图论（一）

李佳衡

实验舱科学辅导中心

2020 年 8 月 10 日

并查集

维护若干个不相交的集合，支持集合合并。
(维护无向图的连通性，支持加边。)

并查集

维护若干个不相交的集合，支持集合合并。
(维护无向图的连通性，支持加边。)

路径压缩

在查找一个结点时将其连接到根。
均摊最坏时间复杂度 $O(\log n)$，平均时间复杂度 $O(\alpha(n))$。

并查集

维护若干个不相交的集合，支持集合合并。
(维护无向图的连通性，支持加边。)

路径压缩

在查找一个结点时将其连接到根。
均摊最坏时间复杂度 $O(\log n)$，平均时间复杂度 $O(\alpha(n))$。

按秩合并

根据估价函数（通常为深度或大小），将较小的连至较大的之下。
时间复杂度 $O(\log n)$。

并查集

维护若干个不相交的集合，支持集合合并。
(维护无向图的连通性，支持加边。)

路径压缩

在查找一个结点时将其连接到根。
均摊最坏时间复杂度 $O(\log n)$，平均时间复杂度 $O(\alpha(n))$。

按秩合并

根据估价函数（通常为深度或大小），将较小的连至较大的之下。
时间复杂度 $O(\log n)$。

- 在同时使用路径压缩与按秩合并时，均摊时间复杂度仅为 $O(\alpha(n))$。
- 如果仅使用按秩合并而不使用路径压缩，则可以支持撤销。

并查集

带权并查集

支持查询每个点到根的距离。

每个点记录到现在父亲的距离，在路径压缩的时候更新即可。

并查集

带权并查集

支持查询每个点到根的距离。

每个点记录到现在父亲的距离，在路径压缩的时候更新即可。

由于问题的特殊性，连边两端不能交换，因此不支持按秩合并，只能路径压缩。

时间复杂度不变。

[BZOJ 1015] 星球大战 starwar

给一张 n 个点、m 条边的无向图，执行 k 次操作。每次操作删除一个点，并求操作后有多少个连通块。

$n \leq 2m$，$m \leq 2 \times 10^5$。

[BZOJ 1015] 星球大战 starwar

给一张 n 个点、m 条边的无向图，执行 k 次操作。每次操作删除一个点，并求操作后有多少个连通块。

$n \leq 2m$，$m \leq 2 \times 10^5$。

解

将所有操作离线，倒过来做，就把删边转化为了加边，用并查集维护即可。

时间复杂度 $O(m \log m)$。

[BZOJ 1050] 旅行 comf

给一张 n 个点、m 条边、有边权的无向图。求 s 和 t 之间的一条路径，使得最大、最小边权的比值最小。路径可能不存在。

$n \le 500$，$m \le 5000$。

[BZOJ 1050] 旅行 comf

给一张 n 个点、m 条边、有边权的无向图。求 s 和 t 之间的一条路径，使得最大、最小边权的比值最小。路径可能不存在。

$n \leq 500$，$m \leq 5000$。

解

将所有边按边权排序，枚举最小边 l，从小到大枚举最大边 r 并用并查集判断 $[l, r]$ 中的所有边能否使 s、t 连通。

时间复杂度 $O(m^2\alpha(n))$。

[LibreOJ 121] 动态图连通性

对于一张 n 个点的无向图，维护 m 次加边、删边、查询两点间连通性的操作。

$n \leq 5000$，$m \leq 5 \times 10^5$。

[LibreOJ 121] 动态图连通性

对于一张 n 个点的无向图，维护 m 次加边、删边、查询两点间连通性的操作。

$n \leq 5000$，$m \leq 5 \times 10^5$。

解

先将所有操作离线，问题变为，每条边有一个存在的时间区间，求某个时刻的连通性。

考虑分治，对于某个分治到的区间，将所有完全包含这个区间的边加入并查集，然后递归左右；在回溯时撤销这些边（需要使用仅按秩合并的并查集）；在单点处查询答案即可。

时间复杂度分析类似线段树，总时间复杂度 $O(m \log m\alpha(n))$。

[BZOJ 4025] 二分图

对于一张 n 个点、m 条边的无向图，每条边在 T 时刻内的某一时刻出现、某一时刻消失。求每个时刻这张图是否为二分图。

$n \le 10^5$，$m \le 2 \times 10^5$，$T \le 10^5$。

[BZOJ 4025] 二分图

对于一张 n 个点、m 条边的无向图，每条边在 T 时刻内的某一时刻出现、某一时刻消失。求每个时刻这张图是否为二分图。

$n \leq 10^5$，$m \leq 2 \times 10^5$，$T \leq 10^5$。

解

思路类似上一题，我们需要用并查集维护一张图是否为二分图。

二分图的充要条件是不存在奇环，因此只要在插入边时查询两点间是否存在长为偶数的路径；如果不存在，则插入这条边也不会改变任意两点间的路径奇偶性（因为所有环都是偶环）。

使用带权并查集维护每个点到根距离的奇偶性，由于走重复的路原路返回不影响奇偶性，可以任意连边，因此可以按秩合并，支持撤销。

时间复杂度 $O(m \log T\alpha(n))$。

拓扑排序

在一个**有向无环图**（Directed Acyclic Graph，DAG）中，将所有点排序，使得对于所有有向边 $u \to v$，都满足 u 排在 v 前面。

拓扑排序

在一个**有向无环图**（Directed Acyclic Graph，DAG）中，将所有点排序，使得对于所有有向边 $u \to v$，都满足 u 排在 v 前面。

用队列维护，将入度为 0 的点入队，每次从队列中取出首位加入答案并删除，然后寻找出边中新的度数为 0 的点入队。

时间复杂度 $O(m)$。

[CodeForces 919D] Substring

给定一张 n 个点、m 条边的有向图，每个点上有一个小写字母。定义一条路径的权值为路径上出现最多的字符的出现次数。求路径权值的最大值。最大值可能不存在。

$n, m \leq 3 \times 10^5$。

[CodeForces 919D] Substring

给定一张 n 个点、m 条边的有向图，每个点上有一个小写字母。定义一条路径的权值为路径上出现最多的字符的出现次数。求路径权值的最大值。最大值可能不存在。

$n, m \leq 3 \times 10^5$。

解

对这个图进行拓扑排序，如果无法完全排序则说明有环，最大值不存在。

否则，在拓扑排序的过程中记录每种字母出现次数的最大值，最后全部取最大值即为答案。

时间复杂度 $O(m)$。

[CodeForces 919D] Substring

给定一张 n 个点、m 条边的有向图，每个点上有一个小写字母。定义一条路径的权值为路径上出现最多的字符的出现次数。求路径权值的最大值。最大值可能不存在。

$n, m \leq 3 \times 10^5$。

解

对这个图进行拓扑排序，如果无法完全排序则说明有环，最大值不存在。

否则，在拓扑排序的过程中记录每种字母出现次数的最大值，最后全部取最大值即为答案。

时间复杂度 $O(m)$。

实际上这就是一个 DP 的过程，也可以用记忆化搜索实现。

[CodeForces 909E] Coprocessor

有 n 个任务，任务间的依赖关系构成了一张 m 个点的有向无环图。一些任务在主处理器上执行，其余任务在协处理器上执行。每次执行可以给处理器发出一组（若干个）任务，要求依赖全部已满足或就在本组中，总共只有 1 的代价。求协处理器代价的最小值。

$n, m \le 10^5$。

[CodeForces 909E] Coprocessor

有 n 个任务，任务间的依赖关系构成了一张 m 个点的有向无环图。一些任务在主处理器上执行，其余任务在协处理器上执行。每次执行可以给处理器发出一组（若干个）任务，要求依赖全部已满足或就在本组中，总共只有 1 的代价。求协处理器代价的最小值。

$n, m \le 10^5$。

解

我们需要将主处理器、协处理器的任务分别维护队列，做拓扑排序。

由于要求最小化协处理器的代价，因此先处理主处理器的队列，清空后再处理协处理器的队列。每次的主、协处理器队列可以分别做为一组处理。如此往复，处理协处理器队列（非空）的次数即为答案。

时间复杂度 $O(n)$。

最短路

宽度优先搜索（Breadth First Search，BFS）

用队列维护所有点，从起点开始，每次更新队首的出边并入队（如果未被更新过）。要求边权全部为 1，由队列“先进先出”的性质每个点只需处理一次。

时间复杂度 $O(m)$。

最短路

宽度优先搜索（Breadth First Search，BFS）

用队列维护所有点，从起点开始，每次更新队首的出边并入队（如果未被更新过）。要求边权全部为 1，由队列“先进先出”的性质每个点只需处理一次。

时间复杂度 $O(m)$。

Floyd 算法

可以求出任意两点间的最短路（要求最短路存在）。

枚举一个中间点 k，再枚举 i 和 j，更新 i、j 间最短路经过 k 的情况。

时间复杂度 $O(n^3)$。

最短路

Dijkstra 算法

每次找到当前距源点距离最小的点，更新它的所有出边。要求边权非负，这个点以后不会再被更新。

时间复杂度 $O(n^2)$；如果使用堆维护，时间复杂度 $O(m\log m)$；如果使用 Fibonacci 堆或配对堆，时间复杂度可以做到 $O(n\log n+m)$。

最短路

Dijkstra 算法

每次找到当前距源点距离最小的点，更新它的所有出边。要求边权非负，这个点以后不会再被更新。

时间复杂度 $O(n^2)$；如果使用堆维护，时间复杂度 $O(m\log m)$；如果使用 Fibonacci 堆或配对堆，时间复杂度可以做到 $O(n\log n+m)$。

事实上，如果使用 ZKW 线段树维护，虽然时间复杂度仍为 $O(m\log m)$，但常数因子优秀，且易于实现。

最短路

Dijkstra 算法

每次找到当前距源点距离最小的点，更新它的所有出边。要求边权非负，这个点以后不会再被更新。

时间复杂度 $O(n^2)$；如果使用堆维护，时间复杂度 $O(m\log m)$；如果使用 Fibonacci 堆或配对堆，时间复杂度可以做到 $O(n\log n+m)$。

事实上，如果使用 ZKW 线段树维护，虽然时间复杂度仍为 $O(m\log m)$，但常数因子优秀，且易于实现。

SPFA（Shortest Path Faster Algorithm）

用队列维护所有点，从起点开始，每次更新队首的出边并入队（如果不在队列中）。如果某个点入队超过 n 次，则说明存在负环，最短路不存在。

由于每个点入队不超过 n 次，时间复杂度 $O(nm)$。

差分约束

有若干个变量 $\{x_i\}$，另有若干个限制要求 $x_v - x_u \leq w$。求 $x_t - x_s$ 的最大值。

差分约束

有若干个变量 $\{x_i\}$，另有若干个限制要求 $x_v - x_u \leq w$。求 $x_t - x_s$ 的最大值。

解

对于每个限制，在图中连一条 $u \to v$、权值为 w 的有向边，问题等价于求 $s \to t$ 的最短路。

差分约束

有若干个变量 $\{x_i\}$，另有若干个限制要求 $x_v - x_u \le w$。求 $x_t - x_s$ 的最大值。

解

对于每个限制，在图中连一条 $u \to v$、权值为 w 的有向边，问题等价于求 $s \to t$ 的最短路。

证明.

最短路满足不等式 $\mathrm{dis}_u - \mathrm{dis}_v \le w_{u \to v}$，否则可以用 u 更新 v，因此最短路 dis 即为一组合法的解。

并且，对于每个 dis_v 都是最大的，归纳可得如果更大的话则会不满足限制。 $\square$

最小环问题

给定一张带边权的有向图，求边权和最小的环长。

最小环问题

给定一张带边权的有向图，求边权和最小的环长。

解

枚举环上一个点 u，追加一个新点 u' 并对于环上每条边 $v \to u$，追加一条新边 $v \to u'$，答案即为 $u \to u'$ 的转移视作一条边，答案即为最短路。

如果使用 Dijkstra 算法每次求出最短路，时间复杂度 $O(n^3)$ 或 $O(nm \log m)$。

最小环问题

给定一张带边权的有向图，求边权和最小的环长。

解

枚举环上一个点 u，追加一个新点 u' 并对于环上每条边 $v \to u$，追加一条新边 $v \to u'$，答案即为 $u \to u'$ 的转移视作一条边，答案即为最短路。

如果使用 Dijkstra 算法每次求出最短路，时间复杂度 $O(n^3)$ 或 $O(nm\log m)$。

实际上，如果使用 Floyd 算法，只需将初始时的 $\mathrm{dis}(u, u)$ 设为 $+\infty$，答案即为最终 $\mathrm{dis}(u, u)$ 的最小值。时间复杂度 $O(n^3)$，易于实现。

最小环问题

给定一张带边权的图，求边权和最小的环长（至少包含 3 条边）。

最小环问题

给定一张带边权的图，求边权和最小的环长（至少包含 3 条边）。

朴素做法

枚举环上的一条边 (u, v)，如果最小环包含这条边，答案应为 $w + \mathrm{dis}(u, v)$，其中 w 表示这条边的长度，dis 表示删去这条边后的距离。

可以使用 Dijkstra 算法每次求出最短路，时间复杂度 $O(n^2 m)$ 或 $O(m^2 \log m)$。

最小环问题

给定一张带边权的图，求边权和最小的环长（至少包含 3 条边）。

朴素做法

枚举环上的一条边 (u, v)，如果最小环包含这条边，答案应为 $w + \mathrm{dis}(u, v)$，其中 w 表示这条边的长度，dis 表示删去这条边后的距离。

可以使用 Dijkstra 算法每次求出最短路，时间复杂度 $O(n^2 m)$ 或 $O(m^2 \log m)$。

Floyd 算法

在一般 Floyd 最外层枚举到点 k 时，当前 dis 只经过了 $[1, k)$ 中的点。

设 k 为最小环上编号最大的点，我们只需枚举与其相邻两点 u、v，答案即为 $\mathrm{dis}(u, v) + w(v \to k) + w(k \to u)$，其中 w 表示初始边权。

时间复杂度 $O(n^3)$。

[BZOJ 1003] 物流运输

给定一张 m 个点的无向图，边有正权，每条边可能在 n 天中的若干个时间段不可用。每天你都需要从 1 号点走到 m 号点；如果相邻两天的路线不同，则需要额外花费 k 的代价。求 n 天总代价的最小值。

$n \leq 100$，$m \leq 20$。

[BZOJ 1003] 物流运输

给定一张 m 个点的无向图，边有正权，每条边可能在 n 天中的若干个时间段不可用。每天你都需要从 1 号点走到 m 号点；如果相邻两天的路线不同，则需要额外花费 k 的代价。求 n 天总代价的最小值。

$n \le 100$，$m \le 20$。

解

考虑 DP，用 f_i 表示前 i 天的最小代价，转移时枚举 j 表示 $[j, i]$ 这段时间路线相同，从 f_{j-1} 转移，额外产生 $k + (j - i + 1)\text{dis}$ 的代价，其中 dis 表示这段时间内两点间的最短路。

时间复杂度 $O(n^2 m^2)$。

[CodeForces 590C] Three States

在 $n \times m$ 的网格中，每个格子可能是障碍、空地或三个国家的领土之一，保证每个国家的领土连通。求最小在多少空地上修路，使得三个国家彼此连通。

$n, m \leq 1000$。

[CodeForces 590C] Three States

在 $n \times m$ 的网格中，每个格子可能是障碍、空地或三个国家的领土之一，保证每个国家的领土连通。求最小在多少空地上修路，使得三个国家彼此连通。

$n, m \leq 1000$。

解

发现最终的情况一定可以表示为一个点向三个国家分别引路径，因此只需枚举这个点，三个国家到这个点的距离和即为答案。

[CodeForces 590C] Three States

在 $n \times m$ 的网格中，每个格子可能是障碍、空地或三个国家的领土之一，保证每个国家的领土连通。求最小在多少空地上修路，使得三个国家彼此连通。

$n, m \leq 1000$。

解

发现最终的情况一定可以表示为一个点向三个国家分别引路径，因此只需枚举这个点，三个国家到这个点的距离和即为答案。

可以用 0-1 BFS 求距离，即枚举起点国家，将这个国家的点全部入队，每次优先走已有的国家领土（代价为 0），然后再更新修路的情况即可。

时间复杂度 $O(nm)$。

[BZOJ 2118] 墨墨的等式

给定非负整数 $a_1, a_2, \ldots, a_n$，求有多少个整数 $b \in [b_{\min}, b_{\max}]$ 使得方程 $a_1 x_1 + a_2 x_2 + \cdots + a_n x_n = b$ 有非负整数解 $x_1, x_2, \ldots, x_n$。

$n \leq 12$，$a_i \leq 5 \times 10^5$，$1 \leq b_{\min} \leq b_{\max} \leq 10^{12}$。

[BZOJ 2118] 墨墨的等式

给定非负整数 $a_1, a_2, \ldots, a_n$，求有多少个整数 $b \in [b_{\min}, b_{\max}]$ 使得方程 $a_1 x_1 + a_2 x_2 + \cdots + a_n x_n = b$ 有非负整数解 $x_1, x_2, \ldots, x_n$。

$n \leq 12$，$a_i \leq 5 \times 10^5$，$1 \leq b_{\min} \leq b_{\max} \leq 10^{12}$。

解

显然如果 $b = b_0$ 有解，则 $b = b_0 + a_1$ 也有解。因此考虑 $b \bmod a_1$ 的值，对于每个 $r \in [0, a_1)$，只需求出 $b \equiv r \pmod{a_1}$ 的最小 b 即可。

对于每个数 a_i 和任意 $x \in [0, a_1)$，我们可以连一条 $x \to (x + a_i) \bmod a_1$、边权为 a_i 的边。答案即为从 0 号点出发的最短路。

令 $m = na_1$，时间复杂度 $O(m \log m)$。

[Luogu P4926] [1007] 倍杀测量者

有 n 个变量 $x_1, x_2, \ldots, x_n$，其中 t 个是已知的。有 s 个假设，每个假设形如 $x_u < (k - T)x_v$ 或 $x_v \geq (k + T)x_u$，其中 u、v、k 给定。求最大的 T 使得所有假设一定至少有一个成立，可能不存在。

$n, s \leq 1000$，$k \leq 10$，已知变量在 $[1, 10^9]$ 中。输出绝对误差不超过 10^{-4}。

[Luogu P4926] [1007] 倍杀测量者

有 n 个变量 $x_1, x_2, \ldots, x_n$，其中 t 个是已知的。有 s 个假设，每个假设形如 $x_u < (k-T)x_v$ 或 $x_v \geq (k+T)x_u$，其中 u、v、k 给定。求最大的 T 使得所有假设一定至少有一个成立，可能不存在。

$n, s \leq 1000$，$k \leq 10$，已知变量在 $[1, 10^9]$ 中。输出绝对误差不超过 10^{-4}。

解

显然 T 具有可二分性，因此先二分 T，问题变为对于确定的 T，是否存在一种可能使得所有假设都不成立。

对所有变量取对数，限制变形为 $x'_u \geq x'_v + \log(k-T)$ 或 $x'_v \geq x_u + \log(k+T)$，差分约束判断无解即可。

时间复杂度 $O\left(ns \log \frac{k}{\epsilon}\right)$。

[LibreOJ 6009] 软件补丁

软件出现了 n 个错误，需要一些补丁来修复。共有 m 个补丁，每个补丁分别有参数 w 表示所需时间，还有参数集合 B_1、B_2、F_1、F_2，表示当且仅当目前集合 B_1 中的错误全部存在、集合 B_2 中的错误全部不存在时，可以安装这个补丁以修复 F_1 中的错误、引入 F_2 中的错误。求修复全部错误的最小耗时。

$n \leq 20$，$m \leq 100$。

[LibreOJ 6009] 软件补丁

软件出现了 n 个错误，需要一些补丁来修复。共有 m 个补丁，每个补丁分别有参数 w 表示所需时间，还有参数集合 B_1、B_2、F_1、F_2，表示当且仅当目前集合 B_1 中的错误全部存在、集合 B_2 中的错误全部不存在时，可以安装这个补丁以修复 F_1 中的错误、引入 F_2 中的错误。求修复全部错误的最小耗时。

$n \le 20$，$m \le 100$。

解

考虑状压 DP，f_S 表示将现有错误集合变为 S 需要的最小耗时，每次枚举一个补丁转移即可。

[LibreOJ 6009] 软件补丁

软件出现了 n 个错误，需要一些补丁来修复。共有 m 个补丁，每个补丁分别有参数 w 表示所需时间，还有参数集合 B_1、B_2、F_1、F_2，表示当且仅当目前集合 B_1 中的错误全部存在、集合 B_2 中的错误全部不存在时，可以安装这个补丁以修复 F_1 中的错误、引入 F_2 中的错误。求修复全部错误的最小耗时。

$n \le 20$，$m \le 100$。

解

考虑状压 DP，f_S 表示将现有错误集合变为 S 需要的最小耗时，每次枚举一个补丁转移即可。

但是我们不能保证转移是有向无环的，因此不能直接 DP。但可以将 $S \to T$、代价为 w 的转移视作一条边，答案即为最短路。

时间复杂度 $O(2^n m \log(2^n m))$，常数因子优秀。

[LibreOJ 6009] 软件补丁

软件出现了 n 个错误，需要一些补丁来修复。共有 m 个补丁，每个补丁分别有参数 w 表示所需时间，还有参数集合 B_1、B_2、F_1、F_2，表示当且仅当目前集合 B_1 中的错误全部存在、集合 B_2 中的错误全部不存在时，可以安装这个补丁以修复 F_1 中的错误、引入 F_2 中的错误。求修复全部错误的最小耗时。

$n \le 20$，$m \le 100$。

解

考虑状压 DP，f_S 表示将现有错误集合变为 S 需要的最小耗时，每次枚举一个补丁转移即可。

但是我们不能保证转移是有向无环的，因此不能直接 DP。但可以将 $S \to T$、代价为 w 的转移视作一条边，答案即为最短路。

时间复杂度 $O(2^n m \log(2^n m))$，常数因子优秀。

- 事实上，所有类似的有后效性的 DP 问题都可以使用最短路解决。

[CodeForces 360E] Levko and Game

给一张 n 个点、$m+k$ 条边、有边权的有向图，其中 k 条边的权值可以在该条边的参数 $[l_i, r_i]$ 内任意指定。给定 s_1、s_2、t，问是否能使 $\mathrm{dis}(s_1, t) < \mathrm{dis}(s_2, t)$；如果不能，问是否能使距离相等。

$n, m \leq 10000$，$k \leq 100$，边权在 $[1, 10^9]$ 内。

[CodeForces 360E] Levko and Game

给一张 n 个点、$m+k$ 条边、有边权的有向图，其中 k 条边的权值可以在该条边的参数 $[l_i, r_i]$ 内任意指定。给定 s_1、s_2、t，问是否能使 $\mathrm{dis}(s_1, t) < \mathrm{dis}(s_2, t)$；如果不能，问是否能使距离相等。

$n, m \le 10000$，$k \le 100$，边权在 $[1, 10^9]$ 内。

解

发现每条特殊边要么选 l_i，要么选 r_i。

先考虑距离小的情况，初始时将边权全部设为 r_i，然后求最短路，将满足 $\mathrm{dis}(s_1, u) < \mathrm{dis}(s_2, u)$ 的边 $u \to v$ 边权设为 l_i，继续求最短路，如此反复直至不存在新的更改。

对于距离相等的情况，做法同上，将判断改为 $\mathrm{dis}(s_1, u) = \mathrm{dis}(s_2, u)$ 即可。

时间复杂度 $O(km \log m)$。

最小生成树

对于一张有边权的无向图，求一个边集的子集，构成一棵树，且权值和最小。

最小生成树

对于一张有边权的无向图，求一个边集的子集，构成一棵树，且权值和最小。

环切性质

对于一棵生成树，如果原图存在一条边与生成树上的一条链构成了一个环，且环的最大边在生成树上，则将该边断开，将原图的新边加入生成树，可以得到更小的生成树。

最小生成树

对于一张有边权的无向图，求一个边集的子集，构成一棵树，且权值和最小。

环切性质

对于一棵生成树，如果原图存在一条边与生成树上的一条链构成了一个环，且环的最大边在生成树上，则将该边断开，将原图的新边加入生成树，可以得到更小的生成树。

- 因此，最小生成树同时还满足权值最大的边权值最小。

最小生成树

Kruskal 算法

将所有边从小到大排序，依次枚举每条边，如果加入该条边后生成树不会成环，则将该条边加入生成树。

需要使用并查集维护图的连通性。

时间复杂度 $O(m \log m)$。

最小生成树

Kruskal 算法

将所有边从小到大排序，依次枚举每条边，如果加入该条边后生成树不会成环，则将该条边加入生成树。

需要使用并查集维护图的连通性。

时间复杂度 $O(m \log m)$。

Prim 算法

每次找到未加入生成树的、与当前生成树相连的边权值最小的点，将该边加入生成树。

时间复杂度同 Dijkstra 算法，为 $O(n^2)$ 或 $O(m \log m)$ 或 $O(m + n \log n)$。

[Luogu P1967] 货车运输

给定一张 n 个点、m 条边的无向图，每条边有一个重量限制。q 次询问，每次求 x、y 之间一次运输的最大重量。

$n < 10000$，$m < 5 \times 10^4$，$q < 3 \times 10^4$，限重不超过 10^5。

[Luogu P1967] 货车运输

给定一张 n 个点、m 条边的无向图，每条边有一个重量限制。q 次询问，每次求 x、y 之间一次运输的最大重量。

$n < 10000$，$m < 5 \times 10^4$，$q < 3 \times 10^4$，限重不超过 10^5。

解

先求出最小生成树，最优路径即为生成树上的路径。

树上路径的最小边权可以通过树上倍增（类似求 LCA 的算法）求出。

时间复杂度 $O(m \log m)$。

[BZOJ 1016] 最小生成树计数

给定一张 n 个点、m 条边、有边权的无向图，求不同的最小生成树的个数。答案对 31011 取模。

$n \leq 100$，$m \leq 1000$，边权不超过 10^9，相同边权的边不超过 10 条。

[BZOJ 1016] 最小生成树计数

给定一张 n 个点、m 条边、有边权的无向图，求不同的最小生成树的个数。答案对 31011 取模。

$n \le 100$，$m \le 1000$，边权不超过 10^9，相同边权的边不超过 10 条。

解

可以证明，所有最小生成树中一种边权出现的次数相同，且对连通性的影响相同。

因此，根据 Kruskal 算法的思想，按顺序枚举每种权值，计算所有尽量多地选边、不成环的方案树，相乘即为答案。

时间复杂度不超过 $O(m2^c + m\log m)$，其中 c 表示每种权值出现次数的最大值。

[BZOJ 1016] 最小生成树计数

给定一张 n 个点、m 条边、有边权的无向图，求不同的最小生成树的个数。答案对 31011 取模。

$n \leq 100$，$m \leq 1000$，边权不超过 10^9，相同边权的边不超过 10 条。

解

可以证明，所有最小生成树中一种边权出现的次数相同，且对连通性的影响相同。

因此，根据 Kruskal 算法的思想，按顺序枚举每种权值，计算所有尽量多地选边、不成环的方案树，相乘即为答案。

时间复杂度不超过 $O(m2^c + m \log m)$，其中 c 表示每种权值出现次数的最大值。

实际上，如果使用 Matrix-Tree 定理计算方案数，时间复杂度也可以做到 $O(mc^2 + m \log m)$。

[BZOJ 1977] 次小生成树 Tree

给一张 n 个点、m 条边、有边权的无向图，求严格次小的生成树的权值和。

$n \leq 10^5$，$m \leq 3 \times 10^5$，边权在 $[0, 10^9]$ 内。

[BZOJ 1977] 次小生成树 Tree

给一张 n 个点、m 条边、有边权的无向图，求严格次小的生成树的权值和。

$n \le 10^5$，$m \le 3 \times 10^5$，边权在 $[0, 10^9]$ 内。

解

可以证明，次小生成树一定可以由最小生成树修改一条边得到。

因此，我们枚举这条新增的非树边，问题变成求最小生成树上两点间路径的最大值，但由于要求严格次小，最大值可能不满足条件，还需求出严格次大值。

类似地，用树上倍增计算即可。

时间复杂度 $O(m \log m)$。