极客时间算法训练营
第十五课
图论算法

李煜东

《算法竞赛进阶指南》作者

目录

1. 复习：图的基本概念和算法
2. 最短路
3. 最小生成树

图的基本概念与算法

复习：图

图可以表示为一个点集和一个边集：Graph(V, E)

V – vertex：点

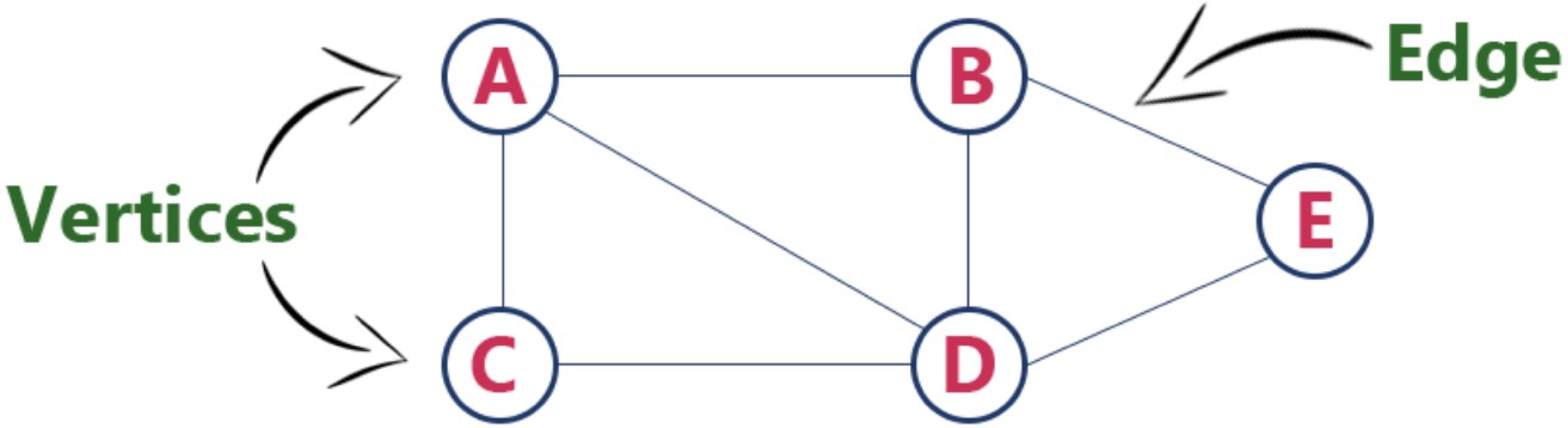
点的度数 (degree)

入度和出度 – 一个点相连的入边/出边数量

E – edge：边

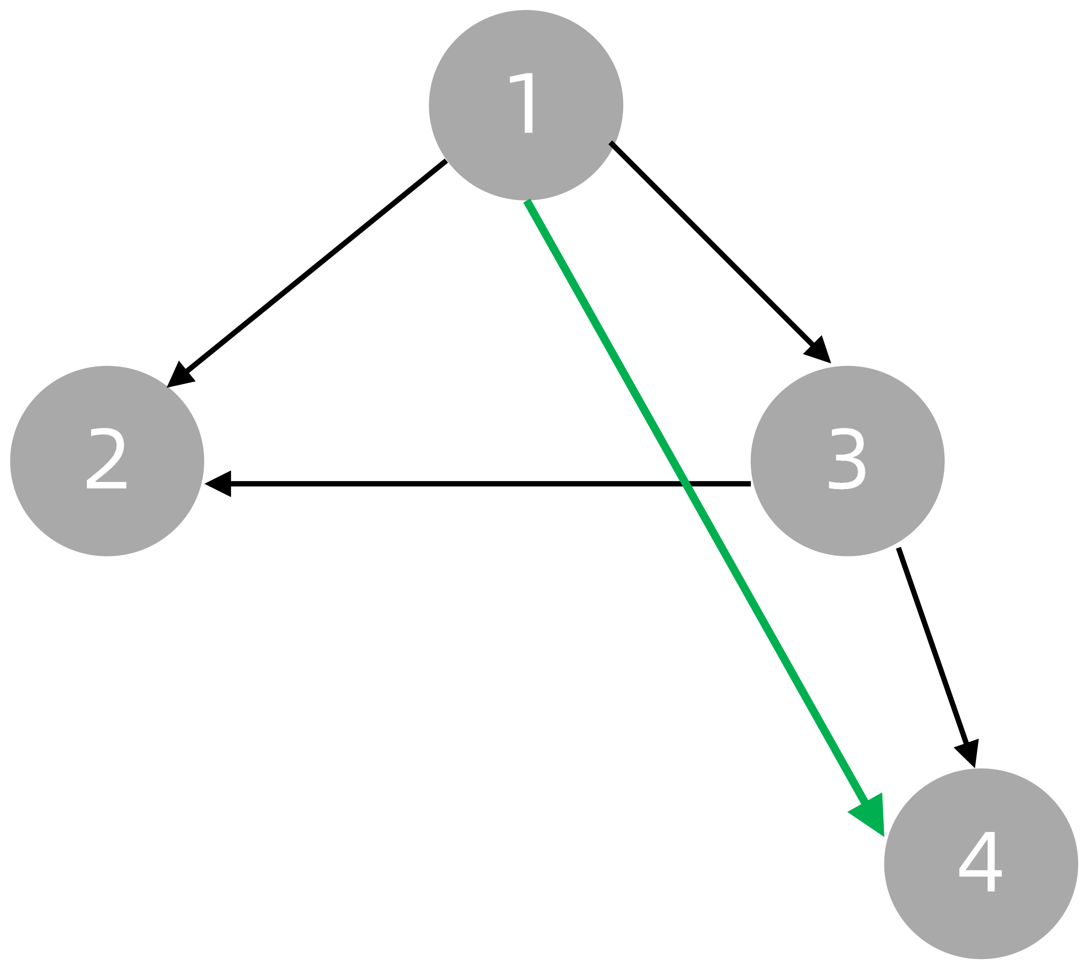
有向和无向

带权图：边的权值（长度）



“连通”、“环”等概念

复习：图的存储



	1	2	3	4
1	0	1	1	1
2	0	0	0	0
3	0	1	0	1
4	0	0	0	0

邻接矩阵

1	2	3	4
2			
3	2	4	
4			

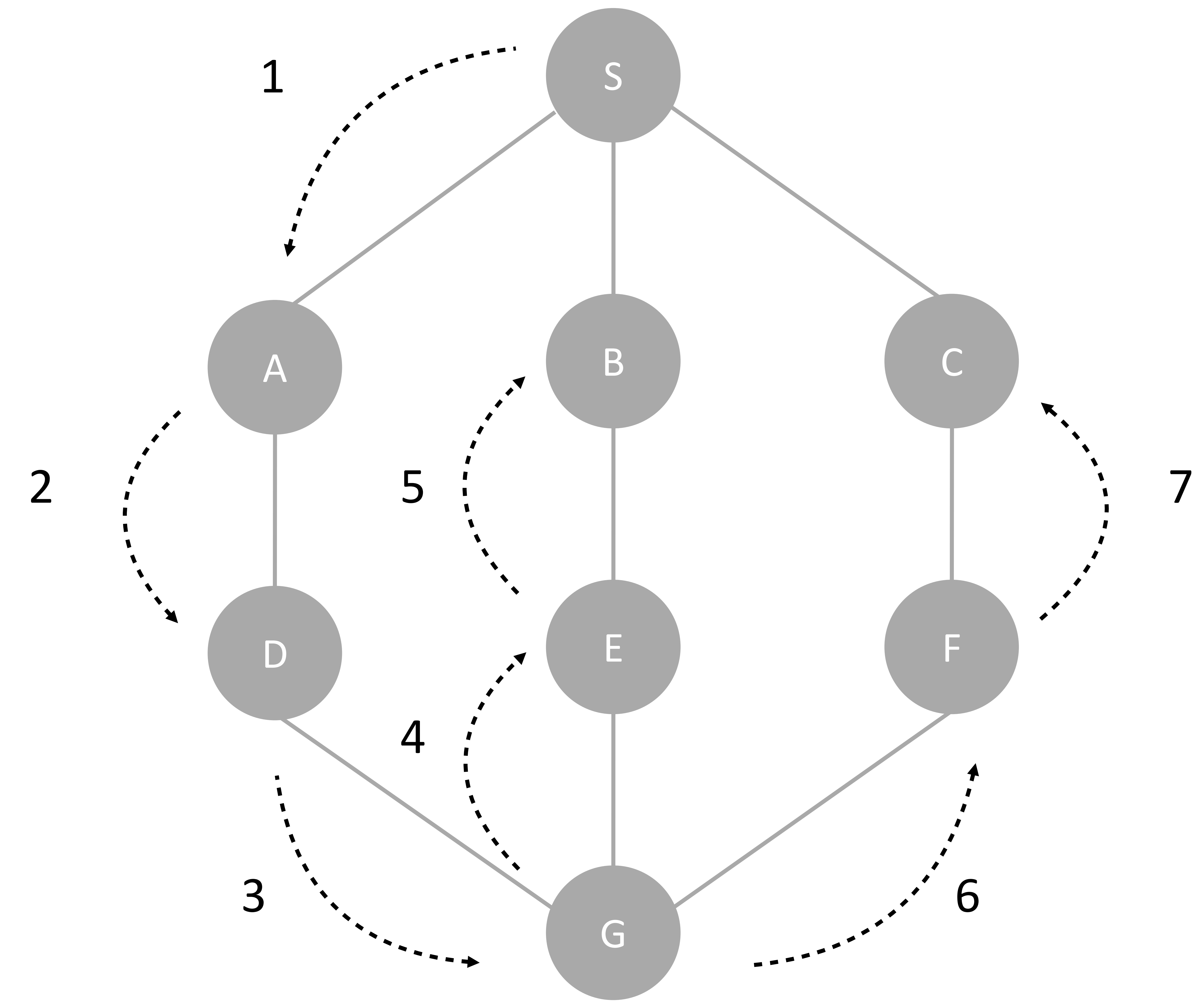
出边数组

head

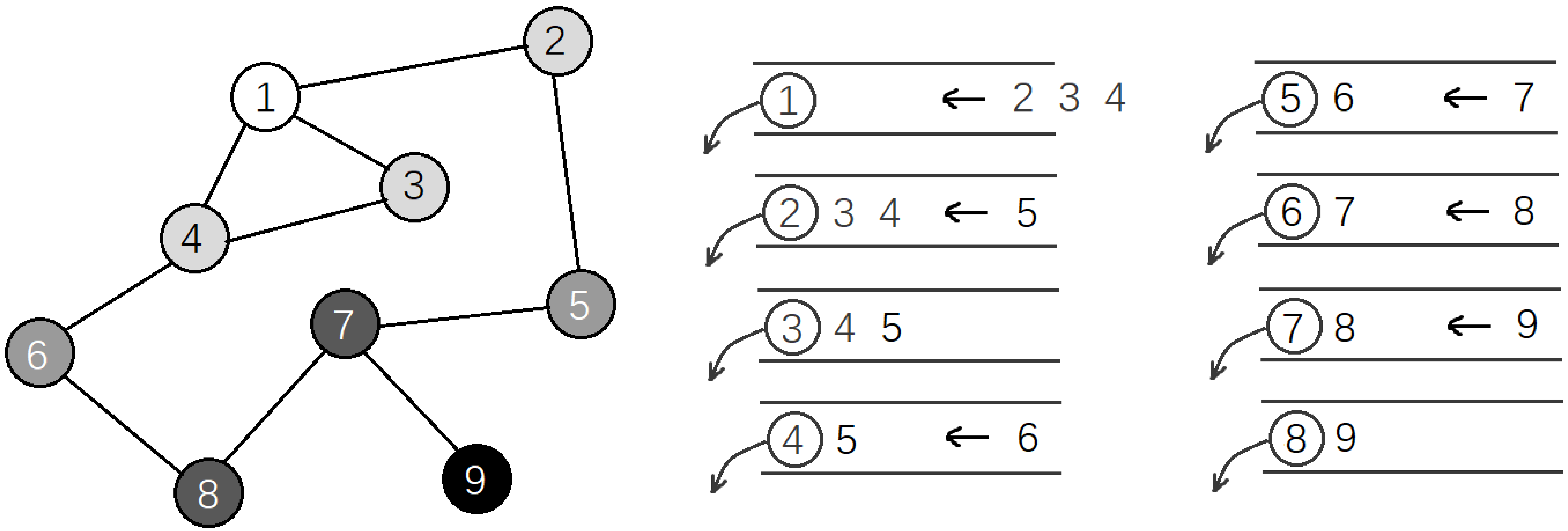
1 → 4 → 3 → 2 → null
2 → null
3 → 4 → 2 → null
4 → null

邻接表

复习：图的深度优先遍历



复习：图的广度优先遍历



最短路

单源最短路径问题

单源最短路径问题（Single Source Shortest Path，SSSP问题）是说：

- 给定一张有向图 $G = (V, E)$，V 是点集，E 是边集，$|V| = n$，$|E| = m$
- 节点以 $[1, n]$ 之间的连续整数编号
- (x, y, z) 描述一条从 x 出发，到达 y，长度为 z 的有向边
- 设1号点为起点

求长度为 n 的数组 $dist$，其中 $dist[i]$ 表示从起点1到节点 i 的最短路径的长度

Bellman-Ford 算法

Bellman-Ford 算法是基于动态规划和迭代思想的

1. 扫描所有边 (x, y, z)，若 $dist[y] > dist[x] + z$，则用 $dist[x] + z$ 更新 $dist[y]$。
2. 重复上述步骤，直到没有更新操作发生

若问题有解（图中没有负环），Bellman-Ford“扫描所有边并更新”的过程至多执行 n – 1 轮

原因：一条最短路至多包含 n – 1 条边

时间复杂度 O(nm)

可以把每一轮看作DP的一个阶段

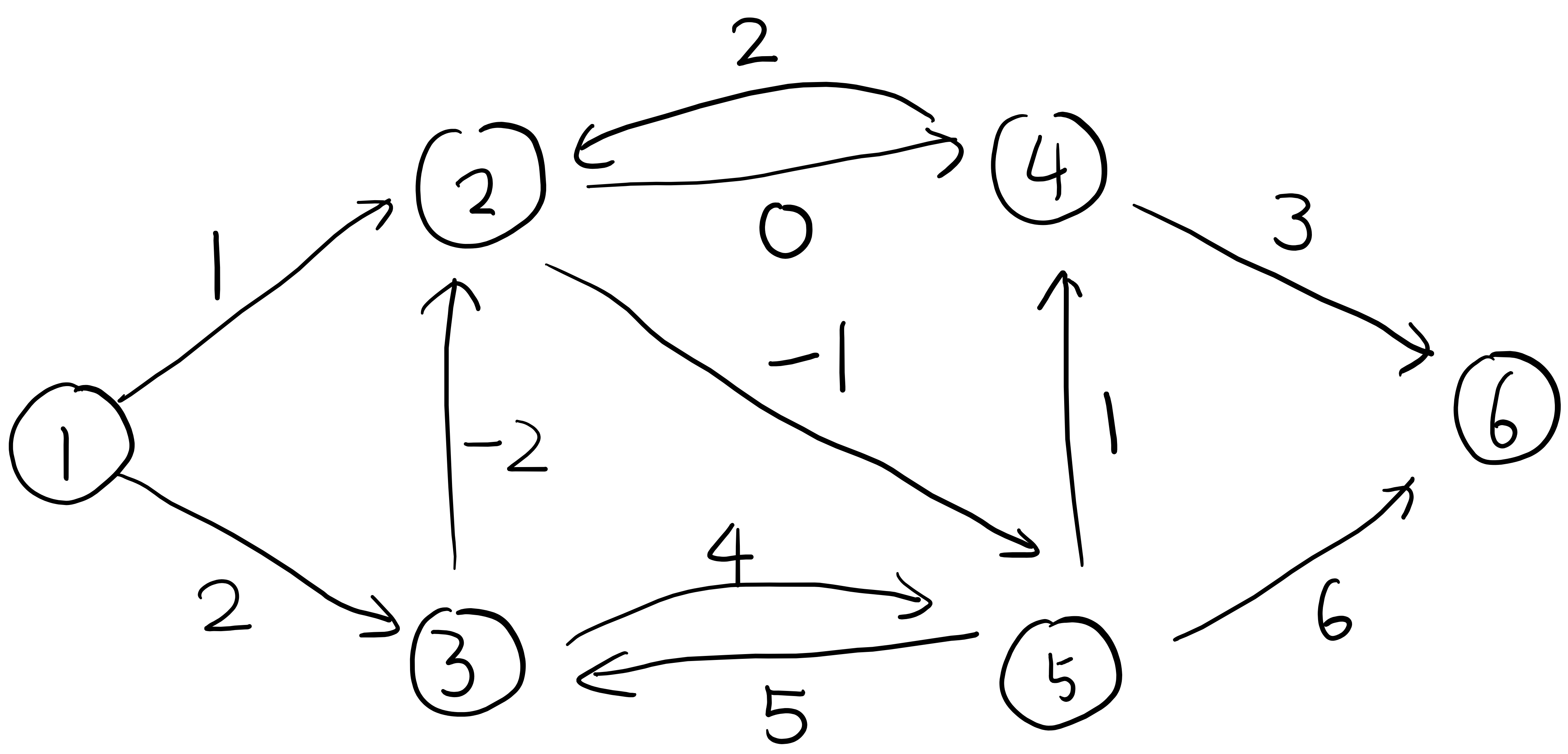
第 i 轮至少已经求出了包含边数不超过 i 的最短路

Bellman-Ford 算法演示

只需要一轮的例子？

需要 n - 1 轮的例子？

一般图上的演示



Dijkstra 算法

Dijkstra 算法是基于贪心思想的，只适用于所有边的长度都是非负数的图

1. 初始化 $dist[1] = 0$，其余节点的 $dist$ 值为正无穷大。
2. 找出一个未被标记的、$dist[x]$ 最小的节点 x，然后标记节点 x。
3. 扫描节点 x 的所有出边 (x, y, z)，若 $dist[y] > dist[x] + z$，则使用 $dist[x] + z$ 更新 $dist[y]$。
4. 重复上述2~3两个步骤，直到所有节点都被标记。

贪心思路：在非负权图上，全局最小的 dist 值不可能再被其他节点更新

因此可以不断取 dist 最小的点（每个点只被取一次），更新其他点

用二叉堆维护最小 dist 值可以做到 O(m*log(n)) 的时间复杂度

Dijkstra 算法

Dijkstra 算法的另一种理解方法：优先队列 BFS

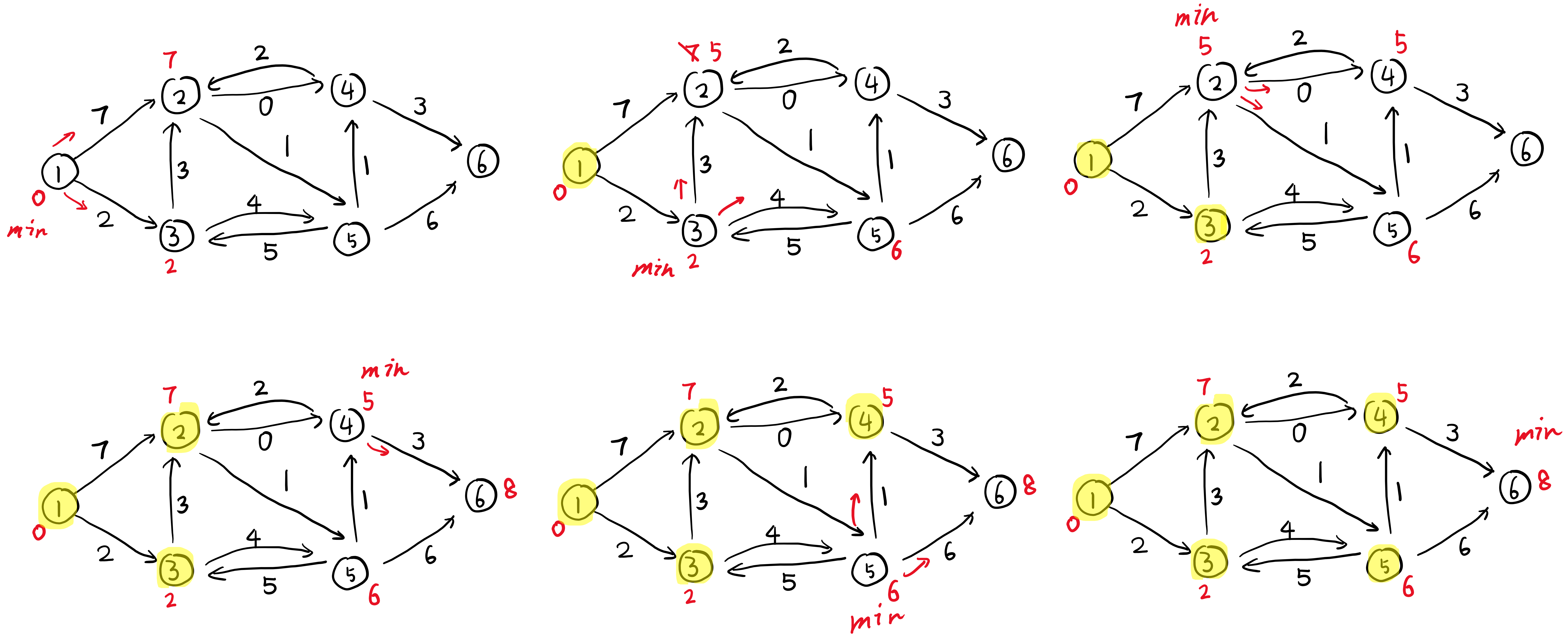
如果边权都是1，求最短路，可以用 BFS

BFS 为什么每个点只需要访问一次？因为队列中点对应的路径长度满足“单调性”和“两段性”

如果边权是任意非负数，怎么保证每个点依然只需要扩展一次?

优先队列 BFS —— 先扩展最短的

Dijkstra 算法演示



实战

网络延迟时间

https://leetcode-cn.com/problems/network-delay-time/

Dijkstra Benchmark

https://www.acwing.com/problem/content/852/

Floyd 算法

Floyd 算法可以在 O(n^3) 时间内求出图中每一对点之间的最短路径

本质上是动态规划算法

$dp[k,i,j]$ 表示经过编号不超过 k 的点为中继，从 i 到 j 的最短路

决策：是否使用 k 这个中继点

$$dp[k,i,j] = \min(dp[k-1,i,j], \quad dp[k-1,i,k] + dp[k-1,k,j])$$

可以省掉第一维，变成

$$d[i,j] = \min(d[i,j], \quad d[i,k] + d[k,j])$$

初态：d 为邻接矩阵（原始图中的边）

与 Bellman-Ford，Dijkstra 的比较： O(n^3) vs O(n^2m) vs O(nmlogn)

实战

阈值距离内邻居最少的城市

https://leetcode-cn.com/problems/find-the-city-with-the-smallest-number-of-neighbors-at-a-threshold-distance/

最小生成树

最小生成树问题

给定一张边带权的无向图 $G=(V,E)$，$n=|V|$，$m=|E|$。

由 V 中全部 n 个顶点和 E 中 $n-1$ 条边构成的无向连通子图被称为 G 的一棵生成树。

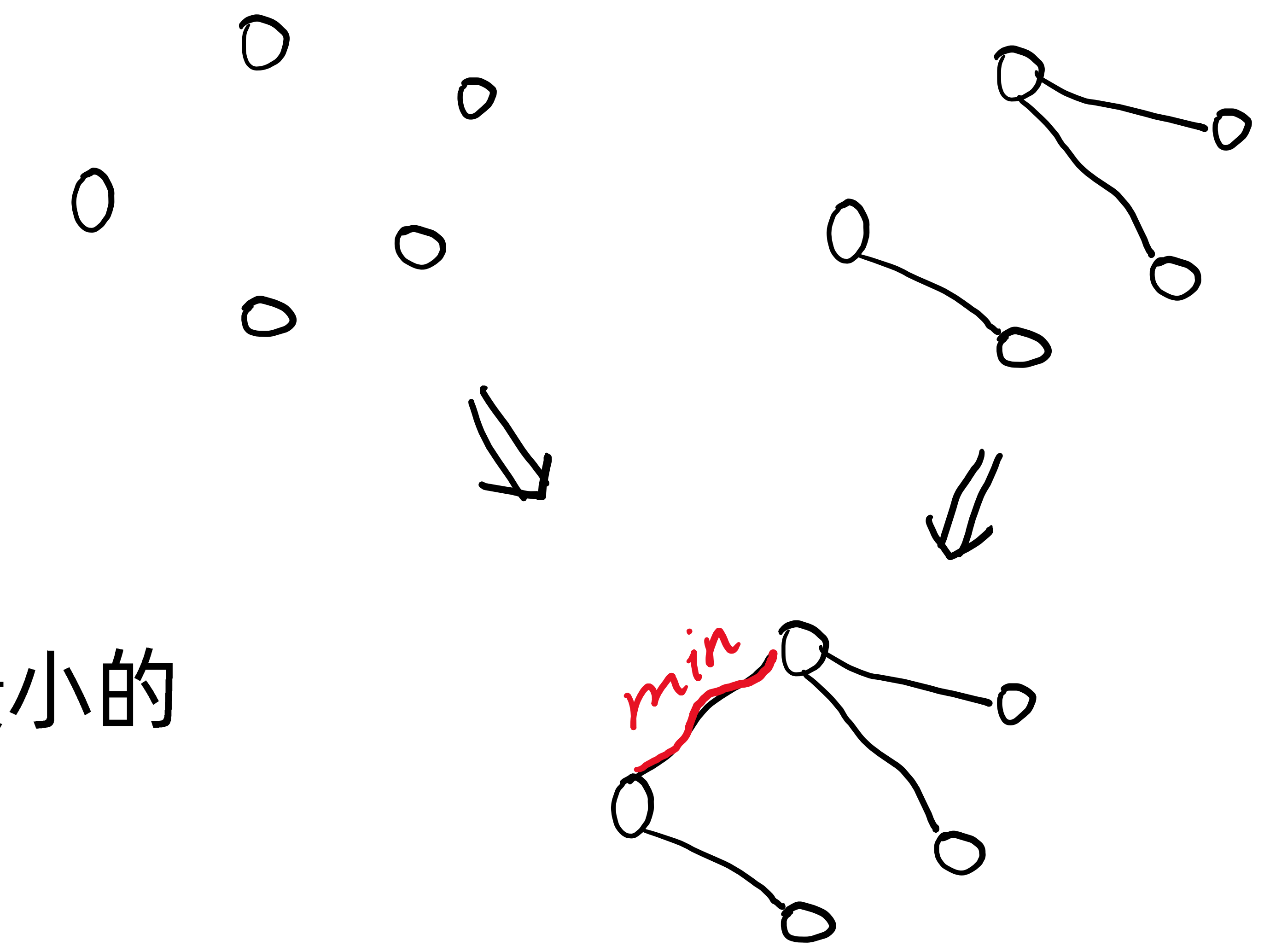
边的权值之和最小的生成树被称为无向图 G 的最小生成树（Minimum Spanning Tree, MST）。

性质：

任意一棵最小生成树一定包含无向图中权值最小的边

推论：

把任何一个生成森林扩展为生成树，一定使用了图中剩余边中权值最小的



证明方法：树上加一条边形成环 + 反证法

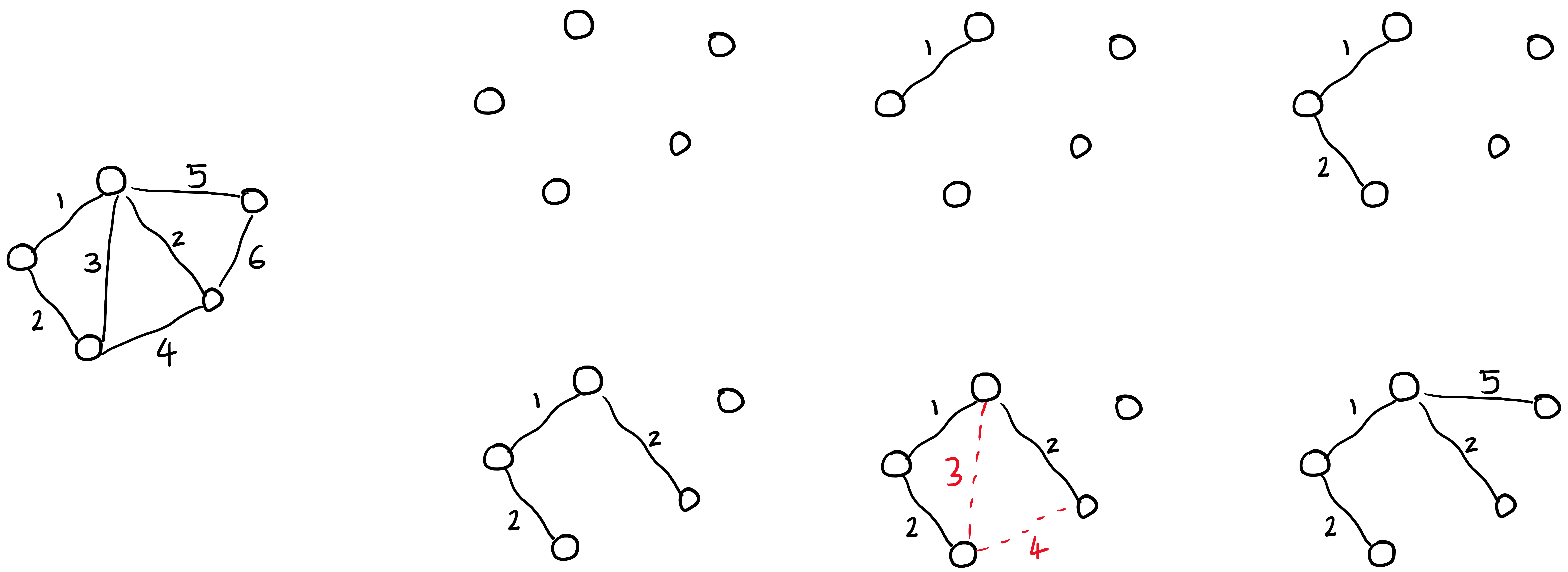
Kruskal 算法

Kruskal 算法总是使用并查集维护无向图的最小生成森林

1. 建立并查集，每个点各自构成一个集合。
2. 把所有边按照权值从小到大排序，依次扫描每条边 (x, y, z)。
3. 若 x, y 属于同一集合（连通），则忽略这条边，继续扫描下一条。
4. 否则，合并 x, y 所在的集合，并把 z 累加到答案中。
5. 所有边扫描完成后，第 4 步中处理过的边就构成最小生成树。

时间复杂度为 $O(m \log m)$。

Kruskal 算法演示



实战

连接所有点的最小费用

https://leetcode-cn.com/problems/min-cost-to-connect-all-points/

经典的“曼哈顿距离最小生成树”

- Kruskal 算法 —— $O(n^2 logn)$
- ~~Prim 算法 —— $O(n^2)$~~
- ~~树状数组优化建图 + Kruskal —— $O(nlogn)$~~

THANKS

极客时间 | 训练营