3 山遥路远 (distant)

给定 n 个点，m 条边的有向图，其中每条边有个长度 w，且有一个左括号或者右括号，称路径合法当且仅当沿着路径走过的括号序列是合法括号序列。有 q 组询问，每次问一组 s 到 t 的最短合法路径长度。

3.1 observation

事实上路径长度不会超过 n^3w，括号**层数**不会超过 n^2。

证明将会在后面的算法叙述中自然而然地出现。

3.2 dijkstra $O(mn^3 \log n)$

我们从每个起点开始做一个 dijkstra，不妨将左括号看做 $+1$，右括号看做 -1，那么要求就是任何一个前缀和都是非负的，且终点位置是 0，我们记 $f(u, d)$ 表示到达 u 节点且现在和为 $d(d \geq 0)$ 的情况下的最短路。因此对于每个起点都有 n^3 个状态（$0 \leq d \leq n^2$），对每个相同的 d 都会有 m 种转移，所以复杂度是 $O(n \cdot mn^2 \log n)$。如果你不知道层数有这个上界而是在程序里内嵌了一个 $MAX(MAX \geq n^2)$ 的话，复杂度会是 $O(n \cdot m \cdot MAX \log n)$。由于数据将括号层数构造到了 $\Omega(n^2)$，所以 MAX 的大小不对的话会显著影响你的分数。构造方法留作习题。

3.3 类 Floyd 转移 $O((m^2 + n^3) \log n)$

我们记 $f(u, v)$ 是 u 到 v 的最短路径，若 $u \neq v$，则说明这个括号序列非空，我们有两种可能的转移情况：

- 存在 a, b，这条路径的经过顺序是 $u \to a \rightsquigarrow b \to v$，其中 $a \rightsquigarrow b$ 是合法的，$u \to a$ 是左括号，$b \to v$ 是右括号。

- 存在 w，这条路径的经过顺序是 $u \rightsquigarrow w \rightsquigarrow v$，其中 $u \rightsquigarrow w, w \rightsquigarrow v$ 分别是合法的。

因此我们考虑贪心，将所有的 $f(u,v)$ 推到优先队列里，每次最小的 $f(u,v)$ 取出来后，说明该点对的答案已经确定，我们用于更新剩下的答案。那么第一种情况是两边加一个括号，我们枚举 u 的所有入边，v 的所有出边检查是否能够更新，注意到每对边最多被更新一次，这个将引发 $O(m^2)$ 次更新。第二种情况是枚举 w，检查能否更新 $u \rightsquigarrow v \rightsquigarrow w$ 和 $w \rightsquigarrow u \rightsquigarrow v$，总共引发 $O(n^3)$ 次更新。如果使用 `priority_queue` 那么复杂度是 $O((m^2+n^3)\log n)$，如果使用诸如 `__gnu_pbds::priority_queue` 中一些支持 $O(1)$ decrease-key 的堆，那么复杂度是 $O(m^2+n^3)$。

3.4 优化 $O((mn+n^3)\log n)$

我们考虑优化边枚举的部分。定义一个中间状态 $g(u,v)$ 表示 g 引出了一个左括号还没有被消掉，那么 f 更新的时候枚举一边先用于更新 g，g 更新的时候枚举一边再用于更新 f 就可以了。

复杂度是 $O((mn+n^3)\log n)$，或者 $O(mn+n^3)$（高效堆）。

3.5 答案上界的证明

根据上述构造，我们记 $d(u,v)$ 是达到 $f(u,v)$ 的最短路径，此时的括号深度。则根据转移形式，

- 存在 a,b，$d(u,v) = d(a,b)+1$
- 存在 w，$d(u,v) = \max(d(u,w), d(w,v))$

我们考虑最短路径的依赖关系，一个极重要的性质是：$f(u,v)$ 依赖的路径中显然不会有它自己。

因此我们考虑 $d(u,v)$，这说明必然在路径中存在一个长为 $d(u,v)$ 的依赖链。但是因为依赖链中不能出现两个相同元素，根据鸽笼原理，$d(u,v) < n^2$。

反观最初的 bfs 算法必然能算出答案，而最短路径必然经过每个点最多一次，故路径长度 $f(u,v) < n \cdot (d+1) \leq n^3$。

3.6　阳间的去 log 方法

我们假设你不知道如何使用 `__gnu_pbds::priority_queue`，设正整数 k，我们如果用一个 $V^{1/k}$ 叉树来用于维护堆的话，那么当一个元素的值减小时只需要 $O(k)$ 就能进行更新，弹出最小值只需要 $O(kV^{1/k})$。

我们的最短路中这两种操作次数其实不太平衡，所以复杂度就是 $O(k(nm+n^3)+kn^{2/k}n^2)$。当 k 取任何 ≥ 2 的常数的时候复杂度都是 $O(nm+n^3)$。