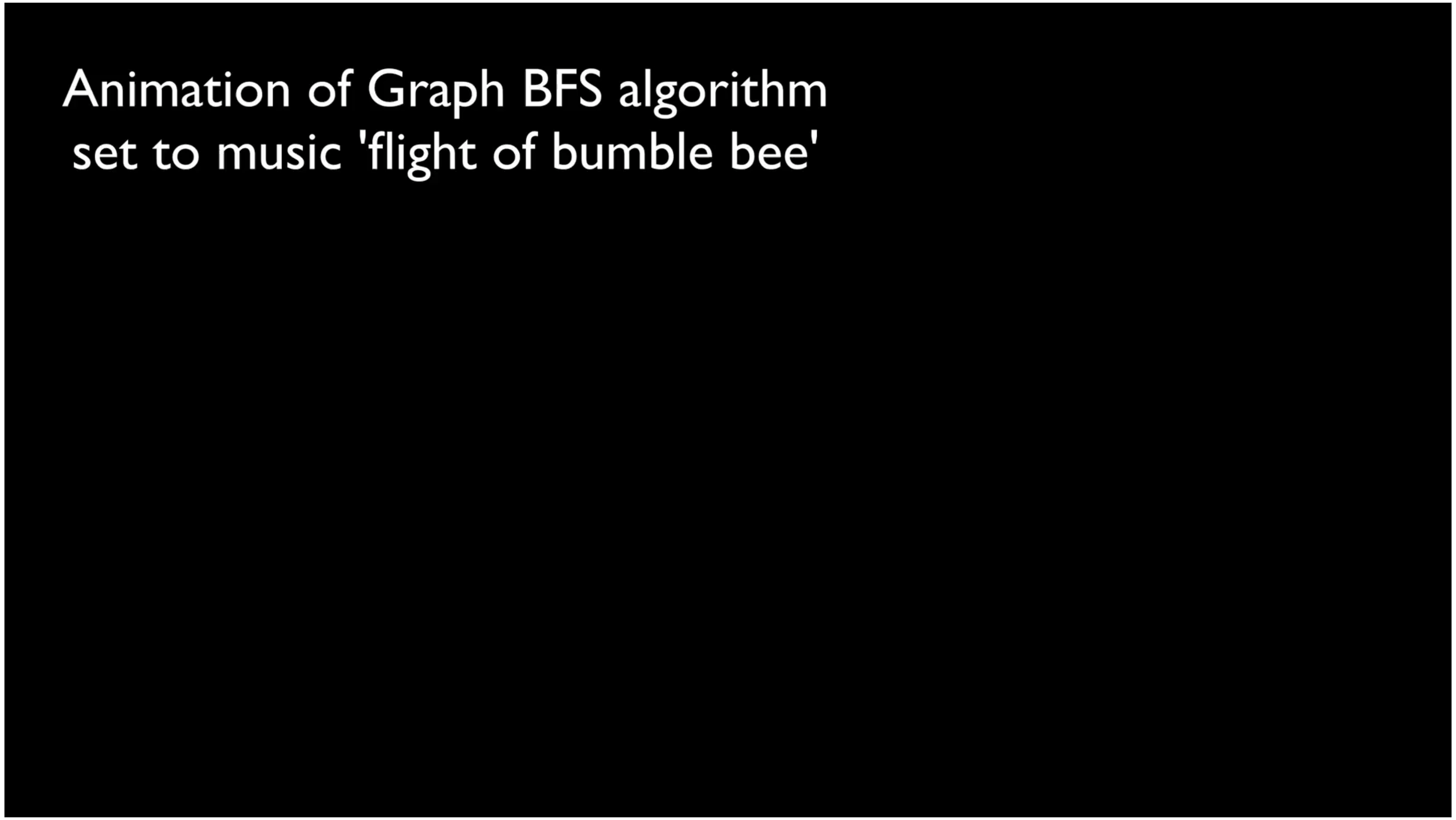
EDA 软件设计 I

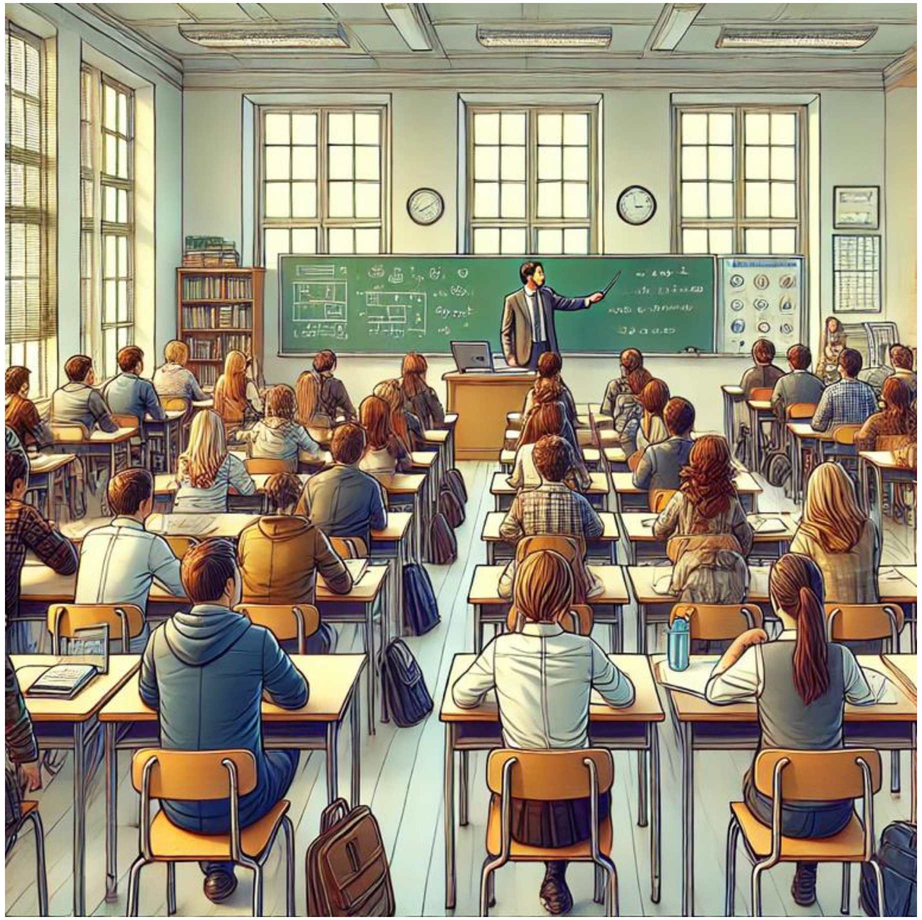
Lecture 6

Visualization of BFS



算法运行的效率

- 算法的运行速度与 input 的规模成“正相关”关系
- 这个“正相关”关系是一个函数关系——时间复杂度
- 算法运行时需要用到的内存资源——空间复杂度
- 一个小例子：教室数人头
  - 资源 v.s. 时间



算法：四大板块

✨⭐算法分析

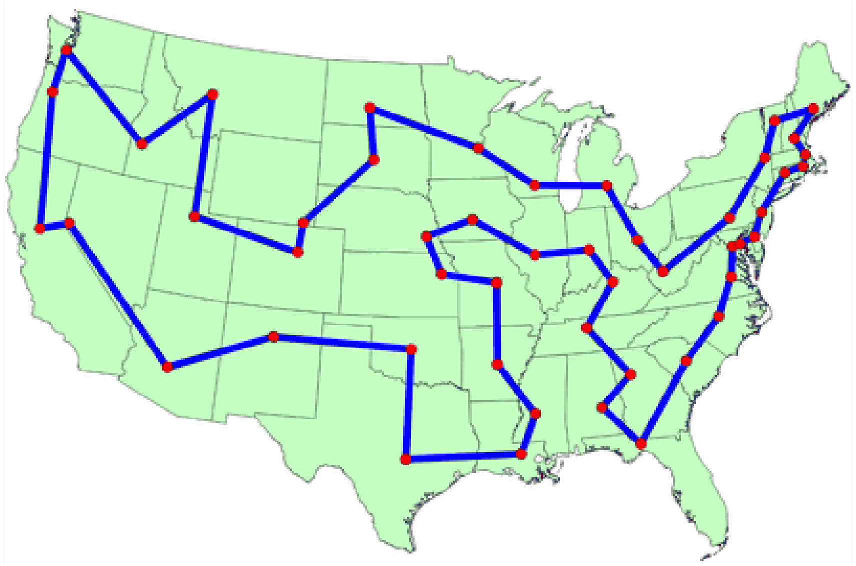
- 面试重点，甚至是必考
- 时间复杂度 T(n):
  - 衡量算法执行所需时间随input的规模增长而变化的函数
- 空间复杂度 S(n):
  - 衡量算法执行时所需的内存空间
  - 通常包括算法执行过程中所用到的额外变量、数组、递归栈等

✨⭐算法分析

1. 最坏情况：算法在最极端输入条件下的表现，提供算法的性能上限
   - 用 O(n) 表示
2. 平均情况：考虑算法在所有可能输入条件下的表现，常用于随机输入的场景下。
   - 用 Ω(n) 表示
3. 最好情况：算法在最理想输入条件下的表现，通常不作为选择算法的主要依据
   - 用 Θ(n) 表示
4. **一般情况下，关心最坏情况下的表现**
5. 理论分析与实际性能不同：虽然时间复杂度为 O(n) 的算法可能看起来比 O(n log n) 更慢，但在实际中，由于常数因素和硬件的影响，实际的表现可能不同

时间复杂度：重要性

- 经典问题：**旅行商问题（TSP, Travelling Salesman Problem）**
  - **问题描述**：给定一组城市和它们之间的距离，旅行商需要从一个城市出发，访问所有其他城市一次，并返回起点，目标是找到一条总路程最短的路径。



时间复杂度：重要性

- 经典问题：**旅行商问题（TSP，Travelling Salesman Problem）**
  - **暴力求解：** 时间复杂度为$O(n!)$
  - **动态规划（Held-Karp 算法）：** 时间复杂度为 $O(n^2 2^n)$
  - 假设：每秒能够执行 10^9次操作（现代计算机的典型速度）
  - **当 n = 30 时:**

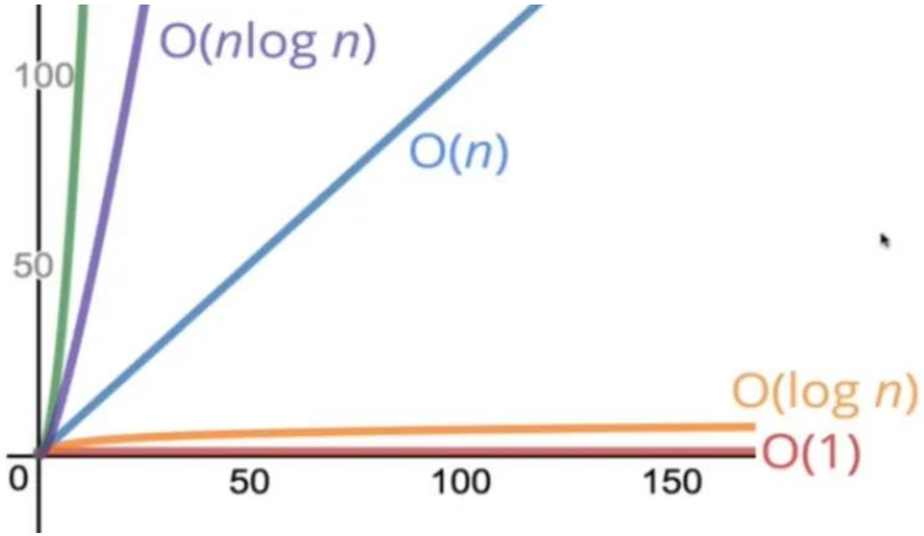
暴力求解需要时间：8.4 万亿年！！

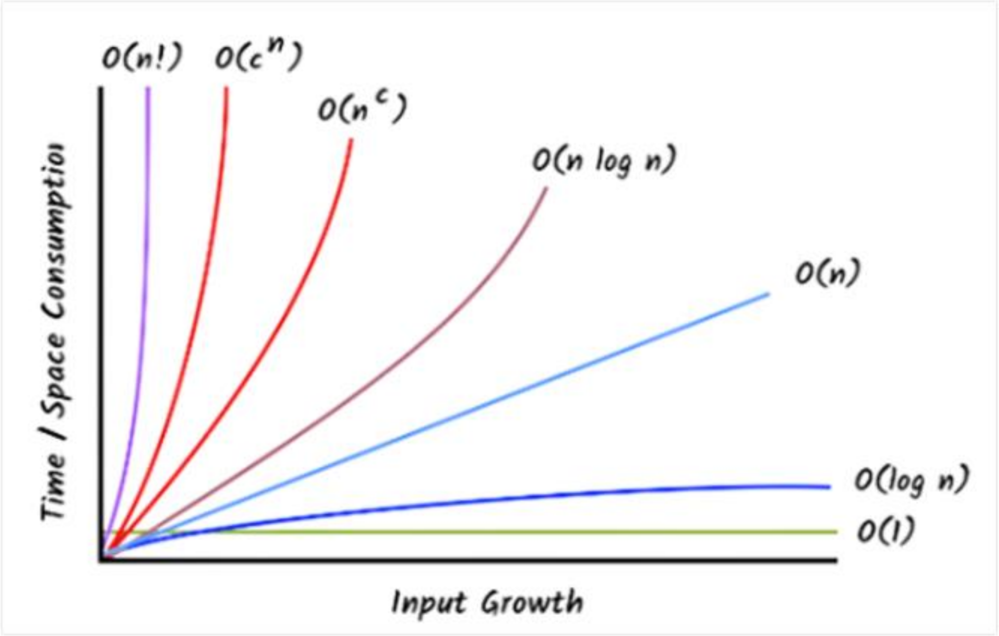
动态规划需要时间：16 分钟

常见时间复杂度

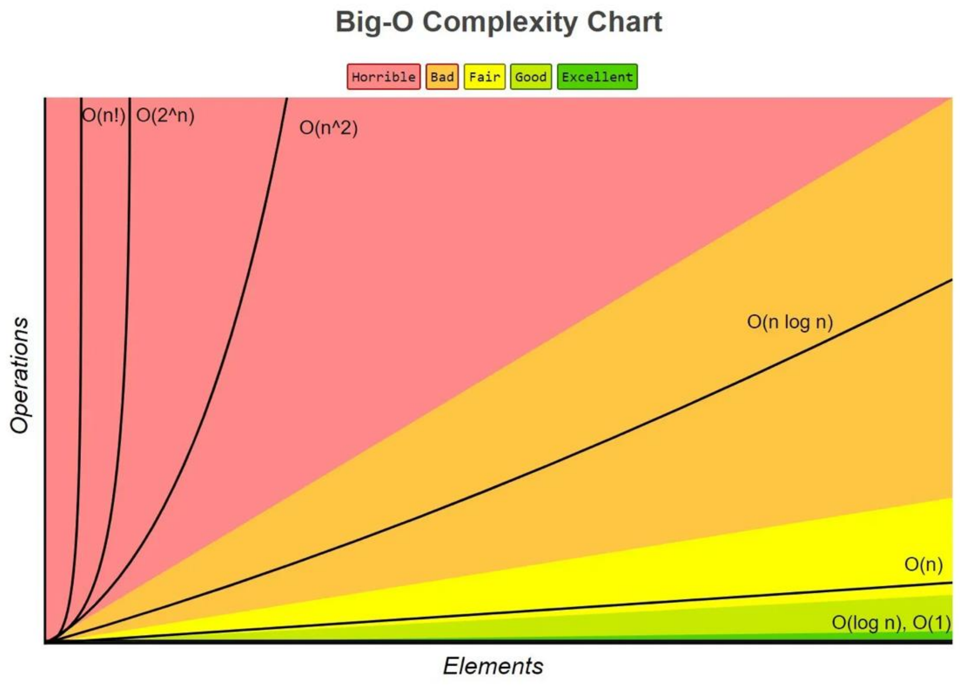
- $O(1)$: **常数时间复杂度**，时间不随着输入规模变化
  - 实例：获取数组中第一个元素
- $O(\log n)$：**对数时间复杂度**，时间增长缓慢，尤其是当 n 变得非常大时
  - 实例：二分查找算法，在一个有序数组中查找某个元素
- $O(n)$：**线性时间复杂度**，时间随着输入规模成等比例增长，适合处理中等规模数据，但在非常大数据的时，性能可能会成为瓶颈
  - 实例：遍历数组或列表中的每个元素，例如线性搜索
- $O(n \log n)$：**常见复杂度**，比线性复杂度略慢
  - 实例：归并排序和快速排序
- $O(n^2)$：**平方复杂度**，即使在中等规模的数据上也可能运行非常缓慢
  - 实例：冒泡排序和选择排序
- $O(2^n)$ ：**指数时间复杂度**，这种复杂度迅速增长，即使输入规模较小，运行时间也会非常快地变得不可接受，只能用于非常小得数据集
  - 实例：TSP的暴力揭发，递归地计算所有可能的路径

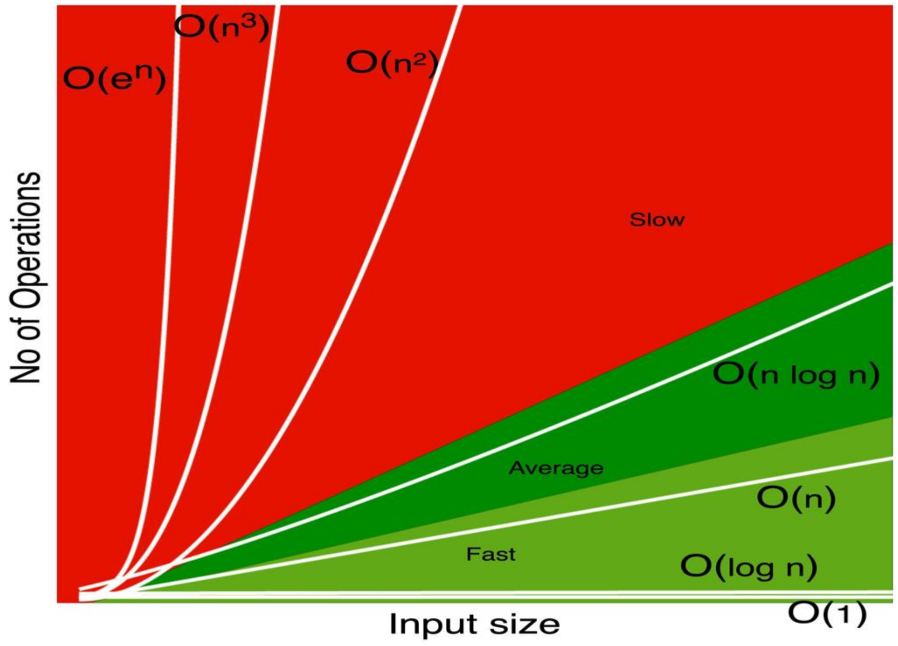
不同时间复杂度的实际差别





不同时间复杂度的实际差别





不同时间复杂度的实际差别

输入规模 (n)	(O(1))	(O(\log n))	(O(n))	(O(n \log n))	(O(n^2))	(O(2^n))
10	1 µs	3.32 µs	10 µs	33.2 µs	100 µs	1024 µs
100	1 µs	6.64 µs	100 µs	664 µs	10 ms	1.27E+30 µs
1,000	1 µs	9.97 µs	1 ms	9.97 ms	1 s	超出合理范围
10,000	1 µs	13.29 µs	10 ms	132.9 ms	100 s	超出合理范围
100,000	1 µs	16.61 µs	100 ms	1.66 s	超出合理范围	超出合理范围
1,000,000	1 µs	19.93 µs	1 s	19.93 s	超出合理范围	超出合理范围

不同算法表现的差别

实际时间差距示例

考虑处理一个数组的两种算法：

1. **线性搜索算法**：时间复杂度 $O(n)$，用于遍历数组中的每个元素，找到目标值。
2. **冒泡排序算法**：时间复杂度 $O(n^2)$，对数组进行排序。

假设我们处理的数组有 $n = 100,000$ 个元素：

- **线性搜索** 的时间复杂度是 $O(n)$，需要遍历每个元素一次，假设每个操作花费 1 微秒，处理时间约为 $100,000 \times 1\text{微秒} = 0.1$ 秒。
- **冒泡排序** 的时间复杂度是 $O(n^2)$，处理时间约为 $100,000^2 \times 1\text{微秒} = 10,000$ 秒（约 2.78 小时）。

显然，处理规模相同的数据，时间复杂度不同的算法在实际运行时间上的差距是非常巨大的。

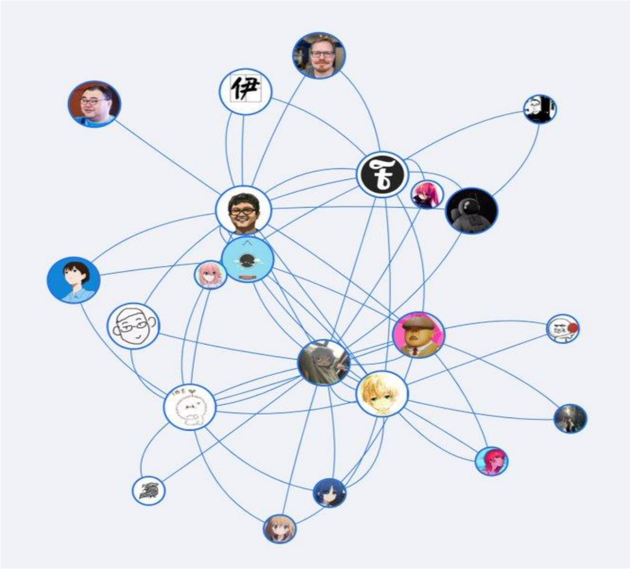
算法分析 @ BFS 遍历

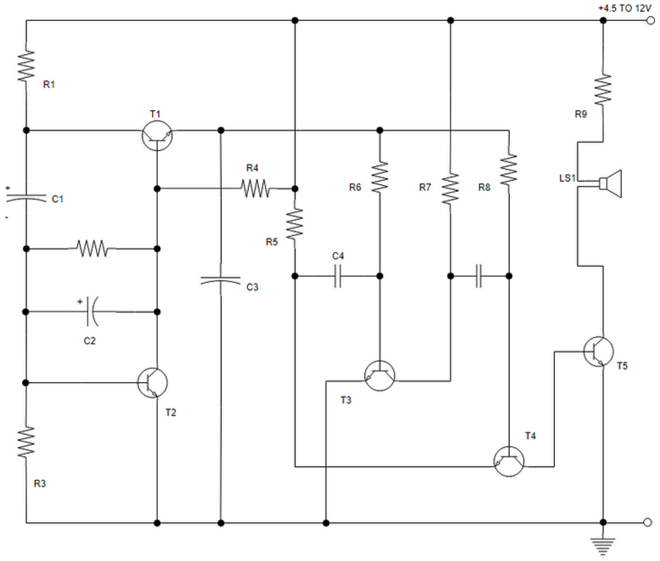
- 在图G=(V，E)中，BFS遍历的时间与空间复杂度：
  1. T(n) = O(|V|+|E|)
     - BFS会走过所有的节点，此为 O(|V|)
     - 在每个节点会检查所有的邻居（即节点的度或者与节点相邻的边的数量），此为 O(|E|)
     - 所以 BFS 遍历的时间复杂度为 O(|V|+|E|)
  2. S(n) = O(|V|)
     - 队列或者已访问列表里面最多可能存放所有节点

算法：四大板块

算法应用

如何利用算法来解决实际问题，是计算机科学和工程领域的核心主题之一





算法应用

使算法从理论走向实践，解决实际问题，提升系统或项目的性能。

算法在各个行业和领域中的应用

#	领域	常用算法	应用
1	大数据与数据挖掘	排序、搜索、聚类、分类	推荐系统、社交媒体分析、客户行为预测
2	机器学习与人工智能	线性回归、决策树、神经网络	图像识别、自然语言处理、自动驾驶
3	图论与网络	最短路径算法、最小生成树算法	导航系统、通信网络优化、社交网络中的好友推荐
4	密码学	AES、RSA 加密算法	网上银行、电子邮件加密、数字签名
5	优化问题与资源分配	动态规划、线性规划、贪心算法	生产计划、投资组合优化、物流路径优化

算法应用

搜索引擎	PageRank、搜索排序算法	快速找到相关网页、瞬间返回结果
推荐系统	协同过滤、矩阵分解	推荐个性化内容给用户
地图与导航	Dijkstra、A* 算法	找到最优路线
社交媒体	图算法、社交网络分析	推荐“可能认识的人”、感兴趣的内容

算法应用 @ BFS

- 基础:
  ① 寻找目标节点（存不存在）
  ② 找两点之间的最短路径（无权图中）
- In EDA
  ① 电路的可达性分析
  ② 时序分析
  ③ 布线优化
- General

算法应用 @ BFS （In EDA）

- In **EDA**

① 电路的可达性分析

- 背景：在集成电路设计中，通常需要确定一个信号从一个元件传递到另一个元件的路径
- 方式：从一个起点（例如信号源）开始，沿着电路网的连接逐层扩展，找到所有能够从源信号到达的节点，判断哪些元件可以被激活。这种方法可以有效地进行**信号传播路径**的分析。

② 时序分析

- 背景：时序分析在现代 EDA 工具中至关重要，用于确保电路在一定时间内能够正确传递信号
- 方式：BFS 可用于从一个时钟源开始，遍历所有逻辑门，分析信号沿着不同路径的延迟。通过这种遍历，EDA 工具可以找到最长路径或关键路径，用于分析**时序收敛**或**时序优化**。

③ 布线优化

- 背景：在物理设计的布线阶段，EDA 工具需要在两个或多个元件之间生成信号路径（即布线）
- BFS 可用于**层次式布线算法**中，帮助从一个起始点逐步探索所有可能的布线路径，以找到最短或资源最少的路径。在实际应用中，EDA 工具可能会结合 BFS 和其他算法来处理信号拥塞、延迟优化等问题。

算法应用 @ BFS （General）

Medium
图的连通性检测
迷宫求解

- **图的连通性检测:**
  - 判断图中是否一个点出发，可以访问到其他所有节点
  - Go deeper：检测网络或**电路**的连通性，查找社交网络中的朋友群体（连通分量）
- **迷宫求解:**
  - 保证找到从起点到终点的最短路径（如果存在）对于无权迷宫（每条边长度相等），BFS会按层次扩展，直到找到终点
  - Go deeper：机器人路径规划

算法应用 @ BFS （General）

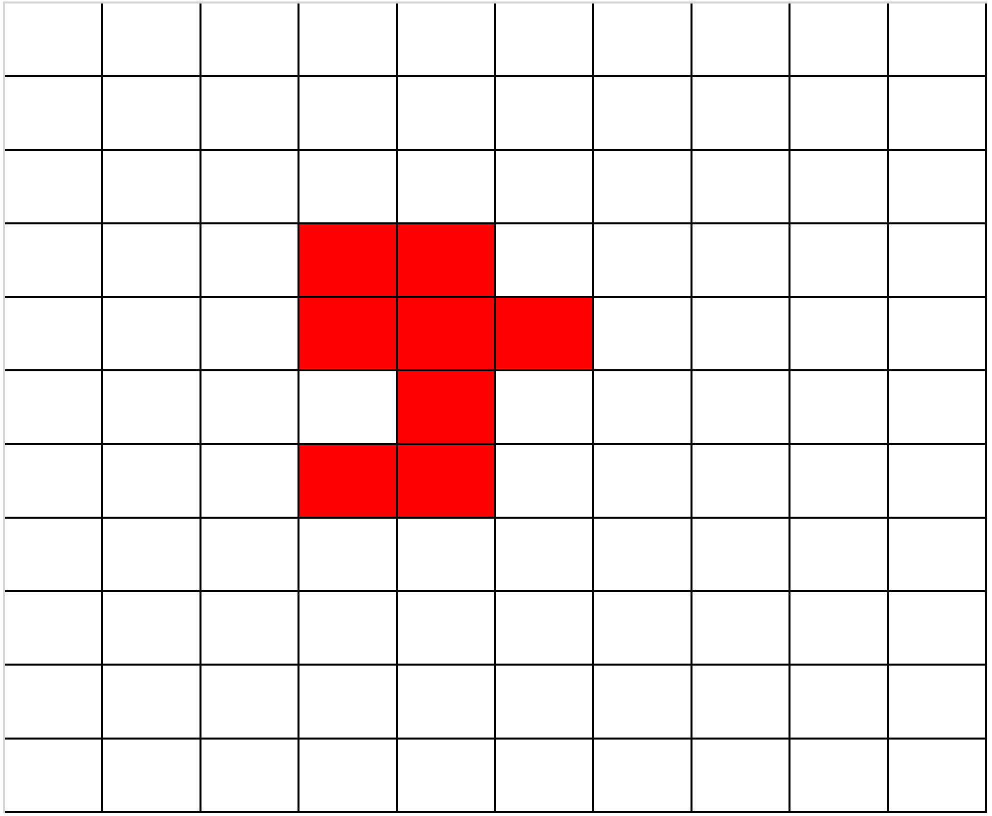
Hard
二分图检测
社交网络中的扩散问题

- **二分图检测:**
  - 检查一个图是否是二分图，即能否将图的节点分成两部分，使得每条边连接的节点都在不同部分
  - 通过BFS进行着色，可以判断一个图是否为二分图
  - Go deeper：用于网络流问题、匹配问题、社交网络中分组
- **社交网络中的扩散问题:**
  - BFS可用于分析社交网络中消息或病毒的传播范围，或者查找某个人与其他人之间的关系层次。
  - Go deeper: 在社交网络中查找“六度分隔理论”的验证，或者消息在社交平台上的扩散路径

算法应用≠应用算法

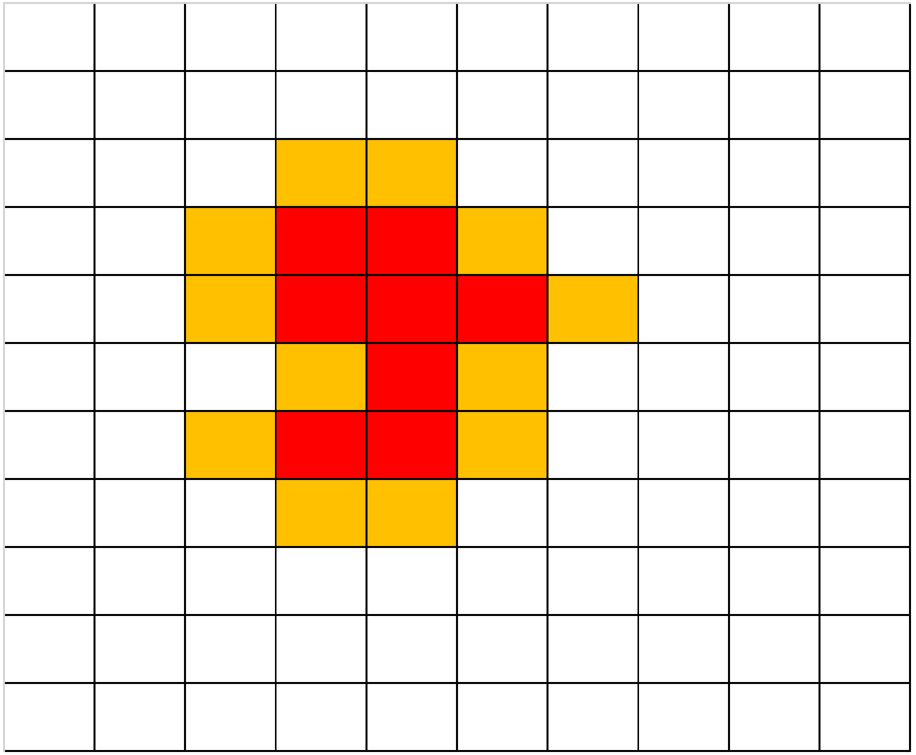
- 实际问题不会提示你运用什么算法或者采用哪个算法的策略
- 需要你对算法融会贯通、举一反三，来解决实际问题

扩展练习



左边的矩阵内存在一个红色区域，求出红色区域距离矩阵边缘的最短距离

扩展练习



黄色区域：bfs迈出的第一步

如果红色区域被定义为第一层，那么黄色区域就是bfs的第二层

扩展练习