图论杂题选讲

wzj52501

Peking University

2018 年 10 月 29 日

约定

若无特殊说明，所有例题的时间限制均为 $2s$，空间限制均为 $256MB$。

MST

例题 1

货车运输 (NOIP2013)

A 国有 n 座城市，编号从 1 到 n，城市之间有 m 条双向道路。每一条道路对车辆都有重量限制，简称限重。现在有 q 辆货车运输货物，第 i 辆货车将从 x_i 城市运输货物到 y_i 城市。司机们想知道每辆车在不超过车辆限重的情况下，最多能运多重的货物。

$n \leq 10000, m \leq 50000, q \leq 30000$

例题 1

货车运输 (NOIP2013)

A 国有 n 座城市，编号从 1 到 n，城市之间有 m 条双向道路。每一条道路对车辆都有重量限制，简称限重。现在有 q 辆货车运输货物，第 i 辆货车将从 x_i 城市运输货物到 y_i 城市。司机们想知道每辆车在不超过车辆限重的情况下，最多能运多重的货物。

$n \leq 10000, m \leq 50000, q \leq 30000$

- 题目所求即为若干点对间的最大瓶颈路，而使用 Kruskal 算法求出的最大生成树即为最大瓶颈树。
- 故进行 Kruskal 算法求出最大瓶颈树后，使用 ST 表等算法维护树上路径最值即可。
- 时间复杂度为 $O(m \log m + (q+n) \log n)$。

“茴香豆”的四种写法

“茴香豆”的四种写法

1. 倍增 ST 表：$O(n \log n) - O(\log n)$。

“茴香豆”的四种写法

1. 倍增 ST 表：$O(n \log n) - O(\log n)$。
2. 树链剖分：$O(n) - O(\log n)$。

“茴香豆”的四种写法

1. 倍增 ST 表：$O(n \log n) - O(\log n)$。
2. 树链剖分：$O(n) - O(\log n)$。
3. 欧拉序列：$O(n \log n) - O(1)$。

“茴香豆”的四种写法

1. 倍增 ST 表：$O(n \log n) - O(\log n)$。
2. 树链剖分：$O(n) - O(\log n)$。
3. 欧拉序列：$O(n \log n) - O(1)$。
4. Tarjan 算法：$O((n+q)\alpha(n))$。

“茴香豆”的四种写法

1. 倍增 ST 表：$O(n \log n) - O(\log n)$。
2. 树链剖分：$O(n) - O(\log n)$。
3. 欧拉序列：$O(n \log n) - O(1)$。
4. Tarjan 算法：$O((n+q)\alpha(n))$。
5. (长链剖分，四毛子，LCT.....)

例题 2

Tree

给定一个 n 个结点 m 条带权边的无向连通图，对于每一条边求出强制包含该边的最小生成树的权值。

$n, m \leq 10^6$

例题 2

Tree

给定一个 n 个结点 m 条带权边的无向连通图，对于每一条边求出强制包含该边的最小生成树的权值。

$n, m \leq 10^6$

- 先求出原图的最小生成树 T，考虑强制加入边 $e_1 = (u_1, v_1, w_1)$ 后会有怎样的变化。

例题 2

Tree

给定一个 n 个结点 m 条带权边的无向连通图，对于每一条边求出强制包含该边的最小生成树的权值。

$n, m \leq 10^6$

- 先求出原图的最小生成树 T，考虑强制加入边 $e_1 = (u_1, v_1, w_1)$ 后会有怎样的变化。
  1. 若 e_1 已在 T 中：无变化。

例题 2

Tree

给定一个 n 个结点 m 条带权边的无向连通图，对于每一条边求出强制包含该边的最小生成树的权值。
$n, m \leq 10^6$

- 先求出原图的最小生成树 T，考虑强制加入边 $e_1 = (u_1, v_1, w_1)$ 后会有怎样的变化。
  1. 若 e_1 已在 T 中：无变化。
  2. 若 e_1 不在 T 中：问题相当于在最开始先加入边 e_1 再对剩余的边进行 Kruskal 算法，不难发现这对于 T 上不属于 u_1 到 v_1 路径上的边没有任何影响，而且必定存在边 $e_2 = (u_2, v_2, w_2)$ 满足 e_2 在 u_1 到 v_1 路径上，且 $w_2 \leq w_1$，使得求 T 过程中加入 e_2 后 u_1 与 v_1 连通。这样的 e_2 肯定在 Kruskal 算法中是 u_1 到 v_1 路径上最后加入 T 的，即 T 中 u_1 到 v_1 路径上权值最大的边，用 e_1 替换 e_2 即为新的最小生成树。

例题 2

Tree

给定一个 n 个结点 m 条带权边的无向连通图，对于每一条边求出强制包含该边的最小生成树的权值。
$n, m \leq 10^6$

- 先求出原图的最小生成树 T，考虑强制加入边 $e_1 = (u_1, v_1, w_1)$ 后会有怎样的变化。
  1. 若 e_1 已在 T 中：无变化。
  2. 若 e_1 不在 T 中：问题相当于在最开始先加入边 e_1 再对剩余的边进行 Kruskal 算法，不难发现这对于 T 上不属于 u_1 到 v_1 路径上的边没有任何影响，而且必定存在边 $e_2 = (u_2, v_2, w_2)$ 满足 e_2 在 u_1 到 v_1 路径上，且 $w_2 \leq w_1$，使得求 T 过程中加入 e_2 后 u_1 与 v_1 连通。这样的 e_2 肯定在 Kruskal 算法中是 u_1 到 v_1 路径上最后加入 T 的，即 T 中 u_1 到 v_1 路径上权值最大的边，用 e_1 替换 e_2 即为新的最小生成树。
- 问题再次转化成了多次询问树上路径最值问题。
- 时间复杂度为 $O(m \log m + n \log n)$。

例题 3

Phone Call (2017 Multi-University Training Contest)

给定一棵 n 个结点的无根树，你需要根据 m 条线路重构一棵生成树。设 $S(u, v)$ 表示包含树上 u 到 v 路径上的所有结点的集合，其中第 i 条线路可以用五元组 $(a_i, b_i, c_i, d_i, w_i)$ 来表达：

- 你可以任意在 $S(a_i, b_i) \cup S(c_i, d_i)$ 的结点之间添加边，每一条的费用都是 w_i。

求总费用之和最小的方案，保证题目有解。

$n, m \leq 10^5$

- 我们考虑沿用 Kruskal 算法，先将所有线路按费用从小到大排序，然后逐个考虑每条线路。

- 我们考虑沿用 Kruskal 算法，先将所有线路按费用从小到大排序，然后逐个考虑每条线路。
- 当考虑到第 i 条线路时，我们的目标是让 $S(a_i, b_i) \cup S(c_i, d_i)$ 的所有结点彼此连通。

- 我们考虑沿用 Kruskal 算法，先将所有线路按费用从小到大排序，然后逐个考虑每条线路。
- 当考虑到第 i 条线路时，我们的目标是让 $S(a_i, b_i) \cup S(c_i, d_i)$ 的所有结点彼此连通。
- 我们可以将其拆成 5 组独立的集合，分别为 $S(a_i, lca(a_i, b_i))$、$S(b_i, lca(a_i, b_i))$、$S(c_i, lca(c_i, d_i))$、$S(d_i, lca(c_i, d_i))$、$\{a_i, c_i\}$，只要使得这 5 组集合分别内部连通就可以达成目标。

- 我们考虑沿用 Kruskal 算法，先将所有线路按费用从小到大排序，然后逐个考虑每条线路。
- 当考虑到第 i 条线路时，我们的目标是让 $S(a_i, b_i) \cup S(c_i, d_i)$ 的所有结点彼此连通。
- 我们可以将其拆成 5 组独立的集合，分别为 $S(a_i, lca(a_i, b_i))$、$S(b_i, lca(a_i, b_i))$、$S(c_i, lca(c_i, d_i))$、$S(d_i, lca(c_i, d_i))$、$\{a_i, c_i\}$，只要使得这 5 组集合分别内部连通就可以达成目标。
- 经过一些观察我们发现前 4 组的集合都是一段树上的“直链”，当我们要合并 $S(a, b)$ 时，若存在 $S(c, d)$ 已经被合并，我们就可以直接跳过这些结点。

- 我们考虑沿用 Kruskal 算法，先将所有线路按费用从小到大排序，然后逐个考虑每条线路。
- 当考虑到第 i 条线路时，我们的目标是让 $S(a_i, b_i) \cup S(c_i, d_i)$ 的所有结点彼此连通。
- 我们可以将其拆成 5 组独立的集合，分别为 $S(a_i, lca(a_i, b_i))$、$S(b_i, lca(a_i, b_i))$、$S(c_i, lca(c_i, d_i))$、$S(d_i, lca(c_i, d_i))$、$\{a_i, c_i\}$，只要使得这 5 组集合分别内部连通就可以达成目标。
- 经过一些观察我们发现前 4 组的集合都是一段树上的“直链”，当我们要合并 $S(a, b)$ 时，若存在 $S(c, d)$ 已经被合并，我们就可以直接跳过这些结点。
- 我们使用一个并查集来维护这些“直链”，设 pa_i 表示 i 号结点向上最远的祖先 c 满足 $S(c, i)$ 包含的所有结点均已被合并，初始将所有 pa_i 均设为 i。

- 我们考虑沿用 Kruskal 算法，先将所有线路按费用从小到大排序，然后逐个考虑每条线路。
- 当考虑到第 i 条线路时，我们的目标是让 $S(a_i, b_i) \cup S(c_i, d_i)$ 的所有结点彼此连通。
- 我们可以将其拆成 5 组独立的集合，分别为 $S(a_i, lca(a_i, b_i))$、$S(b_i, lca(a_i, b_i))$、$S(c_i, lca(c_i, d_i))$、$S(d_i, lca(c_i, d_i))$、$\{a_i, c_i\}$，只要使得这 5 组集合分别内部连通就可以达成目标。
- 经过一些观察我们发现前 4 组的集合都是一段树上的“直链”，当我们要合并 $S(a, b)$ 时，若存在 $S(c, d)$ 已经被合并，我们就可以直接跳过这些结点。
- 我们使用一个并查集来维护这些“直链”，设 pa_i 表示 i 号结点向上最远的祖先 c 满足 $S(c, i)$ 包含的所有结点均已被合并，初始将所有 pa_i 均设为 i。
- 这样我们在合并直链时就可以从底向上依次合并，借助 pa_i 快速找到所有需要合并的“父子关系”，并同时更新 pa_i。

- 我们考虑沿用 Kruskal 算法，先将所有线路按费用从小到大排序，然后逐个考虑每条线路。
- 当考虑到第 i 条线路时，我们的目标是让 $S(a_i, b_i) \cup S(c_i, d_i)$ 的所有结点彼此连通。
- 我们可以将其拆成 5 组独立的集合，分别为 $S(a_i, lca(a_i, b_i))$、$S(b_i, lca(a_i, b_i))$、$S(c_i, lca(c_i, d_i))$、$S(d_i, lca(c_i, d_i))$、$\{a_i, c_i\}$，只要使得这 5 组集合分别内部连通就可以达成目标。
- 经过一些观察我们发现前 4 组的集合都是一段树上的“直链”，当我们要合并 $S(a, b)$ 时，若存在 $S(c, d)$ 已经被合并，我们就可以直接跳过这些结点。
- 我们使用一个并查集来维护这些“直链”，设 pa_i 表示 i 号结点向上最远的祖先 c 满足 $S(c, i)$ 包含的所有结点均已被合并，初始将所有 pa_i 均设为 i。
- 这样我们在合并直链时就可以从底向上依次合并，借助 pa_i 快速找到所有需要合并的“父子关系”，并同时更新 pa_i。
- 由于每个结点只会参与 1 次合并，时间复杂度为 $O((n+m)\log n)$。

例题 4

XOR

给定一个 n 阶完全无向图，每个节点有权值 v_i。
对于任意结点 x、y，从 x 到 y 的无向边权值为 $v_x \bigoplus v_y$，求最小生成树。
$n \leq 10^5$，$0 \leq v_i \leq 10^9$。

例题 4

XOR

给定一个 n 阶完全无向图，每个节点有权值 v_i。
对于任意结点 x、y，从 x 到 y 的无向边权值为 $v_x \bigoplus v_y$，求最小生成树。
$n \leq 10^5$，$0 \leq v_i \leq 10^9$。

这种特殊的位运算生成树问题，一般需要用到最小生成树的第三种算法——Boruvka 算法。

Boruvka 算法

- Step1：对于当前每个连通块，找到其连出的权值最小的出边。
- Step2：将所有这样的出边加入最小生成树（需要去重），并合并相应的连通块。
- Step3：重复 Step1、Step2 直到仅剩 1 个连通块。由于每轮连通块个数至少减半，最多进行 $O(\log n)$ 轮。
- 时间复杂度为 $O(m \log n)$。

Boruvka 算法

- Step1：对于当前每个连通块，找到其连出的权值最小的出边。
- Step2：将所有这样的出边加入最小生成树（需要去重），并合并相应的连通块。
- Step3：重复 Step1、Step2 直到仅剩 1 个连通块。由于每轮连通块个数至少减半，最多进行 $O(\log n)$ 轮。
- 时间复杂度为 $O(m \log n)$。
- 那么我们现在面临的问题就是这样：有 n 个数，每个数有一个颜色，对于每个数 a 求出一个异色的数 b，使得 $a \oplus b$ 最小。

Boruvka 算法

- Step1：对于当前每个连通块，找到其连出的权值最小的出边。
- Step2：将所有这样的出边加入最小生成树（需要去重），并合并相应的连通块。
- Step3：重复 Step1、Step2 直到仅剩 1 个连通块。由于每轮连通块个数至少减半，最多进行 $O(\log n)$ 轮。
- 时间复杂度为 $O(m \log n)$。
- 那么我们现在面临的问题就是这样：有 n 个数，每个数有一个颜色，对于每个数 a 求出一个异色的数 b，使得 $a \oplus b$ 最小。
- 对每个数按二进制从大到小建立 Trie 树，维护一下子树内包含颜色的最大最小值就可以知道是否存在被询问颜色外的其他颜色，从而轻松求出答案。

Boruvka 算法

- Step1：对于当前每个连通块，找到其连出的权值最小的出边。
- Step2：将所有这样的出边加入最小生成树（需要去重），并合并相应的连通块。
- Step3：重复 Step1、Step2 直到仅剩 1 个连通块。由于每轮连通块个数至少减半，最多进行 $O(\log n)$ 轮。
- 时间复杂度为 $O(m \log n)$。
- 那么我们现在面临的问题就是这样：有 n 个数，每个数有一个颜色，对于每个数 a 求出一个异色的数 b，使得 $a \oplus b$ 最小。
- 对每个数按二进制从大到小建立 Trie 树，维护一下子树内包含颜色的最大最小值就可以知道是否存在被询问颜色外的其他颜色，从而轻松求出答案。

总时间复杂度为 $O(n \log n \log v)$。

最短路

例题 5

开车旅行

某地区的地图可以看成是一个 n 个点 m 条带权边的无向图，其中有 k 个点是加油站，而汽车在加油站可以把油箱加满。已知 x 单位的油箱在不加油的情况下最多能走 x 单位的路程。
现在给定 q 组询问，每次询问能否开一辆油箱容量为 v 单位的汽车从点 x 走到点 y。
$k, n, q \leq 200000$，保证所有点 x、y 均为加油站。

例题 5

开车旅行

某地区的地图可以看成是一个 n 个点 m 条带权边的无向图，其中有 k 个点是加油站，而汽车在加油站可以把油箱加满。已知 x 单位的油箱在不加油的情况下最多能走 x 单位的路程。
现在给定 q 组询问，每次询问能否开一辆油箱容量为 v 单位的汽车从点 x 走到点 y。
$k, n, q \leq 200000$，保证所有点 x、y 均为加油站。

- 我们如果能求出加油站之间的最小生成树，问题就转化成了多次询问最小瓶颈路问题。

例题 5

开车旅行

某地区的地图可以看成是一个 n 个点 m 条带权边的无向图，其中有 k 个点是加油站，而汽车在加油站可以把油箱加满。已知 x 单位的油箱在不加油的情况下最多能走 x 单位的路程。
现在给定 q 组询问，每次询问能否开一辆油箱容量为 v 单位的汽车从点 x 走到点 y。
$k, n, q \leq 200000$，保证所有点 x、y 均为加油站。

- 我们如果能求出加油站之间的最小生成树，问题就转化成了多次询问最小瓶颈路问题。
- 一种可行的做法是把每个加油站均作为起点计算多源最短路，并同时记录每个点距离最近的加油站编号。

例题 5

开车旅行

某地区的地图可以看成是一个 n 个点 m 条带权边的无向图，其中有 k 个点是加油站，而汽车在加油站可以把油箱加满。已知 x 单位的油箱在不加油的情况下最多能走 x 单位的路程。
现在给定 q 组询问，每次询问能否开一辆油箱容量为 v 单位的汽车从点 x 走到点 y。
$k, n, q \leq 200000$，保证所有点 x、y 均为加油站。

- 我们如果能求出加油站之间的最小生成树，问题就转化成了多次询问最小瓶颈路问题。
- 一种可行的做法是把每个加油站均作为起点计算多源最短路，并同时记录每个点距离最近的加油站编号。
- 然后遍历每条边 (u, v, w)，如果两端点距离最近的加油站编号不同，则向这两个编号的加油站之间连一条长度为 $dist_u + dist_v + w$ 的边，然后在得到的新图上求出最小生成树即可。

例题 5

开车旅行

某地区的地图可以看成是一个 n 个点 m 条带权边的无向图，其中有 k 个点是加油站，而汽车在加油站可以把油箱加满。已知 x 单位的油箱在不加油的情况下最多能走 x 单位的路程。
现在给定 q 组询问，每次询问能否开一辆油箱容量为 v 单位的汽车从点 x 走到点 y。
$k, n, q \leq 200000$，保证所有点 x、y 均为加油站。

- 我们如果能求出加油站之间的最小生成树，问题就转化成了多次询问最小瓶颈路问题。
- 一种可行的做法是把每个加油站均作为起点计算多源最短路，并同时记录每个点距离最近的加油站编号。
- 然后遍历每条边 (u, v, w)，如果两端点距离最近的加油站编号不同，则向这两个编号的加油站之间连一条长度为 $dist_u + dist_v + w$ 的边，然后在得到的新图上求出最小生成树即可。
- 时间复杂度为 $O((n + m + q) \log n)$。

证明

- 考虑从加油站 x 出发可行的路线，肯定是走最短路径到达另一个加油站 y。

证明

- 考虑从加油站 x 出发可行的路线，肯定是走最短路径到达另一个加油站 y。
- 若 x 到 y 路径上存在一条边 (u, v, w) 满足距离 u 最近的加油站为 x 且距离 v 最近的加油站为 y，那么我们会将 $(x, y, dist(x, y))$ 加入生成树。

证明

- 考虑从加油站 x 出发可行的路线，肯定是走最短路径到达另一个加油站 y。
- 若 x 到 y 路径上存在一条边 (u, v, w) 满足距离 u 最近的加油站为 x 且距离 v 最近的加油站为 y，那么我们会将 $(x, y, dist(x, y))$ 加入生成树。
- 否则我们可以证明 x 到 y 的路径不可能在最小生成树中出现：

证明

- 考虑从加油站 x 出发可行的路线，肯定是走最短路径到达另一个加油站 y。
- 若 x 到 y 路径上存在一条边 (u, v, w) 满足距离 u 最近的加油站为 x 且距离 v 最近的加油站为 y，那么我们会将 $(x, y, dist(x, y))$ 加入生成树。
- 否则我们可以证明 x 到 y 的路径不可能在最小生成树中出现：
  1. 找到一条边 (u, v, w)，设距离 u 最近的加油站为 a，距离 v 最近的加油站为 b，u 距离 x 近，v 距离 y 近。

证明

- 考虑从加油站 x 出发可行的路线，肯定是走最短路径到达另一个加油站 y。
- 若 x 到 y 路径上存在一条边 (u, v, w) 满足距离 u 最近的加油站为 x 且距离 v 最近的加油站为 y，那么我们会将 $(x, y, dist(x, y))$ 加入生成树。
- 否则我们可以证明 x 到 y 的路径不可能在最小生成树中出现：
  1. 找到一条边 (u, v, w)，设距离 u 最近的加油站为 a，距离 v 最近的加油站为 b，u 距离 x 近，v 距离 y 近。
  2. 由条件可得 $dist(a, u) \leq dist(x, u)$，$dist(b, v) \leq dist(y, v)$。

证明

- 考虑从加油站 x 出发可行的路线，肯定是走最短路径到达另一个加油站 y。
- 若 x 到 y 路径上存在一条边 (u, v, w) 满足距离 u 最近的加油站为 x 且距离 v 最近的加油站为 y，那么我们会将 $(x, y, dist(x, y))$ 加入生成树。
- 否则我们可以证明 x 到 y 的路径不可能在最小生成树中出现：
  1. 找到一条边 (u, v, w)，设距离 u 最近的加油站为 a，距离 v 最近的加油站为 b，u 距离 x 近，v 距离 y 近。
  2. 由条件可得 $dist(a, u) \leq dist(x, u)$，$dist(b, v) \leq dist(y, v)$。
  3. 因为 $dist(a, b) = dist(a, u) + dist(v, b) + w \leq dist(x, b) = dist(x, u) + dist(v, b) + w, dist(a, y) = dist(a, u) + dist(v, y) + w \leq dist(x, y) = dist(x, u) + dist(v, y) + w$，
  4. 所以直接从 x 到 y 不会比从 x 到 b 再从 b 到 a 最后从 a 到 y 优。

证明

- 考虑从加油站 x 出发可行的路线，肯定是走最短路径到达另一个加油站 y。
- 若 x 到 y 路径上存在一条边 (u, v, w) 满足距离 u 最近的加油站为 x 且距离 v 最近的加油站为 y，那么我们会将 $(x, y, dist(x, y))$ 加入生成树。
- 否则我们可以证明 x 到 y 的路径不可能在最小生成树中出现：
  1. 找到一条边 (u, v, w)，设距离 u 最近的加油站为 a，距离 v 最近的加油站为 b，u 距离 x 近，v 距离 y 近。
  2. 由条件可得 $dist(a, u) \leq dist(x, u)$，$dist(b, v) \leq dist(y, v)$。
  3. 因为 $dist(a, b) = dist(a, u) + dist(v, b) + w \leq dist(x, b) = dist(x, u) + dist(v, b) + w, dist(a, y) = dist(a, u) + dist(v, y) + w \leq dist(x, y) = dist(x, u) + dist(v, y) + w$，
  4. 所以直接从 x 到 y 不会比从 x 到 b 再从 b 到 a 最后从 a 到 y 优。
- 证毕。

例题 6

图图的旅行

图图计划去 Bzeroth 的精灵王国去旅游，精灵王国由 n 座城市组成，第 i 座城市有 3 个属性 x_i, w_i, t_i。
在精灵王国的城市之间穿行只能依靠传送阵，第 i 座城市的传送阵可以将图图从城市 i 传送到距离城市 i 不超过 w_i 的任意一个城市，并需要 t_i 的时间完成传送。现在图图知道了每个城市的坐标 x_i，想知道他从城市 s 到城市 t 的最小时间。
$1 \leq s, t \leq n \leq 152501, 0 \leq w_i, t_i, |x_i| \leq 10^9$，保证 x_i 严格递增。

- 我们注意到每个城市通过传送阵能够到达的城市是一段连续的区间，设为 $[l_i, r_i]$，这很容易用二分求出。

cont'd

- 我们注意到每个城市通过传送阵能够到达的城市是一段连续的区间，设为 $[l_i, r_i]$，这很容易用二分求出。
- 回忆线段树的结构，本题的连边方式与线段树区间查询的过程很相似，考虑将线段树的结构嵌入到最短路算法的建图中。

- 我们注意到每个城市通过传送阵能够到达的城市是一段连续的区间，设为 $[l_i, r_i]$，这很容易用二分求出。
- 回忆线段树的结构，本题的连边方式与线段树区间查询的过程很相似，考虑将线段树的结构嵌入到最短路算法的建图中。
- 我们在图中新建 $2n-1$ 个线段树结点，由父亲向左右两个儿子分别连一条长度为 0 的边，并从每个叶结点向对应的城市连一条长度为 0 的边。

- 我们注意到每个城市通过传送阵能够到达的城市是一段连续的区间，设为 $[l_i, r_i]$，这很容易用二分求出。
- 回忆线段树的结构，本题的连边方式与线段树区间查询的过程很相似，考虑将线段树的结构嵌入到最短路算法的建图中。
- 我们在图中新建 $2n-1$ 个线段树结点，由父亲向左右两个儿子分别连一条长度为 0 的边，并从每个叶结点向对应的城市连一条长度为 0 的边。
- 然后对于每个城市 i，在线段树上将 $[l_i, r_i]$ 分解成 $O(\log n)$ 个区间，并从城市 i 向这些区间分别连一条长度为 t_i 的边。

- 我们注意到每个城市通过传送阵能够到达的城市是一段连续的区间，设为 $[l_i, r_i]$，这很容易用二分求出。
- 回忆线段树的结构，本题的连边方式与线段树区间查询的过程很相似，考虑将线段树的结构嵌入到最短路算法的建图中。
- 我们在图中新建 $2n-1$ 个线段树结点，由父亲向左右两个儿子分别连一条长度为 0 的边，并从每个叶结点向对应的城市连一条长度为 0 的边。
- 然后对于每个城市 i，在线段树上将 $[l_i, r_i]$ 分解成 $O(\log n)$ 个区间，并从城市 i 向这些区间分别连一条长度为 t_i 的边。
- 这样我们就把边数从 $O(n^2)$ 降至了 $O(n \log n)$！

- 我们注意到每个城市通过传送阵能够到达的城市是一段连续的区间，设为 $[l_i, r_i]$，这很容易用二分求出。
- 回忆线段树的结构，本题的连边方式与线段树区间查询的过程很相似，考虑将线段树的结构嵌入到最短路算法的建图中。
- 我们在图中新建 $2n-1$ 个线段树结点，由父亲向左右两个儿子分别连一条长度为 0 的边，并从每个叶结点向对应的城市连一条长度为 0 的边。
- 然后对于每个城市 i，在线段树上将 $[l_i, r_i]$ 分解成 $O(\log n)$ 个区间，并从城市 i 向这些区间分别连一条长度为 t_i 的边。
- 这样我们就把边数从 $O(n^2)$ 降至了 $O(n \log n)$！
- 在新得到的图上运行堆优化的 Dijkstra 算法，时间复杂度为 $O(n \log^2 n)$ 或 $O(n \log n)$。

例题 7

网格

给定一个 $n \times m$ 的网格，每个网格有一个颜色，只可能是红绿黄蓝四种之一。
现在你需要找到一条从左上角 $(1,1)$ 走到右下角 (n,m) 的总费用最小的路线，每次只能向上下左右走一步，且不能走出边界。
设你的路线中一共经过了 x_1 个红格子、x_2 个绿格子、x_3 个黄格子、x_4 个蓝格子，则你的路线总费用为 $x_1 \times a + x_2 \times b + x_3 \times c + x_4 \times d$。
$1 \leq n, m \leq 2501$，$1 \leq a, b, c, d \leq 10^9$。

cont'd

- 若直接使用堆优化的 Dijkstra 算法，时间复杂度为 $O(nm \log nm)$，无法在时间限制内通过。

- 若直接使用堆优化的 Dijkstra 算法，时间复杂度为 $O(nm \log nm)$，无法在时间限制内通过。
- 注意本题的边权种类只有 4 种，而对于入边边权相同的结点（若 $dist_{v_1} = dist_{u_1} + w$ 而 $dist_{v_2} = dist_{u_2} + w$，我们认为 v_1 和 v_2 是入边边权相同的结点），它们入堆时的 $dist_v$ 也是随时间单调递增的。

- 若直接使用堆优化的 Dijkstra 算法，时间复杂度为 $O(nm \log nm)$，无法在时间限制内通过。
- 注意本题的边权种类只有 4 种，而对于入边边权相同的结点（若 $dist_{v_1} = dist_{u_1} + w$ 而 $dist_{v_2} = dist_{u_2} + w$，我们认为 v_1 和 v_2 是入边边权相同的结点），它们入堆时的 $dist_v$ 也是随时间单调递增的。
- 故我们可以对于每种入边边权相同的结点分别维护一个队列，每次用最多 4 个队首更新当前 $dist$ 最小的结点 u，就可以代替原来堆的作用。

- 若直接使用堆优化的 Dijkstra 算法，时间复杂度为 $O(nm \log nm)$，无法在时间限制内通过。
- 注意本题的边权种类只有 4 种，而对于入边边权相同的结点（若 $dist_{v_1} = dist_{u_1} + w$ 而 $dist_{v_2} = dist_{u_2} + w$，我们认为 v_1 和 v_2 是入边边权相同的结点），它们入堆时的 $dist_v$ 也是随时间单调递增的。
- 故我们可以对于每种入边边权相同的结点分别维护一个队列，每次用最多 4 个队首更新当前 $dist$ 最小的结点 u，就可以代替原来堆的作用。
- 时间复杂度为 $O(nm)$。

例题 8

最短套路

给定一个包含 n 个点的单向完全图，已知所有边的长度。设 $D(i,j,k)$ 表示不能经过编号为 k 的点，从 i 到 j 的最短路的长度。特别地，当 $k=i$ 或 $k=j$ 时，$D(i,j,k)=0$。
你需要求出

$$ans=\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}D(i,j,k) \tag{1}$$

$1\le n\le 200$，$0\le w\le 10^9$。

cont'd

- 回忆 Floyd 算法，它的本质是动态规划。dp 数组记录的是经过前若干个中转点每两个点之间的距离，我们发现中转点的顺序是可以随意决定的。

- 回忆 Floyd 算法，它的本质是动态规划。dp 数组记录的是经过前若干个中转点每两个点之间的距离，我们发现中转点的顺序是可以随意决定的。
- 因为每个中转点出现的时间是两段区间，考虑使用分治算法：当分治到 $[l, r] \quad (l < r)$ 区间时，先用 $[l, mid]$ 的中转点更新 dp 数组，分治 $[mid + 1, r]$ 区间后回溯；再用 $[mid + 1, r]$ 的中转点更新 dp 数组，分治 $[l, mid]$ 区间后回溯。

- 回忆 Floyd 算法，它的本质是动态规划。dp 数组记录的是经过前若干个中转点每两个点之间的距离，我们发现中转点的顺序是可以随意决定的。
- 因为每个中转点出现的时间是两段区间，考虑使用分治算法：当分治到 $[l, r] \quad (l < r)$ 区间时，先用 $[l, mid]$ 的中转点更新 dp 数组，分治 $[mid + 1, r]$ 区间后回溯；再用 $[mid + 1, r]$ 的中转点更新 dp 数组，分治 $[l, mid]$ 区间后回溯。
- 当分治到叶结点 $[k, k]$ 时，$dp(i, j) = D(i, j, k)$。

- 回忆 Floyd 算法，它的本质是动态规划。dp 数组记录的是经过前若干个中转点每两个点之间的距离，我们发现中转点的顺序是可以随意决定的。
- 因为每个中转点出现的时间是两段区间，考虑使用分治算法：当分治到 $[l, r] \quad (l < r)$ 区间时，先用 $[l, mid]$ 的中转点更新 dp 数组，分治 $[mid + 1, r]$ 区间后回溯；再用 $[mid + 1, r]$ 的中转点更新 dp 数组，分治 $[l, mid]$ 区间后回溯。
- 当分治到叶结点 $[k, k]$ 时，$dp(i, j) = D(i, j, k)$。
- 时间复杂度为 $O(n^3 \log n)$。

网络流

例题 9

子序列

给定一个长度为 n 的序列 A，你需要从中挑选一个非空子序列（不必连续），使得这个子序列的逆序对数/长度尽量大。
$1 \leq n \leq 200$，$1 \leq A_i \leq 10^9$。

cont'd

- 其实就是最大密度子图模型，二分答案后最小割判定即可。

cont'd

- 其实就是最大密度子图模型，二分答案后最小割判定即可。
- 时间复杂度为 $O(MaxFlow(n^2, n^2) \times \log n)$。

cont'd

- 其实就是最大密度子图模型，二分答案后最小割判定即可。
- 时间复杂度为 $O(MaxFlow(n^2, n^2) \times \log n)$。
- 可以用特殊的方式将时间复杂度优化为 $O(MaxFlow(n, n^2) \times \log n)$。

例题 10

集合

A 集合有 n 个元素，分别为 $A_1, A_2, ..., A_n$。B 集合有 m 个元素，分别为 $B_1, B_2, ..., B_m$。两个集合中的元素均为 01 字符串。
每次你可以在以下两种方式中选择一种进行操作：
1、删除：将某个集合中的某个元素删除。（每个元素只能被删除一次，删除一次之后就没了）
2、修改：将某个集合中的某个元素的一段区间倒置。（例如元素 10010 将区间 $[1, 3]$ 倒置后得到 00110）
删除 A 集合中第 i 个元素的花费为 da_i，删除 B 集合中第 i 个元素的花费为 db_i，修改一次 A 集合中第 i 个元素的花费为 ma_i，修改一次 B 集合中第 i 个元素的花费为 mb_i。
请问，让 A 集合和 B 集合相等的最小花费是多少？（这里的集合为可重集合，即可能有重复元素）
$0 \leq n, m \leq 50$ ，$1 \leq |A_i|, |B_i| \leq 16$，$1 \leq da_i, db_i, ma_i, mb_i \leq 1000$。

- 对于 A 集合的每个元素 x 与 B 集合的每个元素 y，我们可以用 BFS 计算出它们通过两种操作得到相同的某个元素的最小代价（若无法通过操作重合则把代价认为是 $+inf$），与 $da_x + db_y$ 取 min。

- 对于 A 集合的每个元素 x 与 B 集合的每个元素 y，我们可以用 BFS 计算出它们通过两种操作得到相同的某个元素的最小代价（若无法通过操作重合则把代价认为是 $+inf$），与 $da_x + db_y$ 取 min。
- 如果 $n = m$ 就可以直接使用二分图最大权匹配计算，否则我们可以在 X 部或 Y 部补上 $|n - m|$ 个辅助结点，边权设为删掉某个元素的代价。

cont'd

- 对于 A 集合的每个元素 x 与 B 集合的每个元素 y，我们可以用 BFS 计算出它们通过两种操作得到相同的某个元素的最小代价（若无法通过操作重合则把代价认为是 $+inf$），与 $da_x + db_y$ 取 min。
- 如果 $n = m$ 就可以直接使用二分图最大权匹配计算，否则我们可以在 X 部或 Y 部补上 $|n - m|$ 个辅助结点，边权设为删掉某个元素的代价。
- 这样就不用算有下界不固定流量的最小费用流了（捂脸

- 对于 A 集合的每个元素 x 与 B 集合的每个元素 y，我们可以用 BFS 计算出它们通过两种操作得到相同的某个元素的最小代价（若无法通过操作重合则把代价认为是 $+inf$），与 $da_x + db_y$ 取 min。
- 如果 $n = m$ 就可以直接使用二分图最大权匹配计算，否则我们可以在 X 部或 Y 部补上 $|n - m|$ 个辅助结点，边权设为删掉某个元素的代价。
- 这样就不用算有下界不固定流量的最小费用流了（捂脸
- 时间复杂度为 $O(MinCostMaxFlow(n, n^2) + n \times |A|^2 \times 2^{|A|})$。

Q & A