实验舱
青少年编程
走近科学 走进名校

提高算法班

经典最短路问题

Mas

Floyd

Floyd 算法适用于任何图，可用于求任意两点间最短路长度

定义 $f[k][i][j]$ 表示仅允许经过节点 $1\sim k$ 时，i 到 j 的最短路长度

当 $k=0$ 时

- $i=j$ 时，$f[0][i][j]\leftarrow 0$
- i,j 存在一条边权为 w 的边时，$f[0][i][j]\leftarrow w$
- i,j 不存在直接相连的边时，$f[0][i][j]\leftarrow +\infty$

当 $1\le k\le n$ 时

$$f[k][i][j]=\min(f[k-1][i][k]+f[k-1][k][j])$$

$f[n][i][j]$ 为最终 i,j 之间最短路长度，时/空间复杂度 $O(n^3)$

将 k 作为阶段递推求解

对于任意 $f[k]$ 仅与 $f[k-1]$ 有关原地修改并不影响结果，可省略最高维度 空间复杂度 $O(n^2)$

Floyd

下图 $1 \to 3$ 最短路为 $1 \to 5 \to 4 \to 3$, 长度为 6

若不将 k 作为阶段 , 将其放于最内层循环

$i = 1$, $j = 3$ 的情况仅出现一次 , 考虑 $k = 4$ 时

当尝试

$$f[4][1][3] \leftarrow f[4][1][4] + f[4][4][3]$$

而 $f[4][1][4]$ 需依赖 $f[5][1][5] + f[5][5][4]$ 更新

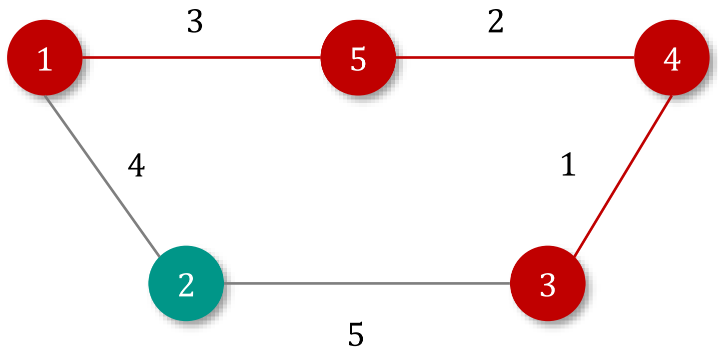
而 $f[4][1][4]$ 应当在 $i = 1$, $j = 4$, $k = 5$ 时更新 , 此时出现错误

而当 k 作为阶段放于最外层时

同样为 $i = 1$, $j = 3$, $k = 4$ 时此时依然无法成功更新

但 $k = 4$ 时 $f[4][5][3]$ 能被正确更新

当 $k = 5$ 时 $f[5][1][5] + f[5][5][3]$ 可正确更新 $f[5][1][3]$



#2931、安全路线

题目描述

saM 生活在一个危险的国家

这个国家有 n 个城市,编号 $1 \sim n$,每个城市都有被绑架或抢劫的风险

其中第 i 个城市的危险程度定义为 r_i

现在 saM 想要从城市 u 去城市 v 但是他不希望经过危险程度超过 X 的城市

输入格式

第一行输入一个整数 n 表示城市个数

第二行输入 n 个整数其中第 i 整数为 r_i

接下来 n 行每行 n 个整数 d_{ij}

其中第 i 行第 j 列表示从 i 到 j 有一条长度为 d_{ij} 的道路

接下来输入一个整数 q ,表示 q 组询问

每组询问输入三个整数 u, v, x ,请你找出 u 到 v 不经过危险长度超过 x 的最短路径

输出格式

对于每组询问输出一行一个整数表示 u 到 v 不经过危险长度超过 x 的最短路径长度

令 $f[k][i][j]$ 表示仅经过$1 \sim k$ 节点 $u \leftrightarrow v$ 最短路长度

原图按 r_i 升序排序重新建图

那么新图点编号与r_i 一一对应且满足单调性

$Floyd$ 预处理出 f 数组

对于每组询问,在新图中找出第一个大于 x 的点编号k

$f[k-1][u][v]$ 即为答案

时间复杂度 $O(n^3+q\log n)$

数据规模

对于 10% 的数据 $1 \le n \le 50, 1 \le q \le 100$

对于 30% 的数据 $1 \le n \le 200, 1 \le q \le 1000$

对于 100% 的数据 $1 \le n \le 200, 1 \le q \le 10^5, 1 \le u, v \le n, 0 \le d_{ij}, r_i \le 10^5$

#1254 奶牛的比赛

题目描述

FJ 的 N 头奶牛们最近参加了场程序设计竞赛:)

在赛场上,奶牛们按 $1 \sim N$ 依次编号

每头奶牛的编程能力不尽相同,并且没有哪两头奶牛的水平不相上下,也就是说,奶牛们的编程能力有明确的排名

整个比赛被分成了若干轮,每一轮是两头指定编号的奶牛的对决

如果编号为 A 的奶牛的编程能力强于编号为 B 的奶牛,那么她们的对决中,编号为 A 的奶牛总是能胜出

FJ 想知道奶牛们编程能力的具体排名,于是他找来了奶牛们所有 M 轮比赛的结果
希望你能根据这些信息,推断出尽可能多的奶牛的编程能力排名

比赛结果保证不会自相矛盾

输入格式

第 1 行: 2 个用空格隔开的整数: N 和 M

第 $2 \sim M+1$ 行: 每行为 2 个用空格隔开的整数 A、 B ,描述了参加某一轮比赛的奶牛的编号,以及结果
编号为 A 的奶牛为胜者

输出格式

第 1 行: 输出 1 个整数,表示排名可以确定的奶牛的数目

数据规模与提示

当一头奶牛输的场数和赢得场数之和为 $N-1$ 时,它的排名可以确定

对于全部的数据 $1 \leq N \leq 100, 1 \leq A, B \leq N, A \neq B, 1 \leq M \leq 4500$

#1254 奶牛的比赛

传递性

令 $\odot$ 是 S 上的二元关系

$\forall a,b,c \in S$,若$a \odot b \land b \odot c$,则$a \odot c$,则称 $\odot$ 具有传递性(或称 $\odot$ 是传递关系)

图的连通性/大于/整除等都满足传递性

令 $f[u][v] = 0/1$ 表示 u,v 之间的胜负情况是否被确定

Floyd 算法中 考虑了 三点间 所有情况

当处于第 k 个阶段时 $i \leftrightarrow k, k \leftrightarrow j$之间的关系已确定

当出现 $f[i][k]$ 确定 且 $f[k][j]$ 也确定时,那么 $f[i][j]$ 也成立

当关系存在传递性时可以使用 Floyd 扩展关系

可使用 bitset 优化至 $O(\frac{n^3}{W})$, 其中 W 为计算机一个整型变量的大小

Dijkstra

Dijkstra 是一种求解 **非负权图** 上单源最短路径的算法

将结点分成两个集合：已确定最短路长度的点集(记为 S 集合)的和未确定最短路长度的点集(记为 T 集合)

对于起点 s，开始时时所有的点都属于 T 集合，令 $\mathrm{dis}_s = 0$ 其他点的 dis 均为 $+\infty$

重复下列操作：

1. 从 T 集合中,选取一个最短路长度最小的结点 u ，加入 S 集合中
2. 对那些刚刚被加入 S 集合的结点的所有出边执行松弛操作

直到 T 集合为空算法结束（共进行 $|V|$ 次）

记 D_u 为 $s \to u$ 的真实最短路长度，dis_u 为预估最短路长度

Dijkstra

命题

对于非负权图 $G=(V,E)$ ，Dijkstra 算法进行第 k 步时 $\forall u \in S$ 都有 $\mathrm{dis}_u = \mathrm{D}_u$

证明

当 $k = 1$ 时 $S = \{s\}$ 显然 $\mathrm{dis}_s = \mathrm{D}_s = 0$ 成立

假设当 $k = n$ 时成立，考虑 $k = n + 1$ 时

设第 $k + 1$ 步选择节点 u

对于路径 $s \to x \to v \to u$ 其长为 L

其中 u 为第一个属于 T 中的节点 x 为 v 的前驱(显然 $v \in S$)

当 $s = x$ 或 $v = u$ 并不影响结论正确性

由于第 $k + 1$ 步选择节点 u 而非节点 v 所以有

$$\mathrm{dis}_u \le \mathrm{dis}_v$$

Dijkstra

记 $v \to w$ 路径长度为 w，显然 $\text{dis}_v + w$ 不可能超过真实路径长度

$$\text{dis}_v + w \le L$$

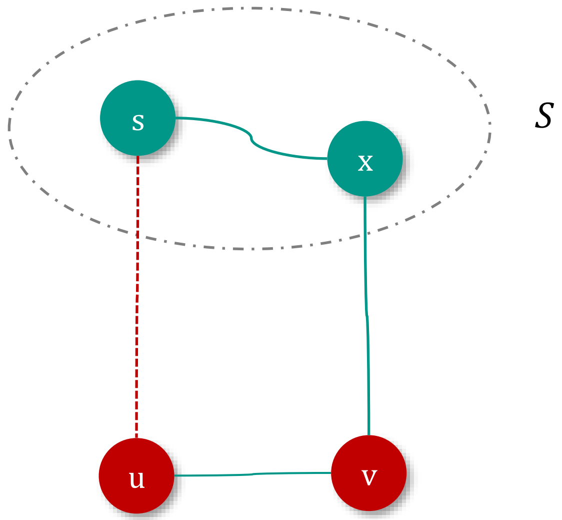
由于 $w \ge 0$ 那么

$$\text{dis}_u \le \text{dis}_v \le \text{dis}_v + w \le L$$

对于任意路径长度 L 都有 $\text{dis}_v \le L$ 即

$$\text{dis}_u = \text{D}_u$$

命题得证



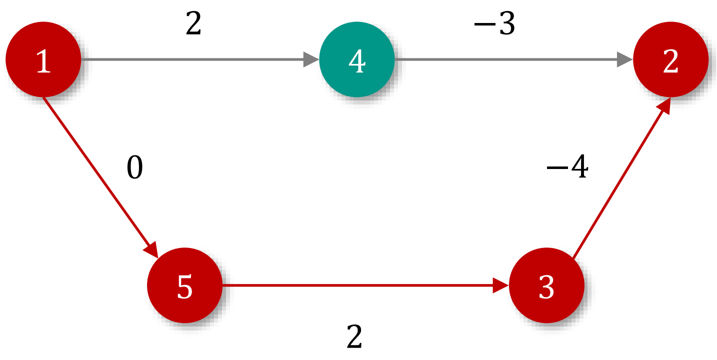
若图中出现负边权?

一种容易想到的方法是给所有边的边权同时加上一个正数 x，从而让所有边的边权均非负

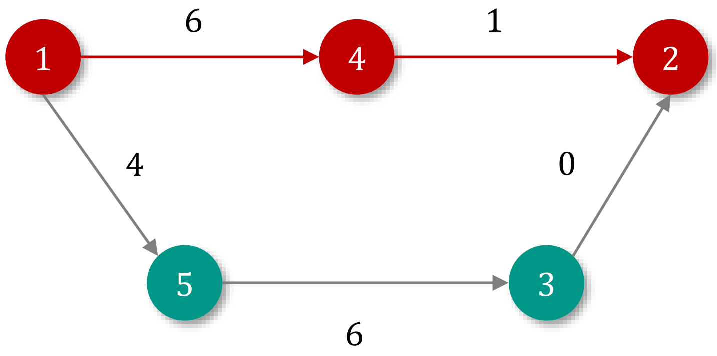
若新图上起点到终点的最短路经过了 k 条边，则将最短路减去 kx 即可得到实际最短路

Dijkstra

下图中 1 → 2 最短路为 1 → 5 → 2 → 3 → 2，长度为 −2



每条边都增加 4 后



新图中最短路为 1 → 4 → 2 并不正确

Dijkstra

有多种方法来维护 1 操作中最短路长度最小的结点,不同的实现导致了 Dijkstra 算法时间复杂度上的差异

暴力

每次 2 操作执行完毕后,在 T 集合中暴力寻找最短路长度最小的结点

2 操作总时间复杂度为 $O(m)$, 1 操作总时间复杂度为 $O(n^2)$,总时间复杂度 $O(n^2+m)$

二叉堆

每成功松弛一条边 (u,v) ,将 v 插入二叉堆中(若 v 已在堆中，直接修改相应元素的权值)，1 操作直接取堆顶即可

共计 m 次插入/修改, n 次删除堆顶操作

插入/修改/删除 的时间复杂度均为 $O(\log n)$, 总时间复杂度为 $O((n+m)\log n)$

优先队列

和二叉堆类似，但使用优先队列时，若一点最短路被更新多次，因先前插入的元素不能被删除/修改，只能留在优先队列中

故优先队列内的元素最多为 m 个，总时间复杂度为 $O((n+m)\log m)$

Dijkstra 和线段树

上页 提到了三种 Dijkstra 算法最常见的维护方法，事实上还有第四种维护方法

线段树

线段树的下标是点的标号,维护的值是目前起点到这个点的最短距离

每成功松弛一条边 (u, v)，把线段树下标 v 的值和 $\text{dis}_u + w$ 取 min，1 操作直接取线段树根节点的最小值

共计 m 次 插入/修改，总时间复杂度为 $O((n+m)\log n)$

线段树的独有优势

用线段树还能做一些特殊的最短路。例如：把常规的 (u, v, w) 的边表示改成 (u, v_1, v_2, w),表示点 u 到 $v_1+1, v_1+2, \ldots, v_2$

这些点都有一条边权为 w 的边,然后还是求起点到每个点的最短路

如果我们用线段树来维护 Dijkstra 算法，可以很自然地解决这个拓展问题

每次松弛一条边 (u, v_1, v_2, w) 时,相当于把线段树的 $[v_1, v_2]$ 这段区间求一个 min ，打上永久化标记即可

总时间复杂度依然为 $O((n+m)\log n)$

Dijkstra

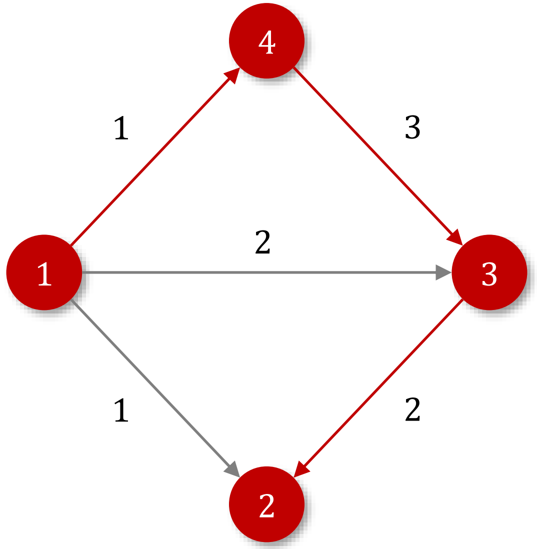
朴素 Dijkstra 算法不可用于求解最长路

右图中 $1 \to 2$ 最长路为 $1 \to 4 \to 3 \to 2$ ，长度为 6

若初始时令 $\mathrm{dis}_s \leftarrow 0$,其他点的 dis 均为 $-\infty$

那么第二个被加入 S 的节点 3 , 此时 $\mathrm{dis}_3 = 2$ 并非最长路

后续不再以 3 为起点进行更新



但若允许一个点多次松弛 , 那么堆优化 Dijkstra 算法效率可能劣于 SPFA

#2512 一个人的旅行

实验舱 青少年编程 走近科学 走进名校

题目描述

每年暑假 Mas 都会很忙碌

在忙碌结束之后, Mas 就会去旅行,因为在旅途中会遇见很多人,很多事,还能丰富自己的阅历,还可以看美丽的风景......

Mas 想去很多地方,想要去东京铁塔看夜景,去威尼斯看电影,去阳明山上看海芋,去纽约纯粹看雪景,去巴黎喝咖啡写信......

眼看新学期又要到了,所以 Mas 决定在要在最短的时间去一个自己想去的地方!

因为 Mas 的家在一个小城上,没有火车经过,所以他只能去邻近的城市坐火车

输入格式

第一行是三个整数 n, m, s, d ,表示有 n 个城市, m 条路,和 Mas 家相邻的城市的有 s 个,想去的地方有 d 个

接着有 m 行,每行有三个整数 u, v, w ,表示 u, v 城市之间的车程是 w 小时(u, v 之间可能有多条路)

接着的第 $m+1$ 行有 s 个数,表示和 Mas 家相连的城市

接着的第 $d+2$ 行有 d 个数,表示 Mas 想去地方

输出格式

输出 Mas 能去某个喜欢的城市的最短时间

数据规模

对于 30% 的数据 $1 \leq n \leq 200$

对于 50% 的数据 $1 \leq n \leq 1000, 1 \leq s, d \leq 5$

对于全部的数据 $1 \leq n, m \leq 100000, 1 \leq s, d \leq 100, 1 \leq u, v \leq n, 1 \leq w \leq 1000$

#2512 一个人的旅行

若考虑 Floyd 时间复杂度 $O(n^3)$

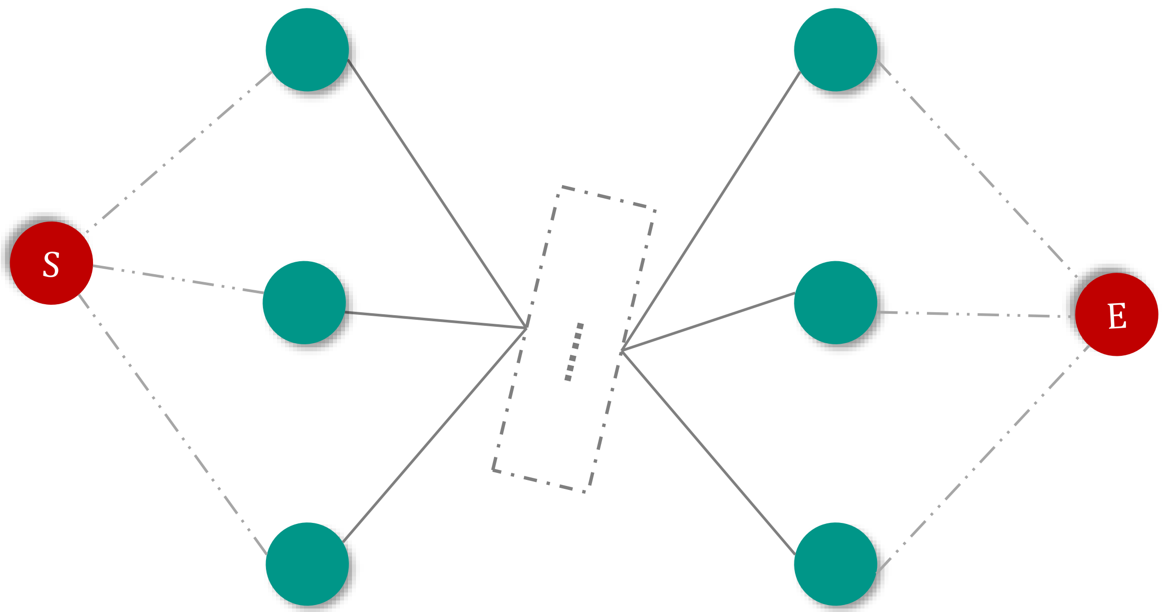
建立虚拟起点 S 虚拟终点 E

所有起点都与 S 连一条边权为 0 的边

所有起点都与 E 连一条边权为 0 的边

从 S 跑一遍 Dijkstra 算法，dis_E 即为答案

时间复杂度 $O(\,(m+n)\log m\,)$



无需显式连边

令所有起点 dis 为 0 跑最短路算法 , 对所有终点 dis 取 min 即可

#2165、送餐

题目描述

给定一个 n 点 m 边的有向带权图表示一座城市,起点为 1

送餐小哥需要给 n 个客户送外卖,第 i 个客户的家在第 i 号点

由于他的车子容量很小,所以一次只能容纳一份外卖

所以送达外卖之后就要回到起点取新的外卖送下一单,直到全部送到位置(最后需要回到起点)

有向图保证联通,外卖小哥一定走的最短路

求送餐小哥走的总路程

输入格式

第一行一个整数 T ,表示数据组数

对于每组数据,第一行两个整数 n 和 m

接下来 m 行,每行三个整数 u_i, v_i, c_i 表示每条有向边

输出格式

对于每组数据,输出一行一个整数表示答案

数据范围

对于 20% 的数据: $0 < n \le 100$

对于 40% 的数据: $0 < n \le 300$

对于 60% 的数据: $0 < n \le 1000$

对于 100% 的数据: $0 < n \le 20000, m \le 60000, 1 \le T \le 10, 0 \le c_i \le 10^9$

保证答案在 `long long` 范围内

要求 1 → 其他点的最短路长度

其他点 → 1的最短路长度

对于每个图上边取反(反向建图)

从 n 跑一遍最短路 即可求出 n 到其他点的最短路

Bellman-Ford

$\mathrm{Bellman-Ford}$ 算法是一种基于松弛操作的最短路算法

$\mathrm{Bellman-Ford}$ 可求出有负权图的最短路,并可对最短路不存在(负环)的情况进行判断

不断尝试对图上每一条边进行松弛

每进行一轮循环,就对图上所有的边都尝试进行一次松弛操作

当一次循环中没有成功的松弛操作时,算法停止

对于每一轮松弛 时间复杂度为 $O(m)$

当最短路存在时, 最短路上至多存在 $n-1$ 条边 , 每一轮松弛使得最短路上的边数至少+1

故需要执行 $n-1$ 轮松弛 , 总时间复杂度 $O(nm)$

若发现第 n 轮依然松弛成功 , 说明从起点出发能够到达一个负环

若需判断图上是否存在负环,需要考虑图的连通情况

SPFA

SPFA(Shortest Path Faster Algorithm) 即队列优化的 Bellman − Ford 算法

很多时候并不需要那么多无用的松弛操作

显然,只有上一次被松弛成功的结点其所连接的边,才有可能引起下一次松弛的成功

考虑用队列来维护 **可能引起松弛成功的点**，每次仅从队列中取出节点尝试松弛

SPFA 也可以用于判断起点是否能抵达一个**负环**

算法过程中记录最短路经过了多少条边，当经过了至少 n 条边时，说明起点可到达一个负环

虽然在大多数情况下 SPFA 跑得很快(在随机图上可认为时间复杂度 $O(m)$)

但其最坏情况下的时间复杂度为 $O(nm)$，且将其卡到$O(nm)$ 并不难

如何看待 SPFA 算法已死这种说法？ - immortalCO 的回答 - 知乎 https://www.zhihu.com/question/292283275/answer/484694411

#2156、有边数限制的最短路

题目描述

给定一个 n 个点 m 条边的有向图,图中可能存在**重边**和**自环**, 边权可能为负数

请你求出从 1 号点到 n 号点的最多经过 k 条边的最短距离,如果无法从 1 号点走到 n 号点,输出 `impossible`

> *图中可能存在负权回路*

输入格式

第一行包含三个整数 n , m , k

接下来m行,每行包含三个整数 u , v , w ,表示存在一条从点 u 到点 v 的有向边,边长为 w

输出格式

输出一个整数,表示从 1 号点到 n 号点的最多经过 k 条边的最短距离

如果不存在满足条件的路径,首先输出能到达的最大点编号再输出 `impossible`

数据范围

对于全部的数据范围 $1 \le n, k \le 500, 1 \le m \le 10000$
任意边长的绝对值不超过 10000

输入样例1

```
3 3 1
1 2 1
2 3 1
1 3 3
```

输出样例1

```
3
```

#2156、有边数限制的最短路

若最短路存在至多经过 n 个点 $n-1$ 条边

故 $\mathrm{Bellman}-\mathrm{Ford}$ 进行 $n-1$ 轮松弛(最坏情况下每轮仅松弛成功一条边)

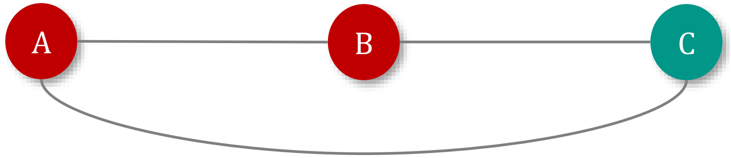
本题进行 $\min(n-1,k)$ 轮松弛

本轮松弛中 $A\sim B$ 松弛成功,若接下来 $B\sim C$ 松弛成功

若在当前 dis 数组基础上修改

实际上 $A\rightarrow C$ 间本轮经过了两条边

滚动数组从上一轮松弛结果中修改



时间复杂度 $O(km)$

#1891、最短路径输出

实验舱 青少年编程 走近科学 走进名校

题目描述

给出一个有向图 $G=(V,E)$,和一个源点 $v_0 \in V$,请写一个程序输出 v_0 和图 G 中其它顶点的最短路径

只要所有的有向环权值和都是正的,我们就允许图的边有负值

顶点的标号从 1 到 n (n 为图 G 的顶点数)

输入格式

第 1 行：两个正数 $n, m(2 \le n \le m \le 100)$,表示图 G 的顶点总数

第 2 行：一个整数,表示源点 $v0$ ($v_0 \in V$, v_0 可以是图 G 中任意一个顶点)

第 3 至第 $n+2$ 行,用一个邻接矩阵 W 给出了这个图

输出格式

每一行输出起点 v_0 到其他所有点的最短路径,按照字典序输出(如果 v_0 无法到达该点,不输出)

> *若最短路径不唯一,输出任意一条*

#1891、最短路径输出

Dijkstra/SPFA

前驱记录，在松弛成功时记录前驱节点编号，逆序输出

Floyd

前驱记录

$\text{path}[u][v]$记录 $u \longrightarrow v$ 路径中除 u 外的第一个节点编号

初始时 $\text{path}[u][v] = v$，松弛成功后令$\text{path}[i][j] = \text{path}[i][k]$

后驱记录

$\text{path}[u][v]$ 记录 $u \longrightarrow v$ 路径中除 v 外最后一个节点编号

初始时 $path[u][v] = u$，松弛成功后令 $\text{path}[i][j] = \text{path}[k][j]$

中间点记录

记录松弛点k，可将路径 u~v 拆分成 u~$\text{path}[u][v]$，$\text{path}[u][v]$~v

#675、最短路计数

题目描述

给出一个 N 个顶点 M 条边的无向无权图,顶点编号为 $1 \sim N$

问从顶点 1 开始,到其他每个点的最短路有几条

输入格式

第一行包含 2 个正整数 N, M ,为图的顶点数与边数

接下来 M 行,每行两个正整数 x, y ,表示有一条顶点 x 连向顶点 y 的边

> *可能有自环与重边*

输出格式

输出 N 行,每行一个非负整数,第 i 行输出从顶点 1 到顶点 i 有多少条不同的最短路
答案有可能会很大,你只需要输出 $\mod 100003$ 后的结果即可

如果无法到达顶点 i 则输出 0

数据范围

对于 20% 的数据, $N \le 100$

对于 60% 的数据, $N \le 1000$

对于 100% 的数据, $1 \le N \le 100000, 0 \le M \le 200000$

考虑 BFS/SPFA/Dijkstra

令 cnt_i 表示 $1 \to i$ 最短路的方案数

当 $\text{dis}_v > \text{dis}_u + w$ 时

最短路更新并令 $\text{cnt}_v = \text{cnt}_u$

当 $\text{dis}_v = \text{dis}_u + w$ 时

说明最短路方案数增加 $\text{cnt}_v \leftarrow \text{cnt}_v + \text{cnt}_u$

若不是单位边权，能否使用 SPFA？

#354、无向图的最小环问题

题目描述

给定一张无向图,求图中一个至少包含 3 个点的环

环上的节点不重复且环上的边的长度之和最小

该问题称为无向图的最小环问题

在本题中,你需要输出最小的环的边权和

若无解,输出 `No solution.`

输入格式

第一行两个正整数 n, m 表示点数和边数

接下来 m 行,每行三个正整数 u, v, d,表示节点 u, v 之间有一条长度为 d 的边

输出格式

输出边权和最小的环的边权和。若无解,输出 `No solution.`

数据规模

对于 20% 的数据, $n, m \le 10$

对于 60% 的数据, $m \le 100$

对于 100% 的数据, $1 \le n \le 100$, $1 \le m \le 5 \times 10^3$, $1 \le d \le 10^5$

设 u 和 v 之间有一条边权为 w 的边

令 $\text{dis}(u,v)$ 表示 **删除 u 和 v 之间的连边** 后 $u \leftrightarrow v$ 最短路即

未考虑 (u,v,w) 条边时的最短路长度

那么 $\text{dis}(u,v) + w$ 为一个环的边权和

考虑所有情况取最小值即为答案

Dijkstra

枚举所有边并每次删除一条边

再对这条边的起点跑一次 Dijkstra，求出上述 $\text{dis}(u,v)$

时间复杂度 $O(m(n+m)\log m)$

#354、无向图的最小环问题

Floyd

$f[k][u][v]$ 表示从 u 到 v 且仅经过编号在 $[1, k)$ 范围内的点的最短路

记 u, v 之间边的边权为 $w(u, v)$

由最小环的定义可知其至少有三个顶点

不妨设其中编号最大的顶点为 k , 环上与 x 相邻两侧的两个点为 u, v

不难发现 $f[k-1][u][v]$ 即为未经过 $u \leftrightarrow v$ 直接连接边时 $u \leftrightarrow v$ 的最短路长度

当 Floyd 处于第 k 阶段时, 环的长度为 $f[k-1][u][v] + w(u, k) + w(k, v)$

对于每个 k 枚举满足 $u < k \land v < k$ 的 (u, v) 对所有环长度取最小值即为答案

时间复杂度 $O(n^3)$

#2668、恰好经过K条边最短路

实验舱 青少年编程 走近科学 走进名校

题目描述

给定一张由 T 条边构成的无向图,点的编号为 $1 \sim 1000$ 之间的整数

求从起点 S 到终点 E 恰好经过 N 条边(可以重复经过)的最短路

> *数据保证一定有解*

输入格式

第 1 行：包含四个整数 N, T, S, E

第 $2 \sim T+1$ 行：每行包含三个整数,描述一条边的边长以及构成边的两个点的编号

输出格式

输出一个整数,表示最短路的长度

数据范围

对于全部的数据 $2 \leq T \leq 100, 2 \leq N \leq 10^6$

输入样例

```
2 6 6 4
11 4 6
4 4 8
8 4 9
6 6 8
2 6 9
3 8 9
```

输出样例

```
10
```

#2668、恰好经过K条边最短路

与 #2156、有边数限制的最短路 相比本题 **必须** 经过 K 条边

图中最多出现 100 条边，即最多 200 个点，可对点进行离散化后重新建图

在 Bellman – Ford 基础上

对每一轮强制松弛一条边更新（dis 数组设为极大值），时间复杂度 $O(km)$

令 $\text{dis}[k][u][v]$ 表示 u 到 v 经过了 k 条边的最短路

当 $k = 1$ 时,若 u 到 v 存在边权为 w 的边 $\text{dis}[1][u][v] = w$ 否则 $\text{dis}[1][u][v] = +\infty$

当 $k > 1$时

$$\text{dis}[k][u][v] = \min_{1 \le x \le k-1} (\text{dis}[x][u][t] + \text{dis}[k-x][t][v])$$

时间复杂度 $O(K^2 n^3)$

#2668、恰好经过K条边最短路

考虑倍增维护

令 $\mathrm{dis}[k][u][v]$ 表示 u 到 u 经过 2^k 条边时最短路长度

$$\mathrm{dis}[k][u][v] = \min(\ \mathrm{dis}[k-1][u][t] +\ \mathrm{dis}[k-1][u][v])$$

进行 $\log K$ 次类似 Floyd 的转移

```
for (int t = 1; t <= __lg(n); t++)
  for (int k = 1; k <= cnt; k++)
    for (int i = 1; i <= cnt; i++)
      for (int j = 1; j <= cnt; j++)
        dis[i][j][t] = min(dis[i][j][t], dis[i][k][t - 1] + dis[k][j][t - 1]);
memset(tmp, 0x3f, sizeof tmp);
tmp[st] = 0;
for (int t = 0; t <= __lg(n); t++)
  if (n >> t & 1)
  {
    memset(res, 0x3f, sizeof res);
    for (int i = 1; i <= cnt; i++)
      for (int j = 1; j <= cnt; j++)
        res[i] = min(res[i], tmp[j] + dis[j][i][t]);
    memcpy(tmp, res, sizeof tmp);
  }
```

对于 K 可以被拆分成若干个 2 的幂次相加得到

对于 2 的幂次进行一轮更新即可

时间复杂度 $O(n^3 \log K)$

上述转移过程与 Floyd 算法流程极为相似，Floyd 得到结果为矩阵，与矩阵加法有什么联系？

#2526、严格次短路

题目描述

贝茜把家搬到了一个小农场,但她常常回到 FJ 的农场去拜访她的朋友

贝茜很喜欢路边的风景,不想那么快地结束她的旅途
于是她每次回农场,都会选择第二短的路径,而不象我们所习惯的那样,选择最短路

贝茜所在的乡村有 R 条双向道路,每条路都联结了所有的 N 个农场中的某两个

贝茜居住在农场 1 ,她的朋友们居住在农场 N (即贝茜每次旅行的目的地)

贝茜选择的第二短的路径中,可以包含任何一条在最短路中出现的道路,并且,一条路可以重复走多次

> *第二短路的长度必须严格大于最短路(可能有多条)的长度,但它的长度必须不大于所有除最短路外的路径的长度*

输入格式

第一行输入两个正整数 N, R
接下来 R 行,每行三个正整数 u, v, w ,表示从 u 到 v 有一条长度为 w 的双向道路 $(1 \le u, v \le n)$

输出格式

输出从 1 到 n 的第二段道路长度

数据规模

对于全部的数据 $1 \le R \le 100000, 1 \le N \le 5000, 1 \le w \le 5000$

思路 1

最短路必然经过一些边

强制不走这些关键边

求出 1 到所有点的最短路 dis1 , n 到所有点最短路 dis2

对严格大于 $dis1_n$ 的结果取最小值即为答案

#2526、严格次短路

思路 2

令 dis1 记录 最短路, dis2 记录严格次短路 , 出队时 d 为入队时 u 的最/次短路径长度

当 $\mathrm{dis1}_v > d + w(u,v)$

更新次短路 $\mathrm{dis2}_v = \mathrm{dis1}_v$

更新最短路 $dis1[v] = d + w(u,v)$ 入队

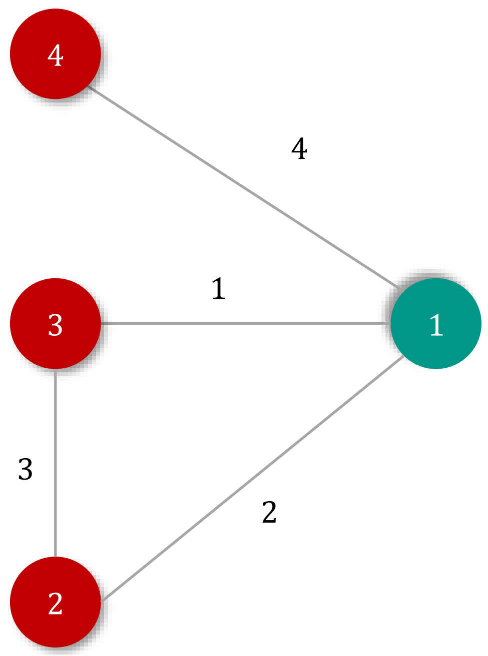
当 $\mathrm{dis2}_v > d + w \wedge \mathrm{dis}_v < d + w$

更新次短路入队

最短路只能被最短路更新,次短路可能被最短路更新,也可能被次短路更新

上图中的 1~4 次短路为 1~1 的次短路 更新得到

由于 次短路 可由 次短路 + 某条边权 得到 , 次短路更新时也需入队



LAB
实验舱
青少年编程
走近科学 走进名校

谢谢观看