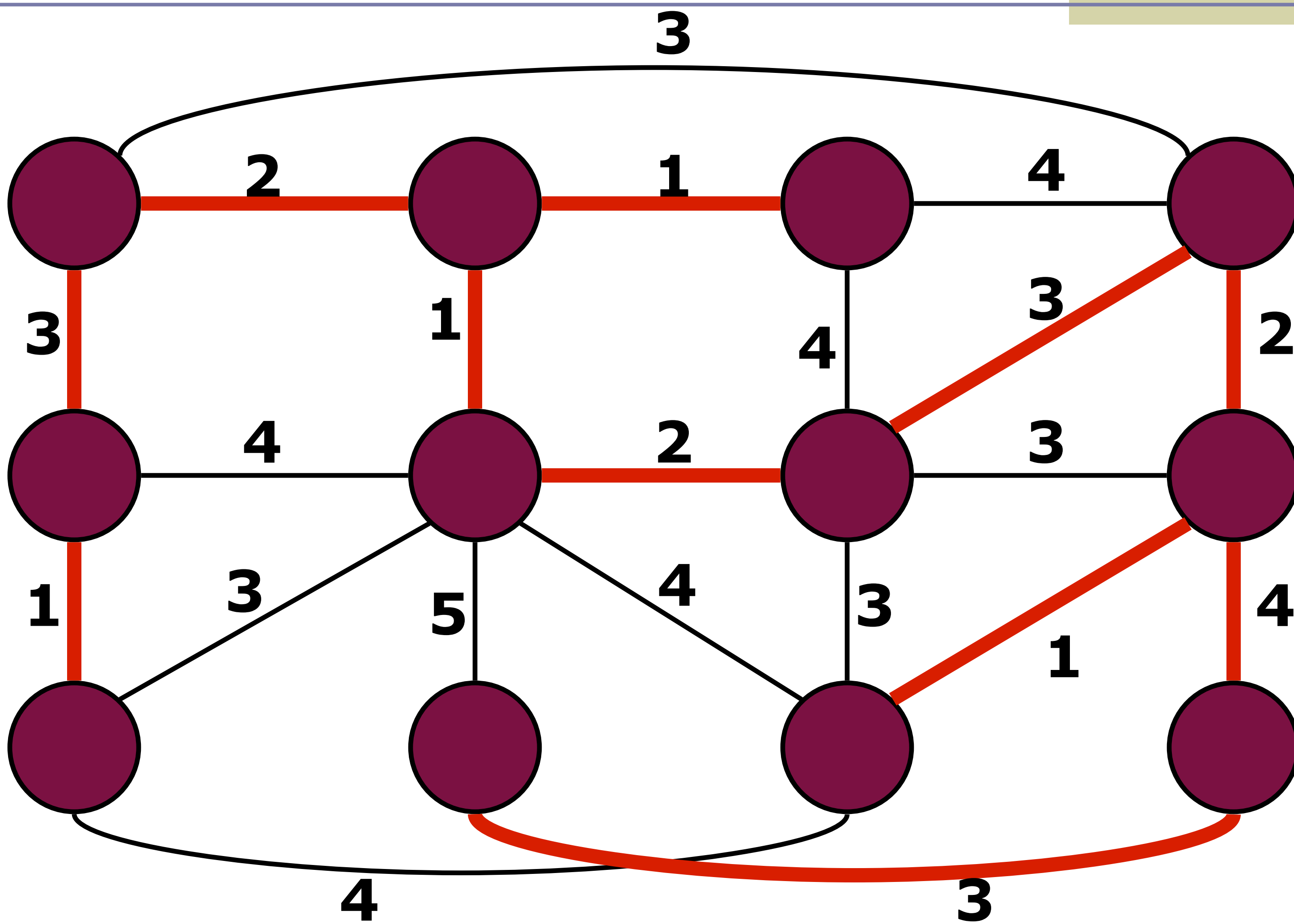
Prim's Algorithm



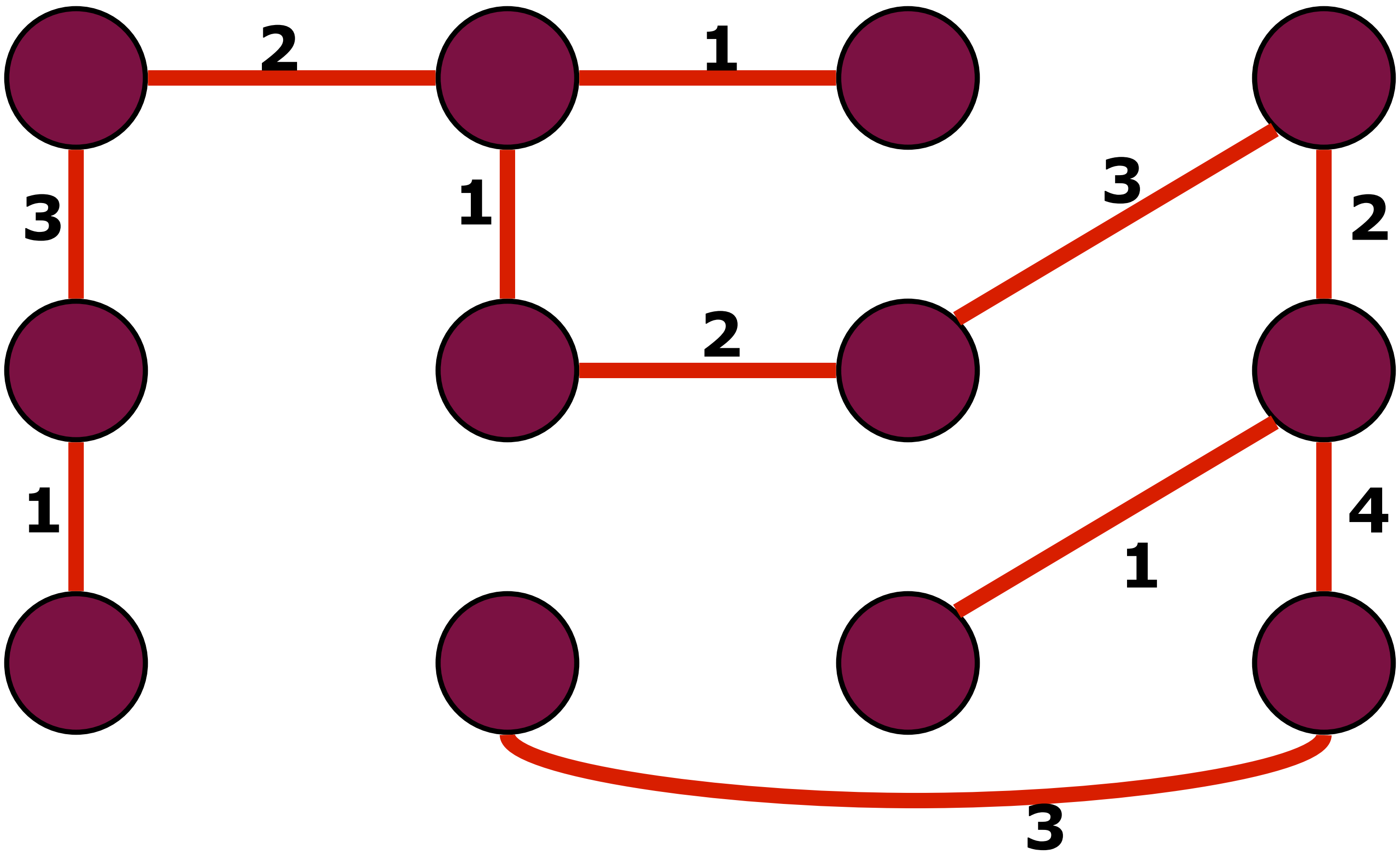
Prim's Algorithm



Prim's Algorithm



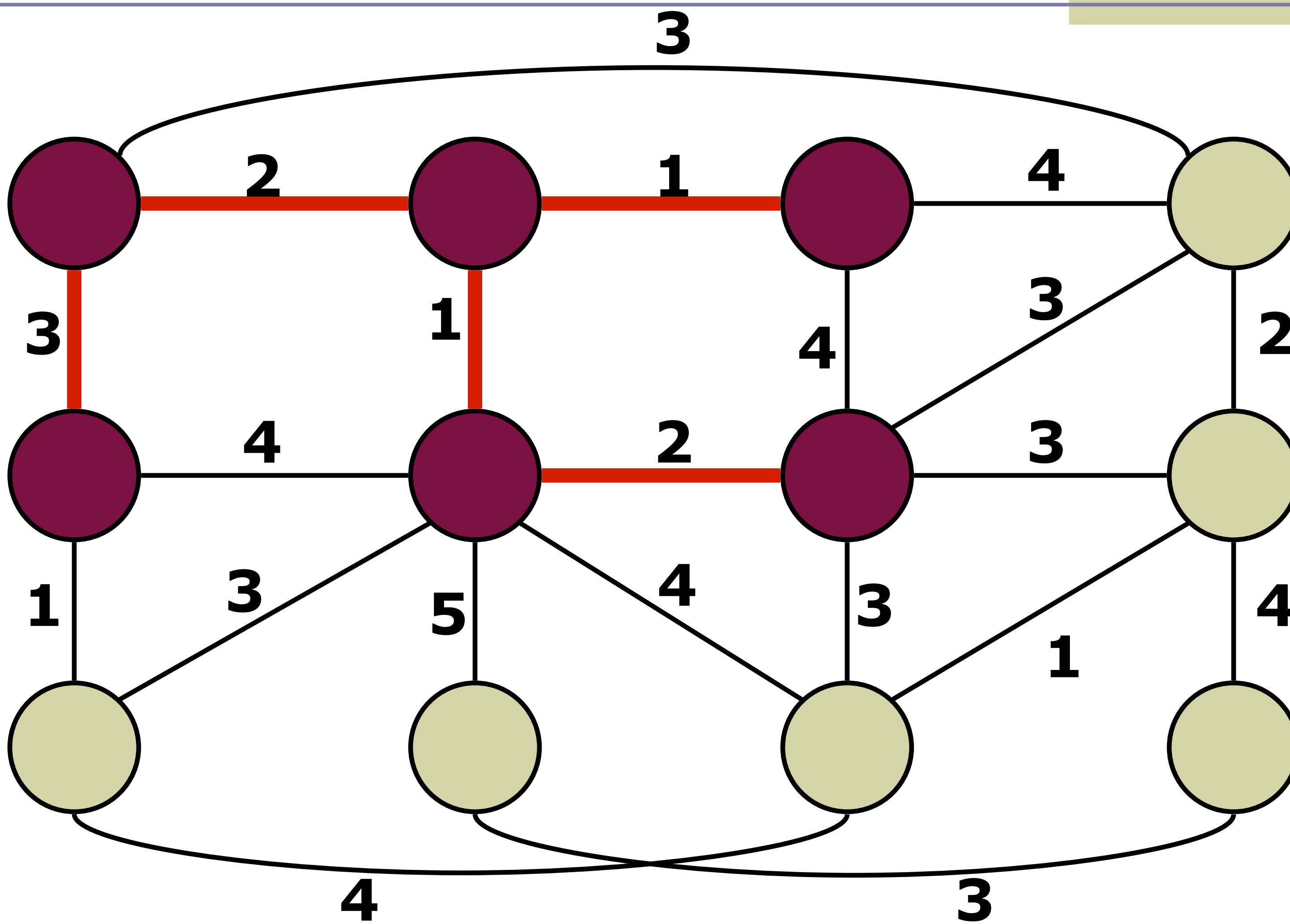
Prim’s Algorithm



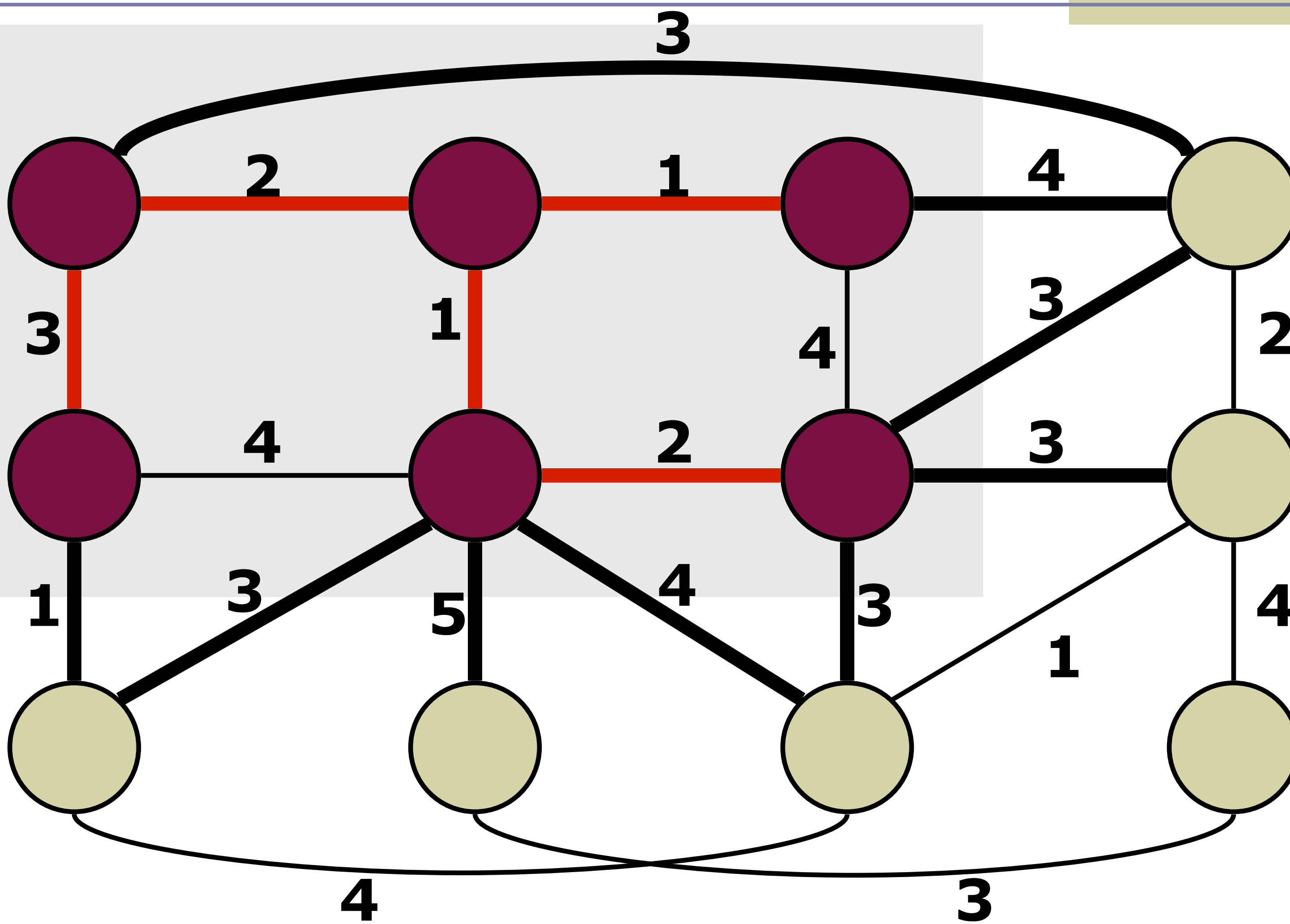
Prim's Greedy Algorithm

```
color all vertices yellow
color the root red
while there are yellow vertices
   pick an edge (u,v) such that
        u is red, v is yellow & cost(u,v) is min
   color v red
```

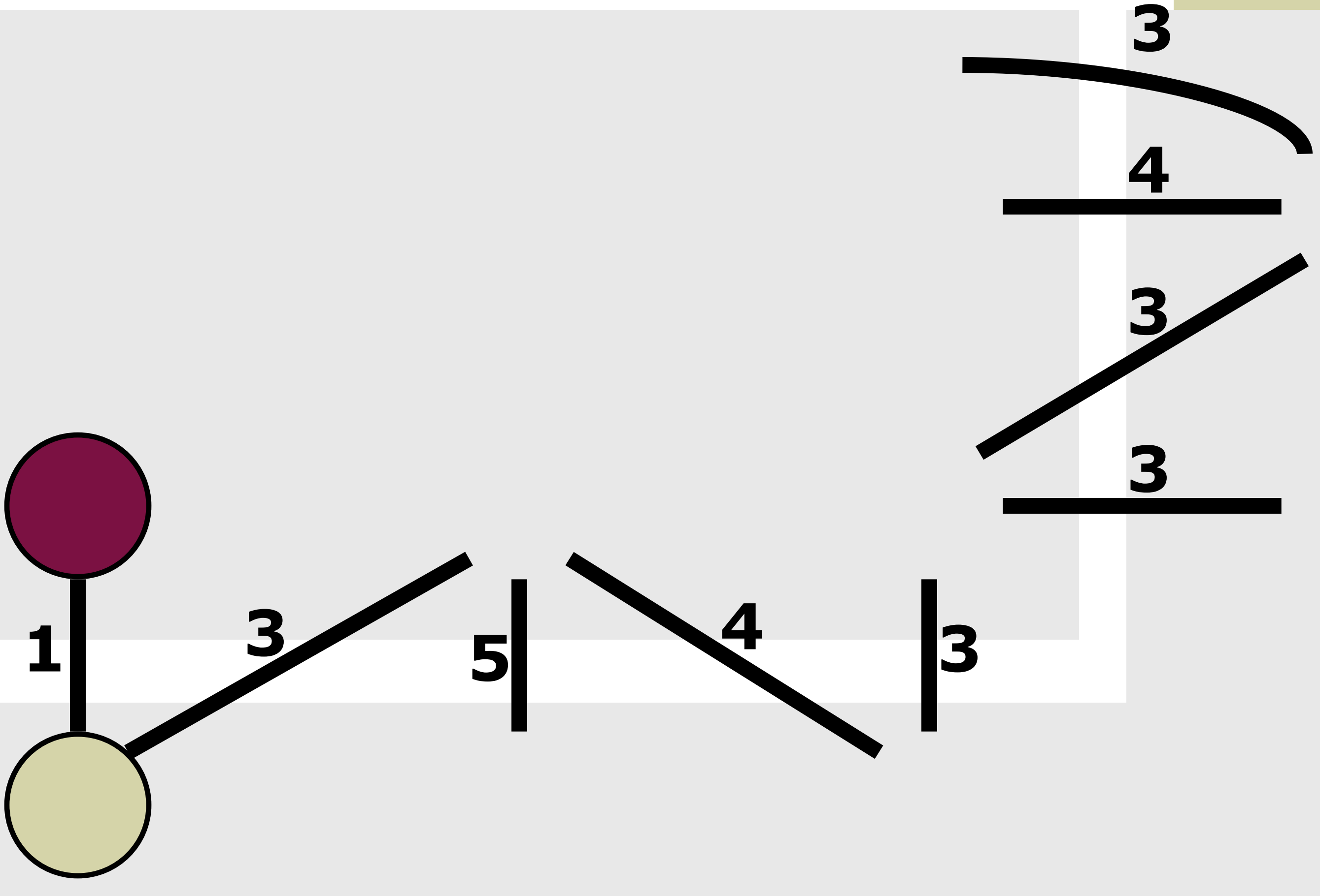
Why Greedy Works?



Why Greedy Works?



Why Greedy Works?



Prim's Algorithm

```
foreach vertex v
    v.key = ∞
root.key = 0
pq = new PriorityQueue(V)
while pq is not empty
    v = pq.deleteMin()
    foreach u in adj(v)
        if v is in pq and cost(v,u) < u.key
            pq.decreaseKey(u, cost(v,u))
```

Complexity: O((V+E)log V)

```
foreach vertex v
    v.key = ∞
root.key = 0
pq = new PriorityQueue(V)
while pq is not empty
    v = pq.deleteMin()
    foreach u in adj(v)
        if v is in pq and cost(v,u) < u.key
            pq.decreaseKey(u, cost(v,u))
```

Variations

- Does Prim and Kruskal works with negative weights?

- How about Maximum Spanning Tree?

Included for your fun reading

Sollin's Algorithm,
Yao's algorithm,
(General idea only)

Sollin's MST Algorithm

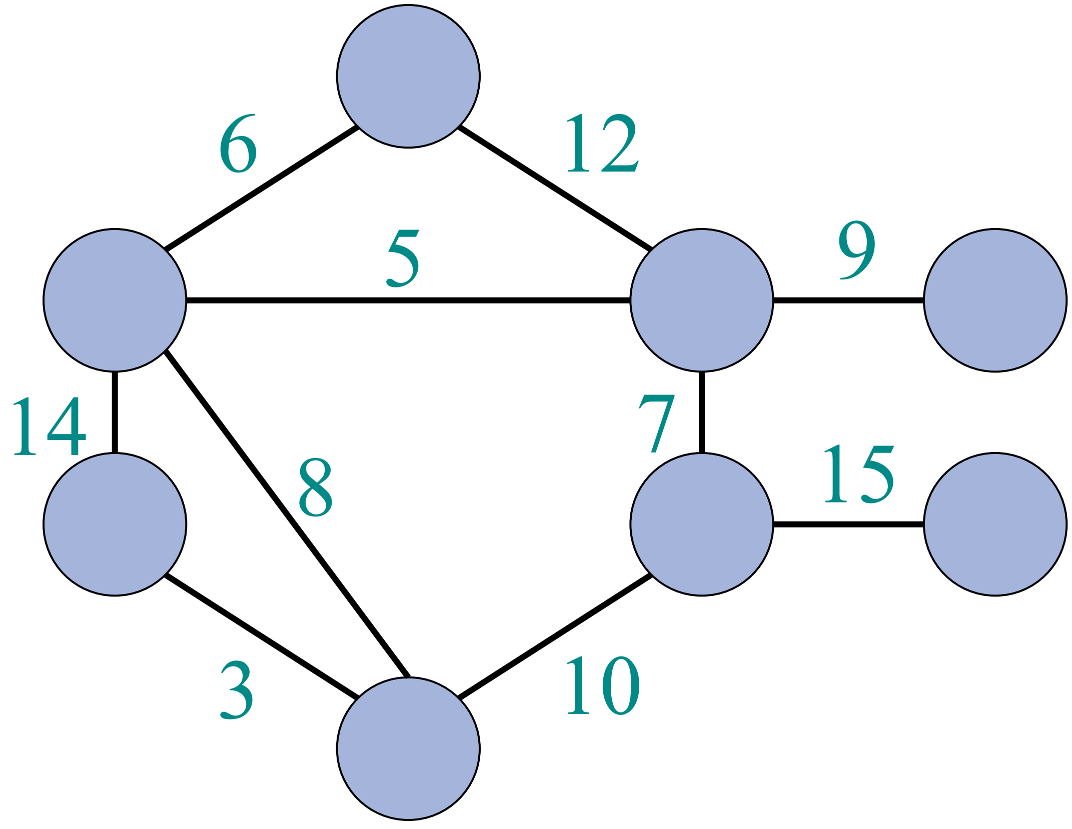
- G. Sollin, "Probleme de l'arbre minimum",(unpublished manuscript prepared for C. Berge Paris' Seminar), 1961.
- G. Sollin, "Problemes de recherche operationelle," Report C.41, Meeting of Technical Directors, S.E.G. Paris, (1962), pp. 15-23.

Idea:

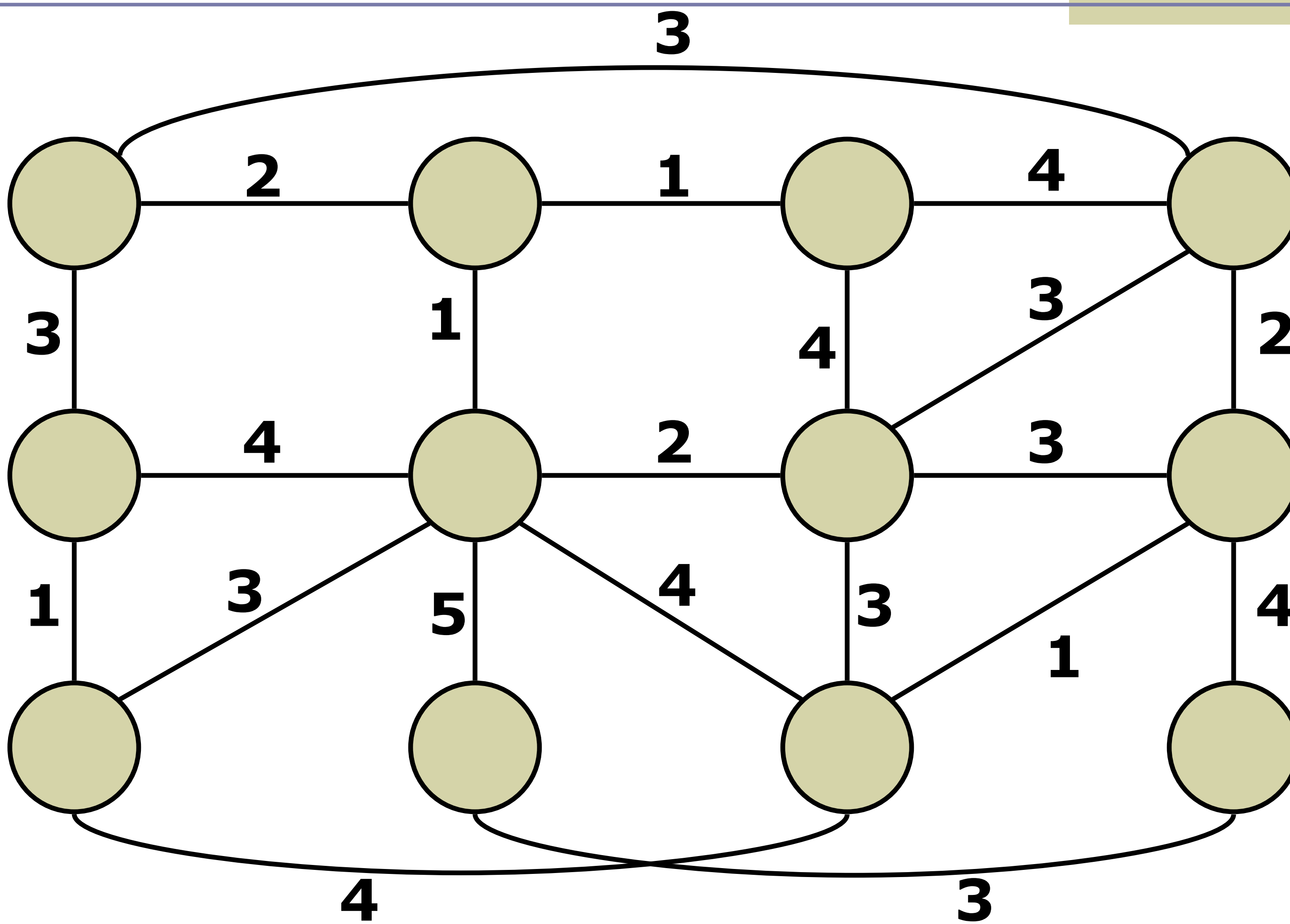
"Repeatedly,

add shortest edges to
each "component" in parallel; "

Sollin's MST Algorithm - DIY



Sollin's Algorithm – DIY #2



Sollin’s Algorithm

T ← empty tree;

foreach vertex v in G,

choose min edge e adjacent to v;
Add e to tree T

Collapse/Merge connected components formed to get reduced graph G’

Repeat process on reduced graph G’

Complexity: O((V+E)log V)

T ← empty tree;

foreach vertex v in G,

choose min edge e adjacent to v;
Add e to tree T

Collapse/Merge connected components formed to get reduced graph G′

Repeat process on reduced graph G′

Andy Yao's MST Algorithm

- A. C.-C. Yao, "An O(*e* log log *v*) algorithm for finding minimum spanning trees", Information Processing Letters, 4 (1975), pp. 21-23.

Key Ideas:

"Improve Sollin's algorithm,

Use smarter priority queues."

Improve from O(e log v) to O(e log log v)

The O(e log log v) paper…

AN O(|E| log log |V|) ALGORITHM FOR FINDING MINIMUM SPANNING TREES *

Andrew Chi-chih YAO
Department of Computer Science, University of Illinois, Urbana, Illinois 61801, USA

Received 30 December 1975, revised version received 9 June 1975

Minimum spanning tree, linear median finding algorithm

1. Introduction

Given a connected, undirected graph $G = (V, E)$ and a function c which assigns a cost $c(e)$ to every edge $e \in E$, it is desired to find a spanning tree T for G such that $\Sigma_{e \in T}\, c(e)$ is minimal. In this note we describe an algorithm which finds a minimum spanning tree (MST) in $O(|E| \log \log |V|)$ time. Previously the best MST algorithms known have running time $O(|E| \times \log |V|)$ for sparse graphs [1], and more recently Tarjan [2] has an algorithm that requires $O(|E| \times \sqrt{\log |V|})$ time.

Our algorithm is a modification of an algorithm by Sollin [3]. His method works by successively enlarging components of the MST. In the first stage the minimum-cost edge incident upon each node of G is found.

plying the linear median-finding algorithm [4]. Having accomplished this, we follow basically Sollin's algorithm as outlined above. Note that the number of operations needed in this phase is now reduced to

$$O\left(\frac{|E|}{k} \log |V|\right)$$

since only approximately $|E|/k$ edges have to be examined at each stage to find the minimum-cost edges incident with all the nodes. Therefore, the total number of operations required by our algorithm is

$$O\left(|E| \log k + \frac{|E|}{k} \log |V|\right),$$

which is $O(|E| \log \log |V|)$ if we choose k to be $\log |V|$.

Yao @UIUC (Oct-29, 2015)

https://cs.illinois.edu/news/alumnus-andrew-yao-sees-quantum-computing-next-great-science

Andy Yao @Tsinghua

Yao Class 姚班 @ 清华

C. L. Liu (刘炯朗) @清华

歷任校長

劉炯朗

1998~2002

劉炯朗 先生，廣東番禺人，民國23年出生，幼年時期在澳門就學，後因為父親在台灣擔任軍職，遂前來台灣就學，並考入當時的台南工學院電機系（成功大學）就讀，獲工學士。大學畢業後，劉校長從軍擔任陸軍少尉預官。退伍後報考清華大學原子科學研究所，獲得正取，但因同時取得美國麻省理工學院獎學金，所以便隻身負笈留美，順利取得麻省理工學院電腦碩士、博士。之後曾經執教麻省理工學院、伊利諾大學、清華大學等，並擔任伊利諾大學香檳校區助理副校長一職。1998年，經過本校及教育部甄選後，出任本校第二任遴選的校長一職。