图论 day1 基础

哈尔滨工业大学　王翰坤

2020 年 8 月 11 日

图 (Graph) 是什么

- 一堆点 (verticle, node) 和一堆边 (edge) 的共同构成的东西

图 (Graph) 是什么

- 一堆点 (verticle, node) 和一堆边 (edge) 的共同构成的东西
- 学术地说，$G =< V, E >$，其中 V 是点集，E 是边集
- 其中，边的表示形式也是二元组 (x, y)
- 如果边的二元组是有序二元组，则为有向图

- n 个点，m 条边的图，其邻接矩阵是一个 $n \times n$ 的 0/1 矩阵

- n 个点，m 条边的图，其邻接矩阵是一个 $n \times n$ 的 0/1 矩阵
- 适用于稠密图（边较多的图）、简单图（无重边和自环）
- 增删和访问某条边均为 O(1)，空间复杂度 O(n^2)
- 邻接矩阵自乘的意义

- 易于增删
- 增 $\mathrm{O}(1)$，删、访为 $\mathrm{O}(m)$，空间 $\mathrm{O}(m)$
- 链表、数组、vector

- 推荐
- 易于用数组方法实现，代码长度短，模板化程度高
- from，to，pre

- 推荐
- 易于用数组方法实现，代码长度短，模板化程度高
- from, to, pre
- 增 $O(1)$，删、访为 $O(m)$，空间 $O(m)$
- 总结：如果节点数规模较小且需要经常访问某已知两点间是否存在边，用邻接矩阵较方便；若要枚举从某个顶点出发的所有边，则用邻接表或前向星存储较方便

图的遍历

- BFS（广度优先，队列实现）
- DFS（深度优先，栈实现）
  - 应用：二分图（偶图）的判定
  - 二分图：这种无向图存在至少一种将所有节点黑白染色的方法，使得每条边两端的节点颜色都不同
  - 经典例子：恋爱关系图

- 定义：连通且无圈
- 其他性质

- 定义：连通且无圈
- 其他性质
  - 设图中节点数为 n，则边数恰为 $n-1$
  - 任意两点间路径唯一确定
  - 删去任意一条边，都不连通
  - 在任意无边相连的两节点间连一条边，恰好有一个圈

- 求最近公共祖先（LCA）、某些树上 DP 等许多问题的重要准备工作
- 直接用途：以某个特定的节点为根节点，获得所有非根节点的父亲节点

经典问题

无根树转有根树

- 求最近公共祖先（LCA）、某些树上 DP 等许多问题的重要准备工作
- 直接用途：以某个特定的节点为根节点，获得所有非根节点的父亲节点
- 一遍 DFS 即可

- 树的直径：树上最长链（边权和最大）

- 树的直径：树上最长链（边权和最大）
- 定理：从树上任意一点 u 出发，做一遍 DFS，找出离 u 最远的点 v，则 v 一定是直径的某一端点
- 思考证明

- 树的直径：树上最长链（边权和最大）
- 定理：从树上任意一点 u 出发，做一遍 DFS，找出离 u 最远的点 v，则 v 一定是直径的某一端点
- 思考证明
- 两遍 DFS 即可

- 也可以树上 DP，请大家自行思考

例题 I

- 给出一张 n 个点、m 条边的无向图，请判断它是否为“RXZ 图”：
  - RXZ 图是一棵树
  - RXZ 图上总存在一条链，使得图上的点要么在链上，要么与链上某个点的相邻
- $n \leq 10^5$，$m \leq 3 \times 10^5$

例题 I / POJ 3310
Solution

- 判断树很简单：$m = n - 1$ 且是连通图即可
- 第二项工作有多种方法
- 第一种，大胆猜测这条链指的就是树的直径：如果存在这样的链，则直径一定满足；反过来，如果直径不是这种链，也就不存在这样的链

例题 I / POJ 3310

Solution

- 判断树很简单：$m = n - 1$ 且是连通图即可
- 第二项工作有多种方法
- 第一种，大胆猜测这条链指的就是树的直径：如果存在这样的链，则直径一定满足；反过来，如果直径不是这种链，也就不存在这样的链
- 请自行思考证明

- 判断树很简单：$m = n - 1$ 且是连通图即可
- 第二项工作有多种方法
- 第一种，大胆猜测这条链指的就是树的直径：如果存在这样的链，则直径一定满足；反过来，如果直径不是这种链，也就不存在这样的链
- 请自行思考证明
- 于是只要两边 DFS 求出树的直径并标记，然后对每个点加以判断即可

例题 I / POJ 3310

Solution

- 判断树很简单：$m = n - 1$ 且是连通图即可
- 第二项工作有多种方法
- 第一种，大胆猜测这条链指的就是树的直径：如果存在这样的链，则直径一定满足；反过来，如果直径不是这种链，也就不存在这样的链
- 请自行思考证明
- 于是只要两边 DFS 求出树的直径并标记，然后对每个点加以判断即可
- 第二种，考虑每个点的的度数（相连的边数）
- 把所有度为 1 的点删去，若剩下的图是一条链，则是 RXZ 图
- 请思考正确性

- 定义：使最大子树的节点数最小的节点，叫重心
- 换句话说：把这个节点以及相邻的边删去后，留下的最大连通块是最小的

- 定义：使最大子树的节点数最小的节点，叫重心
- 换句话说：把这个节点以及相邻的边删去后，留下的最大连通块是最小的
- 先将无根树转化为有根树
- 设 f_i 为以节点 i 为根的子树节点数，显然有 $f_i = \sum_{j \in son} f_j + 1$
- 则最大连通块容易计算
- 一遍 DFS 即可解决问题

- 定义：使最大子树的节点数最小的节点，叫重心
- 换句话说：把这个节点以及相邻的边删去后，留下的最大连通块是最小的
- 先将无根树转化为有根树
- 设 f_i 为以节点 i 为根的子树节点数，显然有 $f_i = \sum_{j \in son} f_j + 1$
- 则最大连通块容易计算
- 一遍 DFS 即可解决问题

- 类似问题：树的最大独立集（选出尽量多的点使得任何两个结点均不相邻）

- 定义：使最大子树的节点数最小的节点，叫重心
- 换句话说：把这个节点以及相邻的边删去后，留下的最大连通块是最小的
- 先将无根树转化为有根树
- 设 f_i 为以节点 i 为根的子树节点数，显然有 $f_i = \sum_{j \in son} f_j + 1$
- 则最大连通块容易计算
- 一遍 DFS 即可解决问题

- 类似问题：树的最大独立集（选出尽量多的点使得任何两个结点均不相邻）
- DFS 过程中 DP 即可

并查集

- 两个操作：合并和查找

并查集

- 两个操作：合并和查找
- 路径压缩

并查集

- 两个操作：合并和查找
- 路径压缩
- 带权并查集

例题 II

- 给出一张 n 个点、m 条边的无向图。再给出一个 n 的排列，按此排列的顺序依次删去图中的点。要求每删一次就输出当前连通块的数量
- $n \leq 2 \times m$，$m \leq 2 \times 10^5$

例题 II / BZOJ 1015

Solution

- 删点在大多数存储结构中都是相当麻烦的操作，更别说实时统计连通块数了
- 我们擅长的是加点

例题 II / BZOJ 1015

Solution

- 删点在大多数存储结构中都是相当麻烦的操作，更别说实时统计连通块数了
- 我们擅长的是加点
- 注意到这是一个离线的题目，我们可以按照自己想要的顺序处理询问

- 删点在大多数存储结构中都是相当麻烦的操作，更别说实时统计连通块数了
- 我们擅长的是加点
- 注意到这是一个离线的题目，我们可以按照自己想要的顺序处理询问
- 倒序处理即可

- 这里的带权图指带边权
- 树是使一堆节点连通的最节省方案
- 最小生成树：原图的“极小连通子图”，即包含原图所有 n 个节点且仍然连通的前提下，使边权之和最小的子图

- 这里的带权图指带边权
- 树是使一堆节点连通的最节省方案
- 最小生成树：原图的“极小连通子图”，即包含原图所有 n 个节点且仍然连通的前提下，使边权之和最小的子图
- Kruskal、Prim 算法均可解决

- 这里的带权图指带边权
- 树是使一堆节点连通的最节省方案
- 最小生成树：原图的“极小连通子图”，即包含原图所有 n 个节点且仍然连通的前提下，使边权之和最小的子图
- Kruskal、Prim 算法均可解决
- 拓展：次小生成树
- 一定是加入剩下的边中最小的那条吗?

- 这里的带权图指带边权
- 树是使一堆节点连通的最节省方案
- 最小生成树：原图的“极小连通子图”，即包含原图所有 n 个节点且仍然连通的前提下，使边权之和最小的子图
- Kruskal、Prim 算法均可解决
- 拓展：次小生成树
- 一定是加入剩下的边中最小的那条吗?
- 枚举所有未被 MST 选中的边加入，形成的环上的最大的边替代掉，取最优解
- 时间复杂度为 O(nm)，较高
- 查找环上最大边，可用树上 LCA 优化

例题 III

- 给出一张 n 个点、m 条边的无向图，求这个图的严格次小生成树的边权和
- $n \leq 10^5$，$m \leq 3 \times 10^5$

- 把加边后找环的过程用倍增思想优化，即可在 $O(nlogn)$ 的时间复杂度下完成求解

树上 LCA

- LCA：Lowest Common Ancestors，最近公共祖先
- 朴素想法

树上 LCA

- LCA：Lowest Common Ancestors，最近公共祖先
- 朴素想法
- 常用优化一：利用倍增思想，每次往上跳 2^k 层，将 $O(n)$ 优化为 $O(logn)$
- 常用优化二：树链剖分，代码比倍增短且效率更高，有兴趣可自行搜索

拓扑图

- 别名：有向无环图、DAG(Directed acyclic graph)
- 小题目：给出一个 DAG，求它的生成树个数

拓扑图

- 别名：有向无环图、DAG(Directed acyclic graph)
- 小题目：给出一个 DAG，求它的生成树个数
- 拓扑排序：对节点排序，使得若存在有序关系 (u, lll)，那么满足排序后的序列中 u 在 v 前
- 如，给出 $a < b$，$b < d$，$c < a$ 这样的若干组无矛盾的关系，要求一个可行的从小到大的顺序

拓扑图

- 别名：有向无环图、DAG(Directed acyclic graph)
- 小题目：给出一个 DAG，求它的生成树个数
- 拓扑排序：对节点排序，使得若存在有序关系 (u, III)，那么满足排序后的序列中 u 在 v 前
- 如，给出 $a < b$，$b < d$，$c < a$ 这样的若干组无矛盾的关系，要求一个可行的从小到大的顺序
- DAG 性质：每个 DAG 都存在至少一个“源”（没有指向它的边），至少存在一个“汇”（没有从它指出的边）

拓扑图

- 别名：有向无环图、DAG(Directed acyclic graph)
- 小题目：给出一个 DAG，求它的生成树个数
- 拓扑排序：对节点排序，使得若存在有序关系 (u, III)，那么满足排序后的序列中 u 在 v 前
- 如，给出 $a < b$，$b < d$，$c < a$ 这样的若干组无矛盾的关系，要求一个可行的从小到大的顺序
- DAG 性质：每个 DAG 都存在至少一个“源”（没有指向它的边），至少存在一个“汇”（没有从它指出的边）
- 拓扑排序也可以用来判断一个图是不是 DAG（排序失败，则说明该图存在有向环）

拓扑图

- 别名：有向无环图、DAG(Directed acyclic graph)
- 小题目：给出一个 DAG，求它的生成树个数
- 拓扑排序：对节点排序，使得若存在有序关系 (u, III)，那么满足排序后的序列中 u 在 v 前
- 如，给出 $a < b$，$b < d$，$c < a$ 这样的若干组无矛盾的关系，要求一个可行的从小到大的顺序
- DAG 性质：每个 DAG 都存在至少一个“源”（没有指向它的边），至少存在一个“汇”（没有从它指出的边）
- 拓扑排序也可以用来判断一个图是不是 DAG（排序失败，则说明该图存在有向环）
- DAG、拓扑序与 DP（动态规划）的联系

经典问题

欧拉回路（“一笔画”）

- 欧拉道路：从无向图的一个点出发，每条边恰好经过一次，这样的路线称为“欧拉道路”
- 欧拉回路：这条路线要回到起点
- 一个图是否存在欧拉道路（回路）的判定方法
  - 无向图

- 欧拉道路：从无向图的一个点出发，每条边恰好经过一次，这样的路线称为“欧拉道路”
- 欧拉回路：这条路线要回到起点
- 一个图是否存在欧拉道路（回路）的判定方法
  - 无向图
  - 有向图

- 欧拉道路：从无向图的一个点出发，每条边恰好经过一次，这样的路线称为“欧拉道路”
- 欧拉回路：这条路线要回到起点
- 一个图是否存在欧拉道路（回路）的判定方法
  - 无向图
  - 有向图
  - 思考：混合图的情形 (UVa 10735)；输出欧拉道路（回路）的路径

- 欧拉道路：从无向图的一个点出发，每条边恰好经过一次，这样的路线称为“欧拉道路”
- 欧拉回路：这条路线要回到起点
- 一个图是否存在欧拉道路（回路）的判定方法
  - 无向图
  - 有向图
  - 思考：混合图的情形 (UVa 10735)；输出欧拉道路（回路）的路径
- 相关问题：哈密尔顿回路，经过所有点恰好一次；无多项式判断方法

- 有 Dijkstra、Bellman-Ford、Floyd 三大算法
- Dijkstra 算法：单源最短路径

- 有 Dijkstra、Bellman-Ford、Floyd 三大算法
- Dijkstra 算法：单源最短路径
- 以起始点为中心向外层层扩展（广度优先搜索 BFS 思想）
- 松弛操作：
  if dis[y] $>$ dis[x] $+$ weight[x, y]
  then dis[y] $=$ dis[x] $+$ weight[x, y]
- 因此，松弛操作也保证了对于每对相邻节点的 (x, y)，都有 dis[x] $\leq$ dis[y] $+$ weight[x, y]
- 这个不等式被称为“三角不等式”

- 有 Dijkstra、Bellman-Ford、Floyd 三大算法
- Dijkstra 算法：单源最短路径
- 以起始点为中心向外层层扩展（广度优先搜索 BFS 思想）
- 松弛操作：
  if dis[y] $>$ dis[x] $+$ weight[x, y]
  then dis[y] $=$ dis[x] $+$ weight[x, y]
- 因此，松弛操作也保证了对于每对相邻节点的 (x, y)，都有 dis[x] $\leq$ dis[y] $+$ weight[x, y]
- 这个不等式被称为“三角不等式”
- 用堆（或优先队列）优化
- 思考：输出最短路径
- 性质：有向正权图的最短路图一定是个 DAG

- Bellman-Ford 算法

- Bellman-Ford 算法
- 当图中存在负权时，最短路甚至不一定存在
- 若图中不含负圈，那么最短路至多经过 $n-1$ 个节点
- 每一轮松弛都会把最短路“扩张一层”
- 因此最多只需要 $n-1$ 轮松弛
- 如果第 n 轮松弛仍能使某些点的最短路径变短，则存在负圈

- Bellman-Ford 算法
- 当图中存在负权时，最短路甚至不一定存在
- 若图中不含负圈，那么最短路至多经过 $n-1$ 个节点
- 每一轮松弛都会把最短路“扩张一层”
- 因此最多只需要 $n-1$ 轮松弛
- 如果第 n 轮松弛仍能使某些点的最短路径变短，则存在负圈
- Bellman-Ford 算法可用队列优化，即所谓的 SPFA
- 思考：为什么带负边权的图不能用 Dijkstra 算法?

例题 IV

- 给出一张 n 个点、m 条边的无向图。问 1 号节点到 n 号节点的次短路径
- $n \leq 5000, m \leq 10^4$

- 到某个路口 v 的次短路要么来源于某个相邻接点 u 的最短路加边权 $w(u, v)$，要么来源于某相邻接点 u 的次短路加边权 $w(u, v)$
- 记录每个点的最短路长度、次短路长度，丢到 Dijkstra 里即可

- 求所有点对间的最短路径，Floyd 算法

- 求所有点对间的最短路径，Floyd 算法
- 本质上是 DP
- 设 $f[k][i][j]$ 为考虑前 k 个点为“中间点”，i 和 j 的最短距离
- $f[k][i][j] = \min\{f[k-1][i][j], f[k-1][i][k] + f[k-1][k][j]\}$
- 显然状态的第一维可以不要
- 最外层的循环一定要是阶段 k，里面双循环枚举两个节点

- 求所有点对间的最短路径，Floyd 算法
- 本质上是 DP
- 设 $f[k][i][j]$ 为考虑前 k 个点为“中间点”，i 和 j 的最短距离
- $f[k][i][j] = \min\{f[k-1][i][j], f[k-1][i][k] + f[k-1][k][j]\}$
- 显然状态的第一维可以不要
- 最外层的循环一定要是阶段 k，里面双循环枚举两个节点
- 思考衍生问题：最小环问题，即求一个无向正权图里的最小环长度（环上边权之和）

- 匹配：指二分图边集 E 的一个子集 E'，其中任意两条边都没有公共点
- 最大匹配：边数最多的匹配；特别地，如果最大匹配中包含所有的点，则称为“完美匹配”

- 匹配：指二分图边集 E 的一个子集 E'，其中任意两条边都没有公共点
- 最大匹配：边数最多的匹配；特别地，如果最大匹配中包含所有的点，则称为“完美匹配”
- 求最大匹配的常用方法：匈牙利算法、* 网络流方法
- 核心概念：增广路径
  - 相对于某个匹配而言
  - 两端是两个未匹配点
  - 假设已经获得了匹配 M，则属于 M 的边和不属于 M 的边交替出现
  - 不难发现，增广路的边数一定是奇数，且总比当前匹配的边数多
  - 通过异或操作即可获得更大的匹配 M'
  - 当前匹配是最大匹配，当且仅当找不到增广路径

- 匹配：指二分图边集 E 的一个子集 E'，其中任意两条边都没有公共点
- 最大匹配：边数最多的匹配；特别地，如果最大匹配中包含所有的点，则称为“完美匹配”
- 求最大匹配的常用方法：匈牙利算法、* 网络流方法
- 核心概念：增广路径
  - 相对于某个匹配而言
  - 两端是两个未匹配点
  - 假设已经获得了匹配 M，则属于 M 的边和不属于 M 的边交替出现
  - 不难发现，增广路的边数一定是奇数，且总比当前匹配的边数多
  - 通过异或操作即可获得更大的匹配 M'
  - 当前匹配是最大匹配，当且仅当找不到增广路径
- 思考匈牙利算法的实现

二分图

- 二分图最小顶点覆盖：假定选择了一个点，就覆盖了所有与之相邻的边。那么，在二分图中最少选择多少个点，才能覆盖所有的边?

二分图

- 二分图最小顶点覆盖：假定选择了一个点，就覆盖了所有与之相邻的边。那么，在二分图中最少选择多少个点，才能覆盖所有的边?
  - 等价于最大匹配，思考原因

二分图

- 二分图最小顶点覆盖：假定选择了一个点，就覆盖了所有与之相邻的边。那么，在二分图中最少选择多少个点，才能覆盖所有的边?
  - 等价于最大匹配，思考原因
  - 推导出增广路，与“最大”矛盾
- 二分图最大独立集：在二分图中选出最多的两两不相邻的节点

二分图

- 二分图最小顶点覆盖：假定选择了一个点，就覆盖了所有与之相邻的边。那么，在二分图中最少选择多少个点，才能覆盖所有的边?
  - 等价于最大匹配，思考原因
  - 推导出增广路，与“最大”矛盾
- 二分图最大独立集：在二分图中选出最多的两两不相邻的节点
  - 最小顶点覆盖选剩下的点，就是一个最大独立集
  - $|$ 最大独立集 $| = n -|$ 最小顶点覆盖 $|$
  - 证明方法与上面类似
- 二分图最大团：在二分图中选出最多的两两相邻的节点

二分图

- 二分图最小顶点覆盖：假定选择了一个点，就覆盖了所有与之相邻的边。那么，在二分图中最少选择多少个点，才能覆盖所有的边?
  - 等价于最大匹配，思考原因
  - 推导出增广路，与“最大”矛盾
- 二分图最大独立集：在二分图中选出最多的两两不相邻的节点
  - 最小顶点覆盖选剩下的点，就是一个最大独立集
  - ｜最大独立集｜$= n$ -｜最小顶点覆盖｜
  - 证明方法与上面类似
- 二分图最大团：在二分图中选出最多的两两相邻的节点
  - 暴力转化：求补图的最大独立集

例题 V

- 有 n 个人，m 对好友关系（好友关系双向但不可传递）。试将所有人分为两组，使得每组内部所有人之间都互为好友
- 问是否存在合法的分配方案；若有，给出任意一种
- $n \le 1000$

- 思考反面，联想到二分图的性质：同色点间都没有边相连
- 若 i 和 j 不是好友，则将 i 和 j 连边，则我们希望同组人间不存在这种“非好友”边
- 找出完美匹配即可

例题 VI

- 给出一张 $n \times n$ 的格子棋盘，每个格子都只有黑白两种颜色。你可以把某两行或某两列进行交换，交换顺序和次数不限。给出所有格子的初始颜色。问，是否可以将棋盘的主对角线（左上角到右下角）上的所有格子全部变成黑色?
- $n \leq 300$

- 注意到无论怎么使用这两种操作，都无法使原本不在同一行的换到同一行、无法使原本不在同一列的换到同一列
- 即，可以将对角线变为全黑等价于存在 n 个初始行列互不相同的黑色格子
- 如何判断是否存在呢?

- 注意到无论怎么使用这两种操作，都无法使原本不在同一行的换到同一行、无法使原本不在同一列的换到同一列
- 即，可以将对角线变为全黑等价于存在 n 个初始行列互不相同的黑色格子
- 如何判断是否存在呢?
- 对每行、每列都建立一个节点，共 $2n$ 个
- 对于黑色格子 (i, j)，在代表行 i 和列 j 的节点之间连一条无向边
- 做一次二分图匹配，若存在完美匹配，则存在这样的 n 个黑色格子
- 建图复杂度为 $O(n^2)$，匹配时间复杂度即为 $O(n^3)$，总的复杂度即为 $O(n^3)$

例题 VII

- 给出一张 n 个点、m 条边的无向图（无重边、自环）。现给定 s, t, d_s, d_t，要求一棵生成树，使得在生成树中，节点 s 的度数不超过 d_s、节点 t 的度数不超过 d_t
- $n \leq 2 \times 10^5$，$m \leq min(4 \times 10^5,\ n \cdot (n-1)/2)$

- 既然 s 和 t 这么重要，我们就先不管它们
- 暂时删去 s、t 以及相关的边，剩下的图（不一定连通）做并查集处理，把每个连通分量缩成一个点
- 把缩后的点分成三类
  - A. 没有与 s、t 相连的边
    B. 仅有与 s 相连的边，或仅有与 t 相连的边
    C. 同时具有与 s、t 相连的边

- 既然 s 和 t 这么重要，我们就先不管它们
- 暂时删去 s、t 以及相关的边，剩下的图（不一定连通）做并查集处理，把每个连通分量缩成一个点
- 把缩后的点分成三类
  - A. 没有与 s、t 相连的边
    B. 仅有与 s 相连的边，或仅有与 t 相连的边
    C. 同时具有与 s、t 相连的边
- 对于 A 型点，存在即无解；对于 B 型点，别无选择，只能连上。
- 剩下两个问题。A、B 型点做完后，如何处理 C 型点？如何处理 s 和 t 的连通问题？

- C 型点的处理：贪心之，取 s 和 t 二者中所剩的度数较多那一个连上即可

- C 型点的处理：贪心之，取 s 和 t 二者中所剩的度数较多那一个连上即可
- s 和 t 的连通
  - 若存在 C 型点，则把两头都连上即可
  - 若不存在 C 型点，但存在 s-t 直接相连的边，就用上它。这种方案显然没有第一种方案节约，所以只有在迫不得已（无 C 型点）的时候才用
  - 若 C 型点和 s-t 直连边都不存在，则无解
- 时间复杂度为 $O(nlogn + m)$

例题 VIII

- 给出一张 n 个点、m 条边的无向图，通过一条边要一定的时间
- 每条路上都可能有几种怪兽，一共有 p 种怪兽
- 每个节点上都有一个装备师，他们各怀技能，可以制造消灭特定几种怪兽的武器免费供你使用
- 你能通过某条边，当且仅当你拥有消灭这条边上所有怪兽的武器
- 现给出每条边的通过时间、存在的怪兽，以及每个装备师可以制造的武器种类，问从 1 号点到 n 号点所需的最短时间
- $n \leq 200, m \leq 3000, p < 13$

- 记第 i 个点可获得的武器集合为 S_i，边 (i, j) 上的怪兽集合为 E_{ij}
- 注意到 p 的规模，我们可以重构这张图
- 把每个节点拆成 2^p 个新节点。节点 i_K 表示到了原图的节点 i 时，可消灭的怪兽集合为 K
- 思考连边方法

Solution

- 记第 i 个点可获得的武器集合为 S_i，边 (i,j) 上的怪兽集合为 E_{ij}
- 注意到 p 的规模，我们可以重构这张图
- 把每个节点拆成 2^p 个新节点。节点 i_K 表示到了原图的节点 i 时，可消灭的怪兽集合为 K
- 思考连边方法
  - 点 (i,K) 到 $(i,K\cup S_i)$，连边权为 0 的有向边
  - 对于点 (i,K) 和 (j,K')，若原图中 i 和 j 相邻，且 K' 是 K 的子集，则连一条边权为原边权的有向边

- 记第 i 个点可获得的武器集合为 S_i，边 (i, j) 上的怪兽集合为 E_{ij}
- 注意到 p 的规模，我们可以重构这张图
- 把每个节点拆成 2^p 个新节点。节点 i_K 表示到了原图的节点 i 时，可消灭的怪兽集合为 K
- 思考连边方法
  - 点 (i, K) 到 $(i, K \cup S_i)$，连边权为 0 的有向边
  - 对于点 (i, K) 和 (j, K')，若原图中 i 和 j 相邻，且 K' 是 K 的子集，则连一条边权为原边权的有向边
- 至此，化归为一般的最短路问题，丢到 Dijsktra 里面即可
- 时间复杂度为 $O(n \cdot 2^p + m \cdot 2^p \cdot log(n \cdot 2^p))$

例题 IX

- 给出一张 n 个点、$n-1$ 条边的连通无向图，节点 i 权值为 w_i，每条边的长度均为 1。若点对 (u,v) 的距离为 2，则它们的联合权值定义为 $w_u \times w_v$；其它点对均不能产生联合权值。问所有能产生联合权值的有序点对中，联合权值最大的是多少？所有联合权值之和是多少？
- $n \le 2 \times 10^5, w_i \le 10000$

例题 IX / NOIP2014 TG Day1 T2

Solution

- 距离为 2 的点对，相当于某个点的两个相邻点
- 预处理出每个点的相邻点中，权值最大的两个，取最大值即可

- 距离为 2 的点对，相当于某个点的两个相邻点
- 预处理出每个点的相邻点中，权值最大的两个，取最大值即可
- 对于权值之和，我们注意到

$$\sum_i (a_i \times \sum_{j \neq i} a_j) = (\sum_i a_i)^2 - \sum_i a_i^2$$

- 即，n 的不同的数两两相乘（不自乘）的和，等于他们和的平方减去平方和
- 预处理出每个点的相邻点的和的平方以及平方和即可

例题 X

- 给出一张 n 个点的无向图，其中有 m 条经济边、k 条商业边，每条边都有边权。现要求从起点 s 走到终点 t，在最多走一次商业边的限制下，问最短路径的路线、长度以及走过的商业边的编号（可以不经过商业边）
- $n \leq 500$，$m, k \leq 1000$

- 商业边是个特殊的存在
- 注意到商业边、经济边的规模都不大，所以可以考虑暴力枚举走哪条商业边
- 相当于已知起点、终点和一条中途边，如何求最短路呢?

例题 X / UVa 11374

Solution

- 商业边是个特殊的存在
- 注意到商业边、经济边的规模都不大，所以可以考虑暴力枚举走哪条商业边
- 相当于已知起点、终点和一条中途边，如何求最短路呢？
- 把中途边的两个端点和起终点分别求最短路，取最优组合即可
- 别忘了求一次只走经济边的最短路
- 时间复杂度为 $O(k \cdot nlogm)$

例题 XI

- 给出一张 n 个点的有向图，把它们从 1 到 n 编号，且各有一个 H 值。每个节点到比自己编号大的各个节点都有一条有向边，边权（距离）为两个节点的 H 值之差的绝对值。各个点的 H 值互不相同。
- 现在，A 和 B 两人要去看看世界，他们约定轮流开车：第一天由 A 驾驶，之后每天轮换一次。
- 他们计划从某个节点 S 开始驾驶。A 总是选择第二近的节点作为目的地，而 B 总是选择第二近的节点（若有同样近的节点，则以海拔低者为更近）
- 只要他们中任意一人无法按照自己的原则选择目的地，或者行驶到目的地就会使总里程数超过 X，他们就会在当前节点结束旅行。

例题 XI

- 现在有两个问题：
  - 对于一个给定的 $X = X_0$，从哪一个节点出发，A 开车行驶的路程总数与 B 行驶的路程总数的比值最小（暂不考虑 B 未驾驶的情形）。若有多个最优解，输出 H 值最高的节点编号。
  - 对任意给定的 $X = X_i$ 和出发城市 S_i，求小 A 开车行驶的路程总数以及小 B 行驶的路程总数。这样的 X_i 和 S_i 共有 m 组
- 数据范围
  - 对于 30% 的数据，有 $1 \le n \le 20, 1 \le m \le 20$
  - 对于 40% 的数据，有 $1 \le n \le 100, 1 \le m \le 100$
  - 对于 50% 的数据，有 $1 \le n \le 100, 1 \le m \le 1000$
  - 对于 70% 的数据，有 $1 \le n \le 1000, 1 \le m \le 10000$
  - 对于 100% 的数据，有 $1 \le n \le 100000, 1 \le m \le 10000$

例题 XI / NOIP2012 TG Day2 T3

Solution

- 首先建图。预处理出在每个点，A 和 B 的目的地各是什么
- 注意到这是一个 DAG，我们可以仿照树上 LCA 的方法，对于每个点，都倍增预处理从它出发、2^k 个轮回后，两人所处的位置和各自的行驶路程

- 首先建图。预处理出在每个点，A 和 B 的目的地各是什么
- 注意到这是一个 DAG，我们可以仿照树上 LCA 的方法，对于每个点，都倍增预处理从它出发、2^k 个轮回后，两人所处的位置和各自的行驶路程
- 对于第一问，暴力枚举起点，模拟开车过程，取最优解即可

- 首先建图。预处理出在每个点，A 和 B 的目的地各是什么
- 注意到这是一个 DAG，我们可以仿照树上 LCA 的方法，对于每个点，都倍增预处理从它出发、2^k 个轮回后，两人所处的位置和各自的行驶路程
- 对于第一问，暴力枚举起点，模拟开车过程，取最优解即可
- 对于第二问，利用预处理的信息依次求出各自的里程即可

例题 XII

- 给出一张 n 个点 m 条边的有向图，每条边有一个正整数边权 c_i。现在请找一个环，它的边权和与边数的比值是图中所有环中最小的。输出这个比值，结果保留两位小数
- $n \le 50, c_i \le 10^7$

- 常用的转化思想：最优化问题转化为判断问题
- 能不能快速判断是否存在比值不大于 x 的环呢?
- 推一推式子

例题 XII / UVa11090

Solution

- 常用的转化思想：最优化问题转化为判断问题
- 能不能快速判断是否存在比值不大于 x 的环呢?
- 推一推式子
- 把所有边减去 x 之后，检查是否存在负圈即可
- 二分答案，复杂度可以承受

例题 XIII

- 给出一张 n 个点 m 条边的有向图（无重边和自环），求从 1 号点到 n 号点的长度在 d 和 $d+K$ 之间（含端点）的路径数目。其中 K 为已知常数，d 为 1 号点到 n 号点的最短路径长度。答案对 P 取模；如果有无数多条合法路线，输出-1

例题 XV

【数据规模与约定】

对于不同的测试点，我们约定各种参数的规模**不会超过**如下

测试点编号	T	N	M	K	是否有 0 边
1	5	5	10	0	否
2	5	1000	2000	0	否
3	5	1000	2000	50	否
4	5	1000	2000	50	否
5	5	1000	2000	50	否
6	5	1000	2000	50	是
7	5	100000	200000	0	否
8	3	100000	200000	50	否
9	3	100000	200000	50	是
10	3	100000	200000	50	是

对于 100%的数据，$1 \le P \le 10^9, 1 \le a_i, b_i \le N, 0 \le c_i \le 1000$。

数据保证：至少存在一条合法的路线。

- 先考虑不存在零边的情形
- 肯定要先跑一遍最短路，把最短路图提出来
- 注意到 K 很小，可以考虑从它下手进行 DP
- 设 $f[i][j]$ 表示到节点 i、多走 j 时的方案数

例题 XIII / NOIP 2017 TG day1 T3

Solution · 70 分做法

- 先考虑不存在零边的情形
- 肯定要先跑一遍最短路，把最短路图提出来
- 注意到 K 很小，可以考虑从它下手进行 DP
- 设 $f[i][j]$ 表示到节点 i、多走 j 时的方案数
- 由于“有向正权图的最短路图一定是个 DAG”，所以 DP 不会带环，答案也不会是无穷多种
- 直接枚举每条边，刷表即可

例题 XIII / NOIP 2017 TG day1 T3

- 提最短路图、状态表示与 70 分做法相同
- 由于有零边，所以状态转移时可能出现“方向不定的同行转移”，即 DP 可能带环。如何处理?

- 提最短路图、状态表示与 70 分做法相同
- 由于有零边，所以状态转移时可能出现“方向不定的同行转移”，即 DP 可能带环。如何处理?
- 把最短路图按拓扑序更新 f 值

- 提最短路图、状态表示与 70 分做法相同
- 由于有零边，所以状态转移时可能出现“方向不定的同行转移”，即 DP 可能带环。如何处理?
- 把最短路图按拓扑序更新 f 值
- 若有点始终没有被更新，说明存在环，该点置为-1

- 提最短路图、状态表示与 70 分做法相同
- 由于有零边，所以状态转移时可能出现“方向不定的同行转移”，即 DP 可能带环。如何处理?
- 把最短路图按拓扑序更新 f 值
- 若有点始终没有被更新，说明存在环，该点置为-1
- 最后统计所有 $f[n][0..K]$

例题 XIV

- WHX 开发了一个找妹子的软件。不幸的是，里面藏有 n 个 bug（编号从 1 到 n）。每个 bug 都有两个状态：存在（用 1 表示）和被修复（用 0 表示）。
- WHX 急忙开发了 m 个补丁，每个补丁都有一定的使用条件，我们用一个长度为 n 的字符串描述：
  - 若第 i 位为'+'，表示使用前第 i 个 bug 必须存在
  - 若第 i 位为'-'，表示使用前第 i 个 bug 必须被修复
  - 若第 i 位为'0'，表示使用前第 i 个 bug 无论存在与否都不影响该补丁的使用
- 打上补丁后的效果也用一个长度为 n 的字符串描述：
  - 若第 i 位为'+'，表示使用后第 i 个 bug 将被修复
  - 若第 i 位为'-'，表示使用后第 i 个 bug 将存在
  - 若第 i 位为'0'，表示使用后第 i 个 bug 将维持原状
- 每种补丁的修复过程都要消耗一定的用时（各补丁的用时可能不同）。现在要求在最短的时间内修复所有的 bug。输出这个最短时间或无解信息
- $n \leq 20$，$m \leq 100$

- d_i 表示达到状态 i 时，所需最小时间
- 用 Dijkstra 或 SPFA 的思想（顺序）DP 即可

- d_i 表示达到状态 i 时，所需最小时间
- 用 Dijkstra 或 SPFA 的思想（顺序）DP 即可
- 状态数多（2^{20} 个），时间复杂度能承受吗？边如何储存？

- d_i 表示达到状态 i 时，所需最小时间
- 用 Dijkstra 或 SPFA 的思想（顺序）DP 即可
- 状态数多（2^{20} 个），时间复杂度能承受吗？边如何储存？
- 很多状态是达不到的，边也没必要储存

- d_i 表示达到状态 i 时，所需最小时间
- 用 Dijkstra 或 SPFA 的思想（顺序）DP 即可
- 状态数多（2^{20} 个），时间复杂度能承受吗？边如何储存？
- 很多状态是达不到的，边也没必要储存
- 需要遍历从某个节点出发的每条边时，直接枚举所有 m 个补丁，用位运算获得后继状态

EOF

- 谢谢大家 $\cdot \omega \cdot$