

EDA 软件设计 I

Lecture 14

Shortest Path Problem

最短路径问题（一类问题及算法）

Shortest Path Problem in Real World



GPS导航系统

应用场景：驾驶者或行人使用导航系统**规划从起点到终点的最短路径**

挑战：需要**处理动态交通信息**（如交通堵塞、事故等）来实时调整最短路径。此外，算法需要在大规模的城市交通网络中高效执行



互联网路由

应用场景：在互联网上，数据包需从源服务器传输到目标服务器。为了提高效率，**路由器需要选择经过网络设备的最短路径，以保证数据传输的低延迟和高可靠性**

挑战：由于互联网是动态的，网络状态经常发生变化，算法需要**实时应对网络故障、链路拥塞**等问题



城市公共交通调度

应用场景：城市中的公交系统或地铁网络希望为乘客**提供最快的出行路线**，尤其是在大城市的复杂交通网络中

挑战：**动态交通状况、公交/地铁时刻表以及换乘等因素增加了问题的复杂性**，需要考虑动态规划和实时优化

Shortest Path Problem in Real World



物流和配送系统优化

应用场景：物流公司希望通过**优化配送路径**来降低运输成本，并缩短送货时间。

挑战：还需要考虑**多重限制条件**（如**时间窗口**、**车辆容量**、**路况变化**等），这使得问题不仅仅是最短路径，而可能是多约束的路径规划问题（如车辆路径问题，VRP）



生物信息学中的序列对比

应用场景：在生物信息学中，最短路径算法用于**DNA或蛋白质序列的比对**，以寻找最短的编辑路径（如插入、删除、替换）

挑战：需要处理大规模序列数据，同时**确保算法在复杂的序列变异中有效**



社会网络中的信息传播

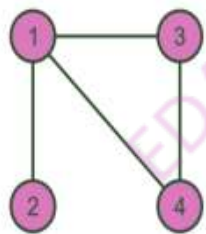
应用场景：在社交网络中，信息或病毒的传播路径是一个经典问题，社交平台希望通过最短路径算法来**分析信息如何在最短时间内传播到目标用户群**

挑战：**大规模社交网络的复杂性**、**传播路径的动态性**等都是此类问题中的重要因素

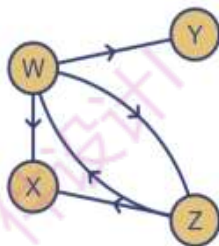
基于最短路径解决的问题

- [1496. 判断路径是否相交](#)
- [2335. 装满杯子需要的最短总时长](#)
- [3095. 或值至少 K 的最短子数组 I](#)
- [LCR 147. 最小栈](#)
- [LCR 088. 使用最小花费爬楼梯](#)
- [2578. 最小和分割](#)
- [1200. 最小绝对差](#)

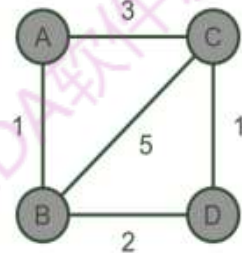
最短路径问题基于图分类



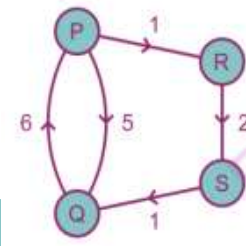
无权无向图



无权有向图



加权无向图

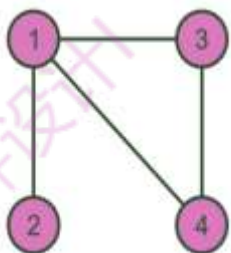


加权有向图

应用场景:

- 社会网络分析
- 网络路径优化
- 推荐系统
- 航班调度
- 游戏迷宫解法
- 任务调度
- 交通网络优化
- 货运物流
- 地铁站路径规划
- 金融交易优化
- 电路优化

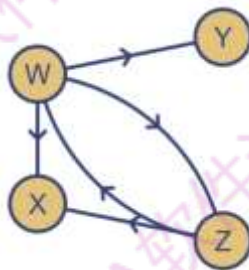
图不同，主流最短路径算法不同



无权无向图

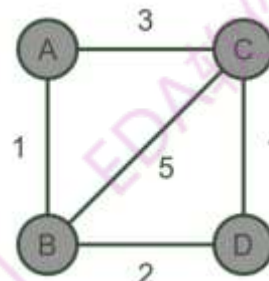
社会网络分析、
游戏迷宫解法、
地铁站路径规划

BFS



无权有向图

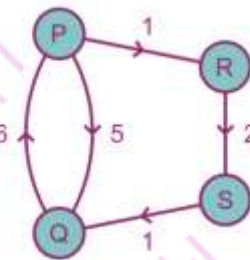
网络路径优化、
任务调度



加权无向图

推荐系统、交通
网络优化

Dijkstra Algorithm



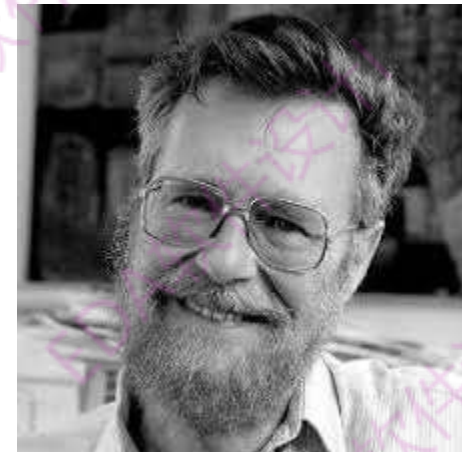
加权有向图

航班调度、货运
物流、金融交易
优化、电路优化

Dijkstra Algorithm

戴克斯特拉算法

Creator of Dijkstra Algorithm



Edsger Wybe Dijkstra

- 荷兰物理学家艾兹赫尔·维伯·戴克斯特拉于1956年设计

- 1972年因其对编程的基础性贡献，获得图灵奖

- 艾兹赫尔·W·戴克斯特拉分布式计算论文奖授予在分布式计算原理方面有杰出贡献的论文

- **分布式计算**是一种计算模型，将复杂任务分解为多个子任务，由多个计算机节点协同处理，从而提高计算速度和效率

- 名言：“Computer Science is no more about computers than astronomy is about telescopes.”

Facts about Dijkstra

- Dijkstra算法用于解决**单源最短路径问题**，适用于**带权重的有向或无向图**




- 它从一个起始顶点起，找到到所有其他顶点的最短路径（i.e., 权重之和最小的路径）
- Limitation: **不适用于具有负权边的图**
- 不同于MST，两顶点之间的最短距离可能不会包含图中的所有顶点
 - MST是一棵包含图中所有顶点的连通无环子图，使得边的总权重最小

- Dijkstra的最短路径算法由荷兰计算机科学家艾兹赫尔·W·戴克斯特拉于1956年在**一次购物时发明**

- 当时他正和未婚妻在阿姆斯特丹外出购物的过程中，**利用了20分钟coffee break时间**
- 发明这个算法的原因是为了测试一台新电脑——ARMAC

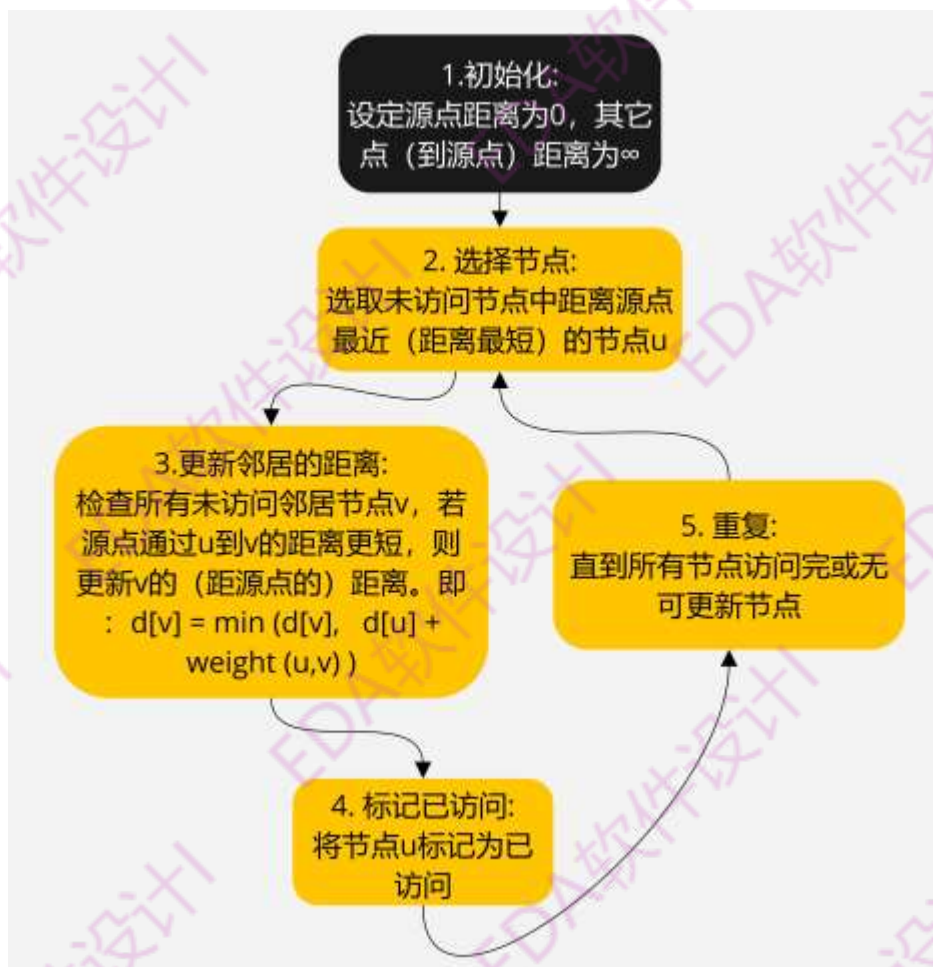
How to achieve it?

- 如果是你，你能够在20分钟内思考出这样一个算法（的思路）吗？
 - **Given:**
 - 加权图（有向或者无向）
 - Source node (源点)
 - **Produce:** 源点到所有其他顶点的最短路径（及其最短路径的权重）

 **思路（算法原理） → 贪心策略（Greedy）：每一步选择当前未访问节点中距离源点最近的节点，并通过它更新其他节点的最短路径**

How?

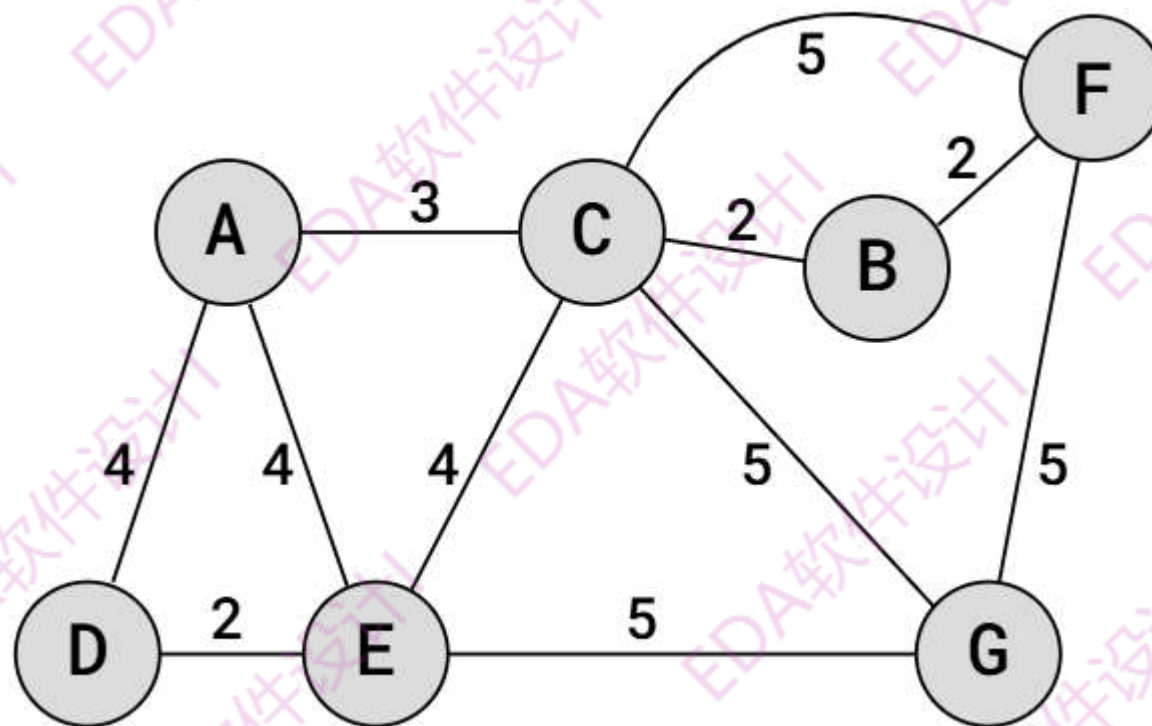
Dijkstra执行过程



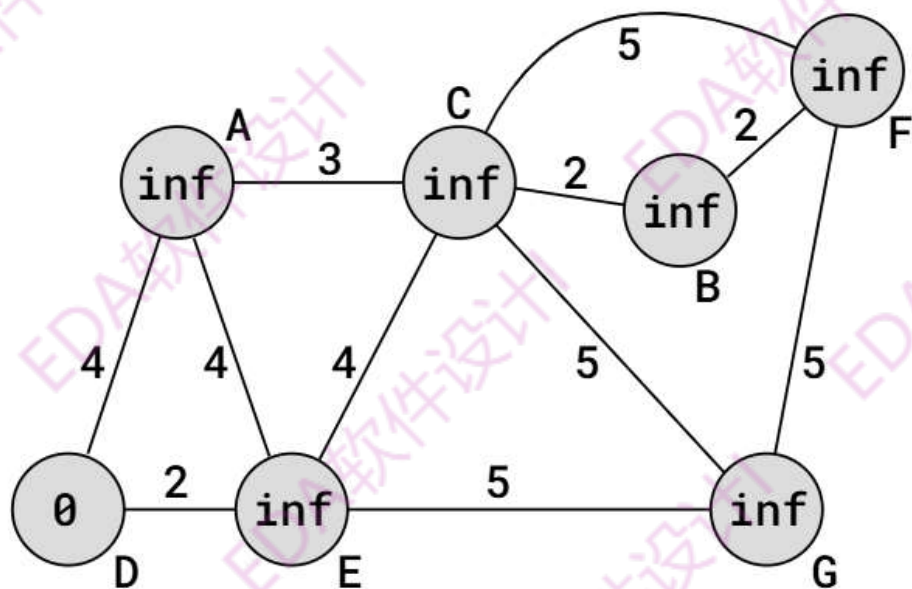
- ① 需要一个记录每个节点到源点最短距离的数据结构
 - $d[v]$ 表示点v到源点的最短距离
 - 选取什么数据结构?
- ② 如何实现访问顺序? 即每次都选择距离源点距离最近的节点访问
 - 使用哪种数据结构实现此操作?
- ③ 如何更新当前节点 (假设为u) 的 (未访问) 邻居的距离?
 - $d[v] = \min(d[v], d[u] + \text{weight}(u,v))$
- ④ 维护已访问节点

算法实现需要考虑和用到的“工具”

加权无向图上Run Dijkstra



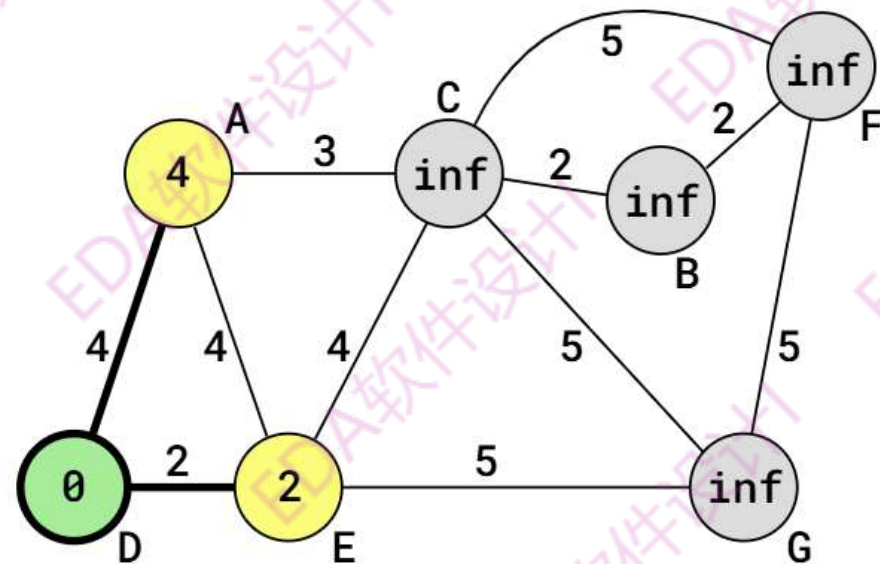
Dijkstra执行过程可视化



1. 初始化:

- 起始节点: D
- 初始距离:
 - $d[D] = 0$ (起点)
 - 对于所有其他节点 v , $d[v] = \infty$
- 已访问节点: 空

Dijkstra执行过程可视化



步骤 1:

- 当前节点: D

- 更新邻居:

- 对于节点 A:

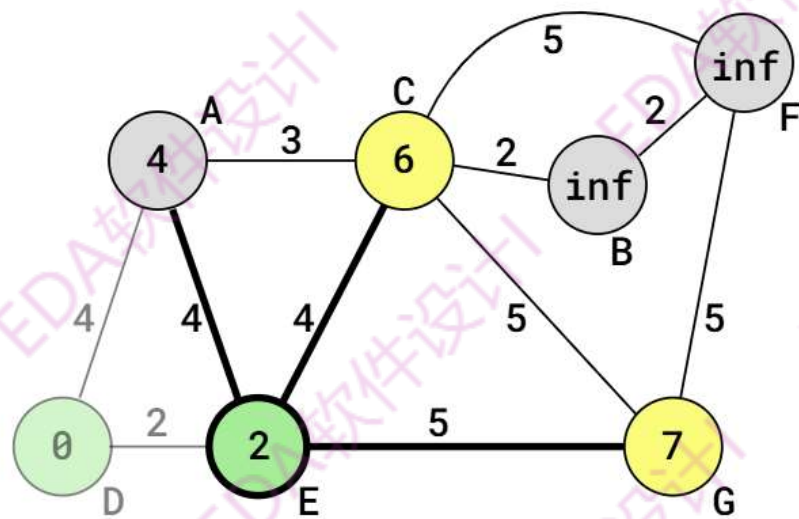
- $d[A] = \min(d[A], d[D] + \text{weight}(D, A)) = \min(\infty, 0 + 4) = 4$

- 对于节点 E:

- $d[E] = \min(d[E], d[D] + \text{weight}(D, E)) = \min(\infty, 0 + 2) = 2$

- 已访问节点: D

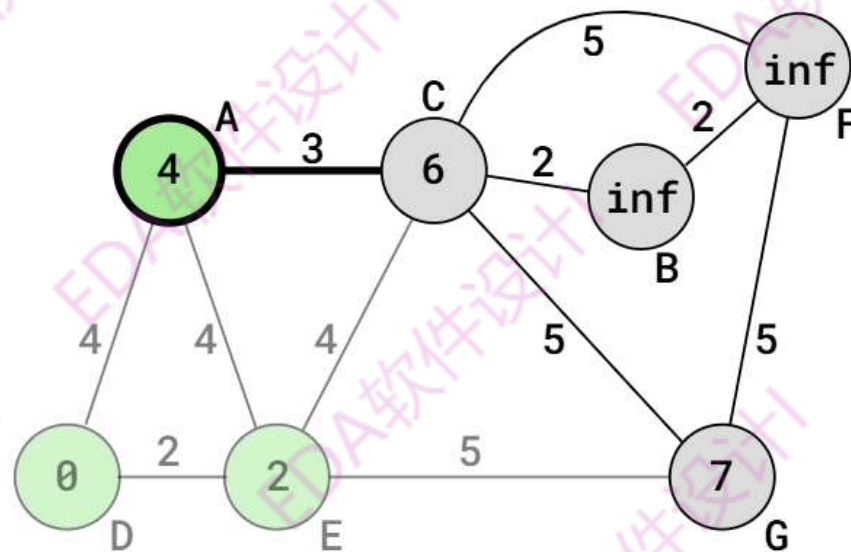
Dijkstra执行过程可视化



步骤 2:

- 当前节点: E ($d[E] = 2$)
- 更新邻居:
 - 对于节点 A:
 - $d[A] = \min(4, 2 + 4) = 4$ (保持不变)
 - 对于节点 C:
 - $d[C] = \min(d[C], d[E] + \text{weight}(E, C)) = \min(\infty, 2 + 4) = 6$
 - 对于节点 G:
 - $d[G] = \min(d[G], d[E] + \text{weight}(E, G)) = \min(\infty, 2 + 5) = 7$
- 已访问节点: D、E

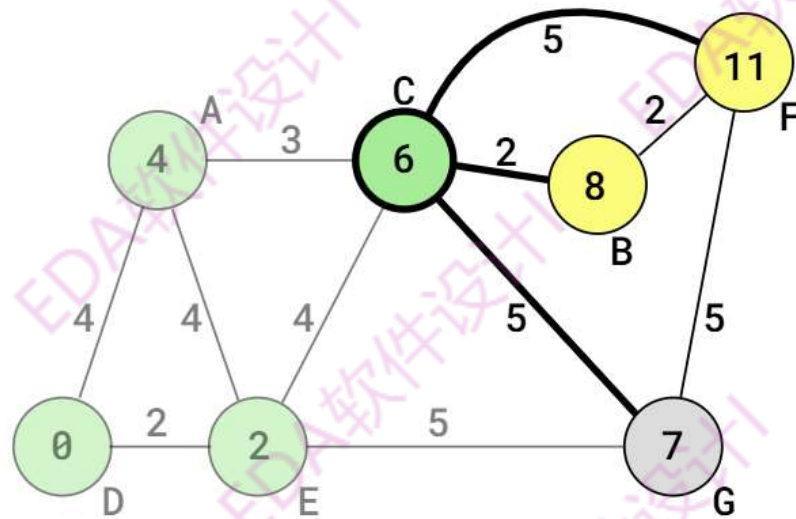
Dijkstra执行过程可视化



步骤 3:

- 当前节点: A ($d[A] = 4$)
- 更新邻居:
 - 对于节点 C:
 - $d[C] = \min(6, 4 + 3) = 6$ (保持不变)
- 已访问节点: D、E、A

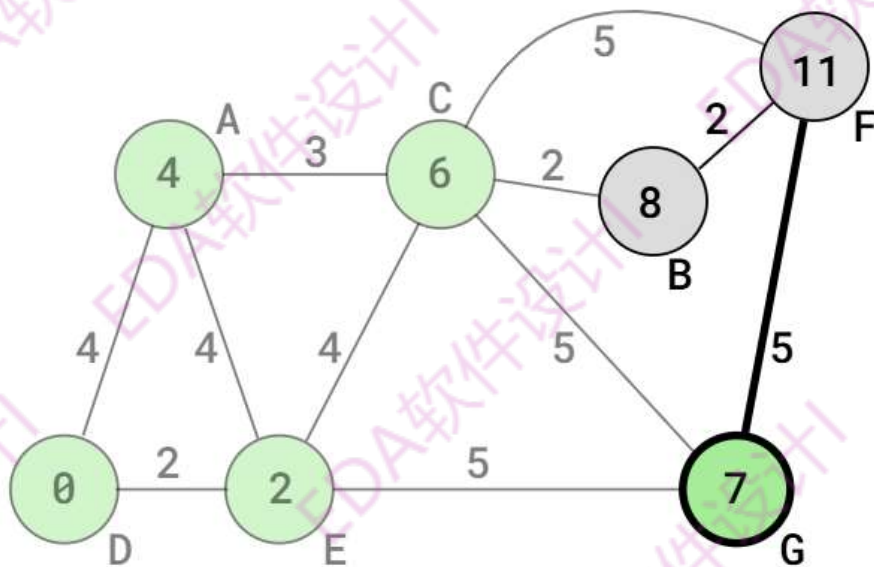
Dijkstra执行过程可视化



步骤 4:

- 当前节点: C ($d[C] = 6$)
- 更新邻居:
 - 对于节点 B:
 - $d[B] = \min(d[B], d[C] + \text{weight}(C, B)) = \min(\infty, 6 + 2) = 8$
 - 对于节点 F:
 - $d[F] = \min(d[F], d[C] + \text{weight}(C, F)) = \min(\infty, 6 + 5) = 11$
 - 对于节点 G:
 - $d[G] = \min(7, 6 + 5) = 7$ (保持不变)

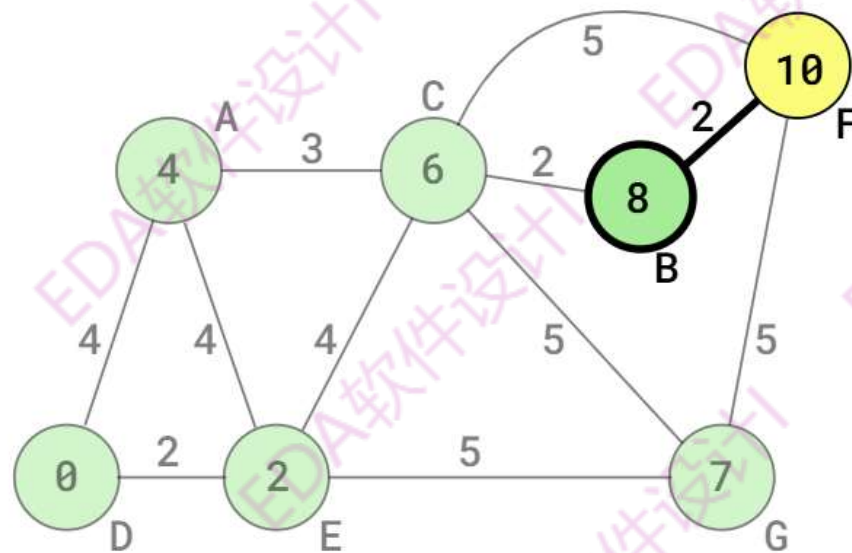
Dijkstra执行过程可视化



步骤 5:

- 当前节点: G ($d[G] = 7$)
- 更新邻居:
 - 对于节点 F:
 - $d[F] = \min(11, 7 + 5) = 11$ (保持不变)
- 已访问节点: D、E、A、C、G

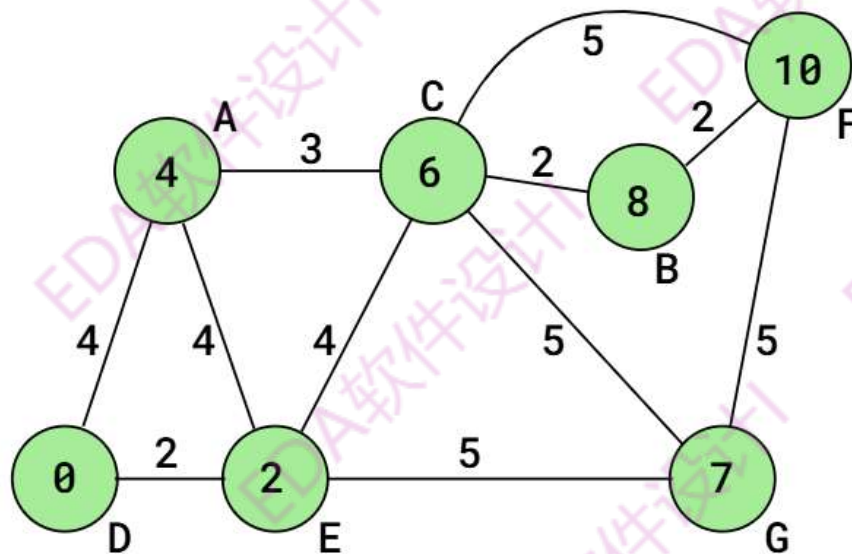
Dijkstra执行过程可视化



步骤 6:

- 当前节点: B ($d[B] = 8$)
- 更新邻居:
 - 对于节点 F:
 - $d[F] = \min(11, 8 + 2) = 10$
- 已访问节点: D、E、A、C、G、B

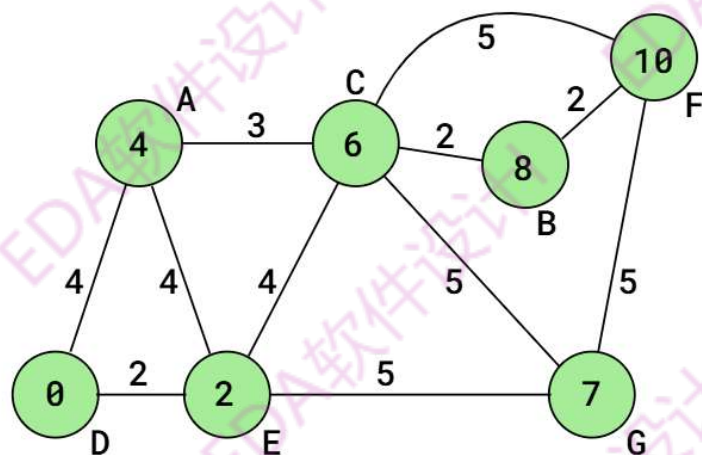
Dijkstra执行过程可视化



步骤 7:

- **当前节点:** F ($d[F] = 10$)
- **更新邻居:**
 - 无未访问的邻居需要更新。
- **已访问节点:** D、E、A、C、G、B、F

Dijkstra Output



最终的最短距离

- $d[D] = 0$
- $d[A] = 4$
- $d[E] = 2$
- $d[C] = 6$
- $d[G] = 7$
- $d[B] = 8$
- $d[F] = 10$

对应的最短路径是谁呢？？

- 💡 在算法的执行过程中需要维护每个节点的**前驱节点（也称为父节点）—— $\text{pred}[v]$**
- 💡 通过记录 **$\text{pred}[v]$** ，我们可以在算法结束构建从源点到目标节点的最短路径