图论选讲

Graph

讲 讲 题

李驰科

Forked from NamomoCamp

图论算法

1. **遍历（BFS/DFS）**
2. **最短路/最小生成树**
3. **拓扑排序**
4. *网络流/匹配
5. *强连通/双连通

/01

知识点速过

DFS（图论）

引入

DFS 全称是 Depth First Search，中文名是深度优先搜索，是一种用于遍历或搜索树或图的算法。所谓深度优先，就是说每次都尝试向更深的节点走。

该算法讲解时常常与 BFS 并列，但两者除了都能遍历图的连通块以外，用途完全不同，很少有能混用两种算法的情况。

DFS 常常用来指代用递归函数实现的搜索，但实际上两者并不一样。有关该类搜索思想请参阅 DFS（搜索）.

过程

DFS 最显著的特征在于其 **递归调用自身**。同时与 BFS 类似，DFS 会对其访问过的点打上访问标记，在遍历图时跳过已打过标记的点，以确保 **每个点仅访问一次**。符合以上两条规则的函数，便是广义上的 DFS。

具体地说，DFS 大致结构如下：

```
DFS(v) // v 可以是图中的一个顶点，也可以是抽象的概念，如 dp 状态等。
  在 v 上打访问标记
  for u in v 的相邻节点
    if u 没有打过访问标记 then
      DFS(u)
    end
  end
end
```

BFS（图论）

BFS 全称是 Breadth First Search，中文名是宽度优先搜索，也叫广度优先搜索。

是图上最基础、最重要的搜索算法之一。

所谓宽度优先。就是每次都尝试访问同一层的节点。 如果同一层都访问完了，再访问下一层。

这样做的结果是，BFS 算法找到的路径是从起点开始的 **最短** 合法路径。换言之，这条路径所包含的边数最小。

在 BFS 结束时，每个节点都是通过从起点到该点的最短路径访问的。

算法过程可以看做是图上火苗传播的过程：最开始只有起点着火了，在每一时刻，有火的节点都向它相邻的所有节点传播火苗。

实现

下文中 C++ 与 Python 的代码实现是基于链式前向星的存图方式，其实现可参考 图的存储 页面。

伪代码 | C++ | Python

```
bfs(s) {
  q = new queue()
  q.push(s), visited[s] = true
  while (!q.empty()) {
    u = q.pop()
    for each edge(u, v) {
      if (!visited[v]) {
        q.push(v)
        visited[v] = true
      }
    }
  }
}
```

我们定义无向连通图的 **最小生成树**（Minimum Spanning Tree，MST）为边权和最小的生成树。

注意：只有连通图才有生成树，而对于非连通图，只存在生成森林。

Kruskal 算法

Kruskal 算法是一种常见并且好写的最小生成树算法，由 Kruskal 发明。该算法的基本思想是从小到大加入边，是个贪心算法。

抽象一点地说，维护一堆 **集合**，查询两个元素是否属于同一集合，合并两个集合。

其中，查询两点是否连通和连接两点可以使用并查集维护。

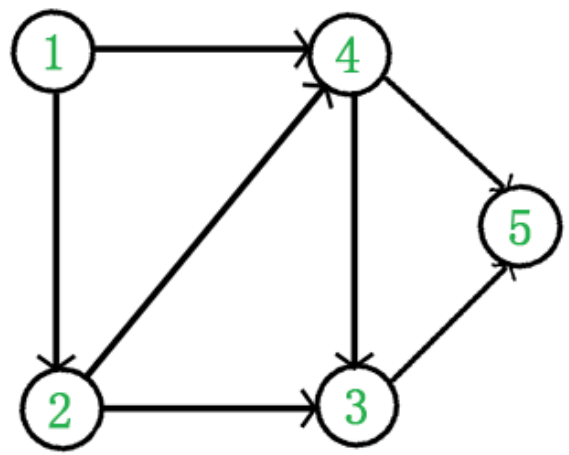
如果使用 $O(m \log m)$ 的排序算法，并且使用 $O(m\alpha(m,n))$ 或 $O(m \log n)$ 的并查集，就可以得到时间复杂度为 $O(m \log m)$ 的 Kruskal 算法。

在图论中，**拓扑排序（Topological Sorting）**是一个**有向无环图（DAG, Directed Acyclic Graph）**的所有顶点的线性序列。且该序列必须满足下面两个条件：

1. 每个顶点出现且只出现一次。
2. 若存在一条从顶点 A 到顶点 B 的路径，那么在序列中顶点 A 出现在顶点 B 的前面。

有向无环图（DAG）才有拓扑排序，非DAG图没有拓扑排序一说。

例如，下面这个图：



它是一个 DAG 图，那么如何写出它的拓扑排序呢？这里说一种比较常用的方法：

1. 从 DAG 图中选择一个 没有前驱（即入度为0）的顶点并输出。
2. 从图中删除该顶点和所有以它为起点的有向边。
3. 重复 1 和 2 直到当前的 DAG 图为空或**当前图中不存在无前驱的顶点为止**。后一种情况说明有向图中必然存在环。

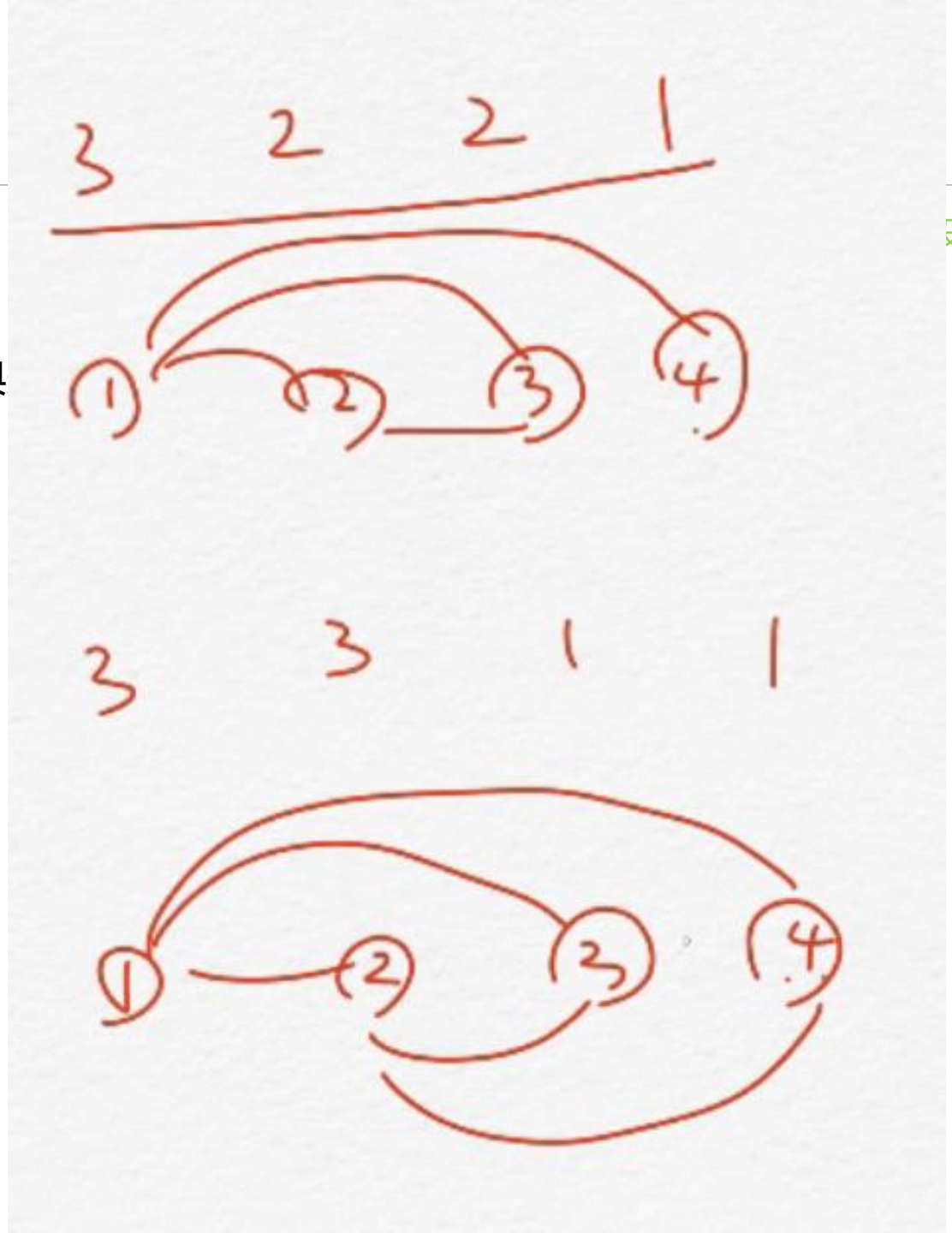
/02

题目选讲

度数序列

设有重边

思考这样一个典

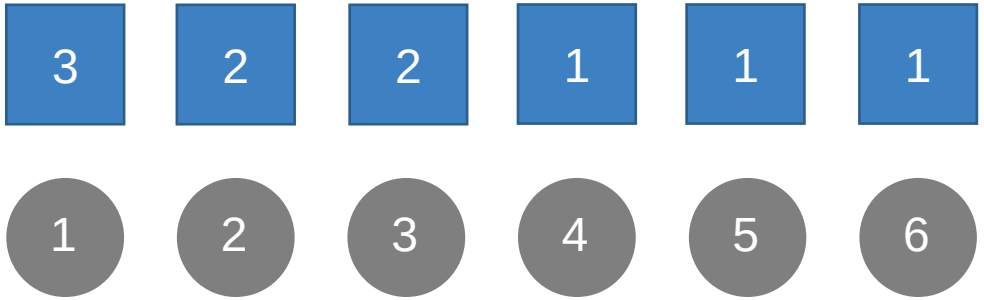


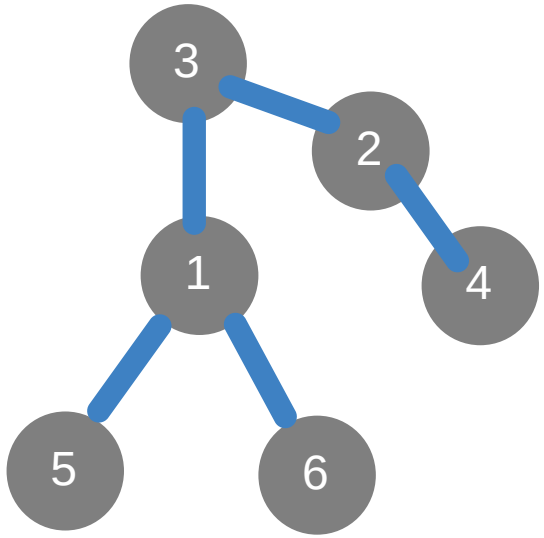
度数序列

① 一个图的度数和一定是偶数
② 一个点的度数一定是≤ n – 1的
③ 树的度数和为**2n – 2**（n – 1条边连接u，v两个点）

度数序列

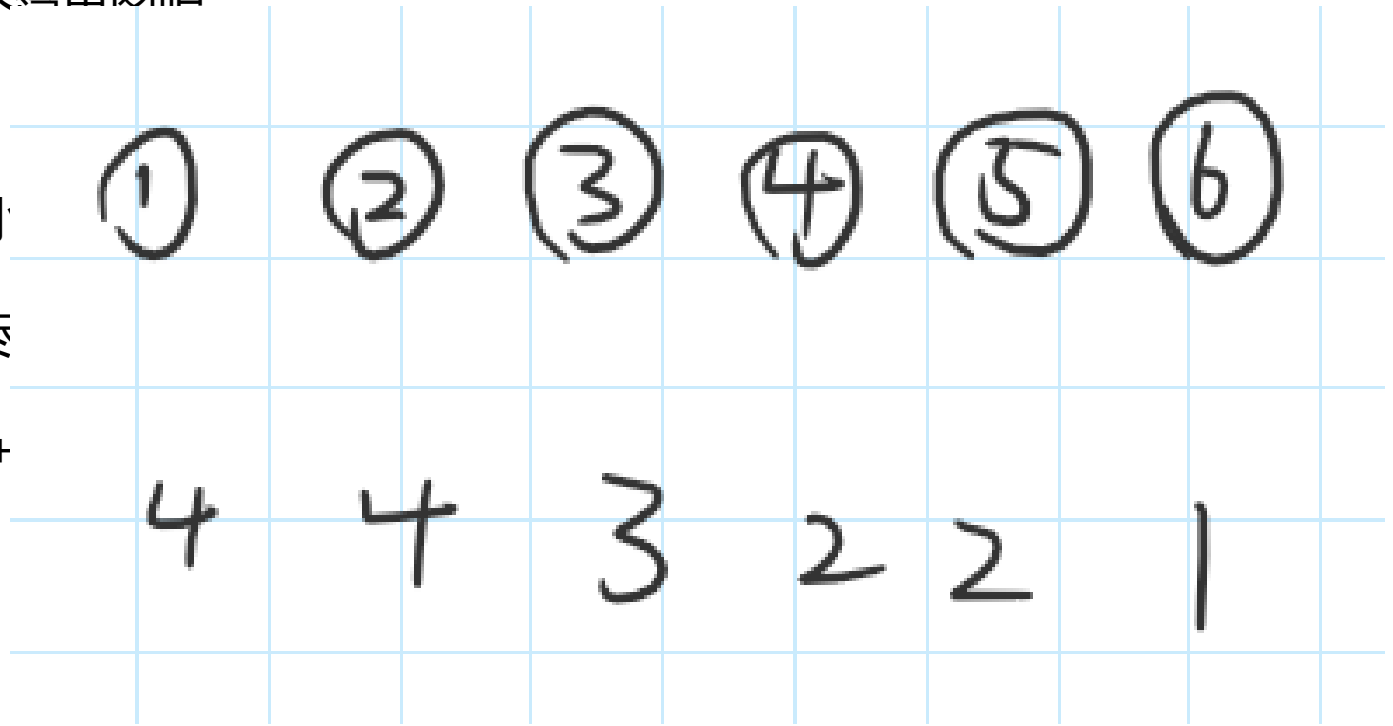
那么，我们如何来构造一棵树呢





度数序列

如何构造一个简单图呢



按度数从大到小

我们希望度数尽

复杂度为O(m +

Killer Sajin's Matrix

"蔚来杯"2022牛客暑期多校训练营（加赛）

K Killer Sajin's Matrix

题意：给定 $n \times m$ 的矩阵，要求在其中填入 k 个 1 使得每行每列 1 的个数均为奇数。输出一个合法方案或者报告无解。$n, m, k \leq 1 \times 10^5$。

Killer Sajin's Matrix

发现当k < max(n, m)或n, m, k的奇偶性不全相同时，一定无解。

① 确定度数序列（尽可能平均，更平均一定不劣）

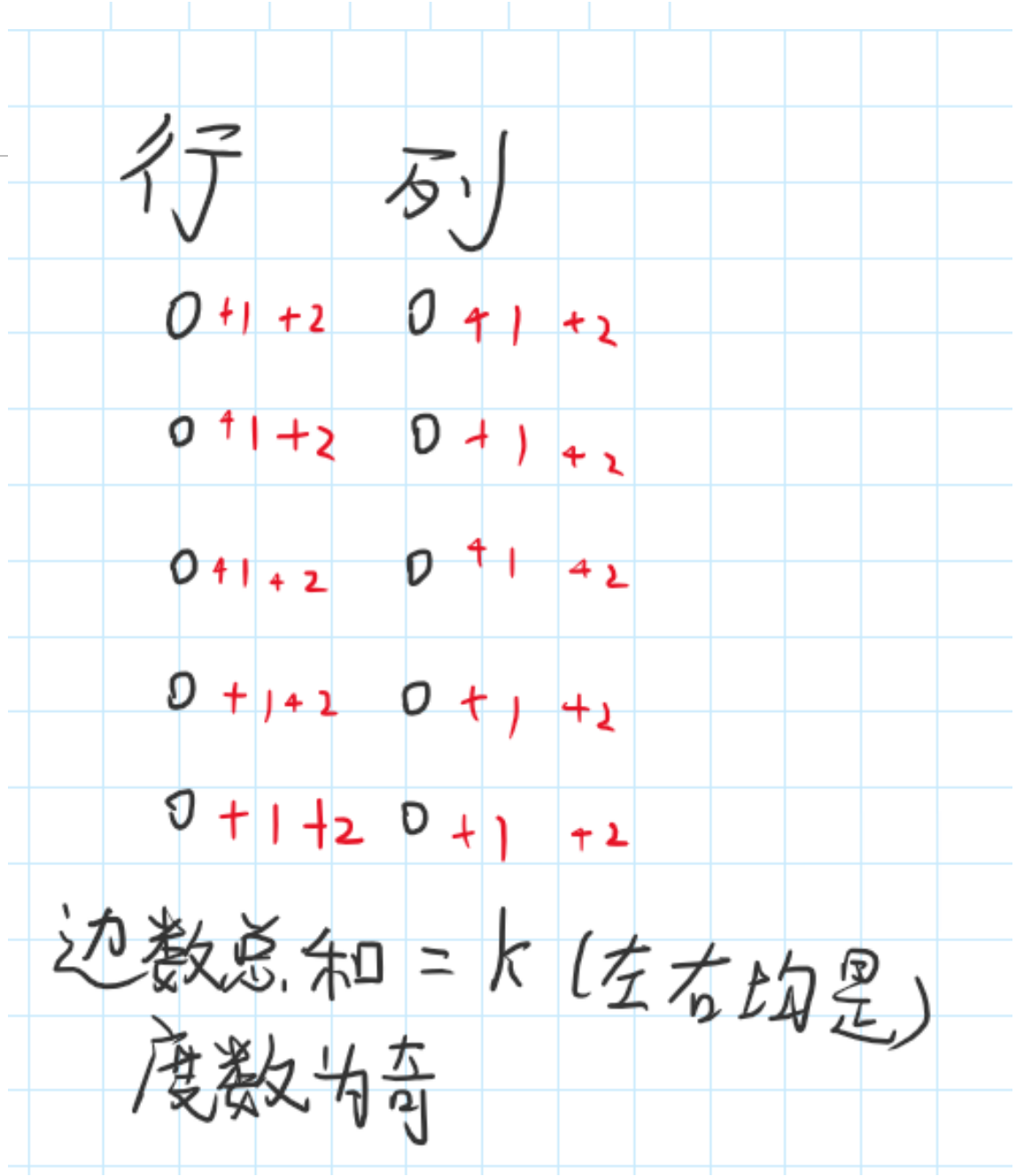
② 确定连边(贪心地选择那些剩余 1 最多的列进行匹配，这可以用优先队列维护。)

构造完成后验证 1 是否用完以及矩阵是否满足条件即可。

https://ac.nowcoder.com/acm/contest/38727/K

C++: 代码查看 (nowcoder.com)

Python: 代码查看 (nowcoder.com)



D - Wizard in Maze

D - Wizard in Maze Editorial

Time Limit: 2 sec / Memory Limit: 1024 MB

Score : 400 points

Problem Statement

A maze is composed of a grid of $H \times W$ squares - H vertical, W horizontal.

The square at the i-th row from the top and the j-th column from the left - (i, j) - is a wall if S_{ij} is `#` and a road if S_{ij} is `.`.

There is a magician in (C_h, C_w). He can do the following two kinds of moves:

- Move A: Walk to a road square that is vertically or horizontally adjacent to the square he is currently in.
- Move B: Use magic to warp himself to a road square in the 5×5 area centered at the square he is currently in.

In either case, he cannot go out of the maze.

At least how many times does he need to use the magic to reach (D_h, D_w)?

Constraints

- $1 \leq H, W \leq 10^3$
- $1 \leq C_h, D_h \leq H$
- $1 \leq C_w, D_w \leq W$
- S_{ij} is `#` or `.`.
- $S_{C_h C_w}$ and $S_{D_h D_w}$ are `.`.
- $(C_h, C_w) \neq (D_h, D_w)$

D - Wizard in Maze (atcoder.jp)

D - Wizard in Maze

题意:
一张 n * m 的地图，上面有空地和墙。给定起点和终点，你可以往上下左右相邻的方格走，但是只能走空地，你也可以用魔法跳到以当前点为中心，5 * 5的范围内的任意空地。要求出走到终点使用魔法的最少次数。

Constraints

- $1 \leq H, W \leq 10^3$
- $1 \leq C_h, D_h \leq H$
- $1 \leq C_w, D_w \leq W$
- S_{ij} is `#` or `.`.
- $S_{C_h C_w}$ and $S_{D_h D_w}$ are `.`.
- $(C_h, C_w) \neq (D_h, D_w)$

D - Wizard in Maze

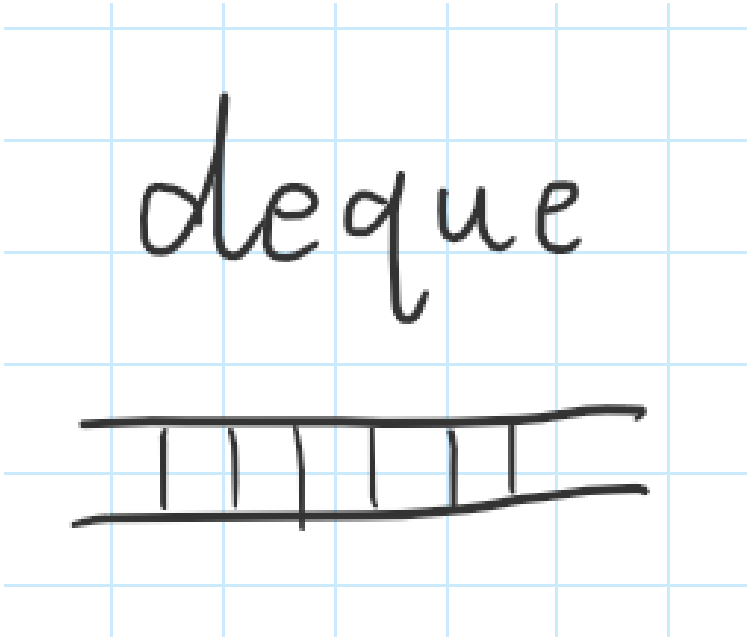
01bfs

如果一条路可以通过第一种方式到达，那就尽量不采用第二种方法。

如果没有达到终点，那么就用第二种方法从第一种方法走的所有路中探索没有走过的路，

如果能达到终点，那么输出最小使用的魔法次数，如果不能那么输出-1。

将第一种移动方法走的路压入队列前端，第二种移动方法压入队列后端，如果第一种方法走的路在队列中没有了且没达到终点，就开始使用魔法。



Cpp：Submission #40907230 - AtCoder Beginner Contest 176

Py：Submission #40907382 - AtCoder Beginner Contest 176

B. Complete The Graph

B. Complete The Graph

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

ZS the Coder has drawn an undirected graph of n vertices numbered from 0 to $n - 1$ and m edges between them. Each edge of the graph is weighted, each weight is a **positive integer**.

The next day, ZS the Coder realized that some of the weights were erased! So he wants to reassign **positive integer** weight to each of the edges which weights were erased, so that the length of the shortest path between vertices s and t in the resulting graph is exactly L. Can you help him?

Input

The first line contains five integers n, m, L, s, t ($2 \le n \le 1000$, $1 \le m \le 10\,000$, $1 \le L \le 10^9$, $0 \le s, t \le n - 1$, $s \ne t$) — the number of vertices, number of edges, the desired length of shortest path, starting vertex and ending vertex respectively.

Then, m lines describing the edges of the graph follow. i-th of them contains three integers, u_i, v_i, w_i ($0 \le u_i, v_i \le n - 1$, $u_i \ne v_i$, $0 \le w_i \le 10^9$). u_i and v_i denote the endpoints of the edge and w_i denotes its weight. If w_i is equal to 0 then the weight of the corresponding edge was erased.

It is guaranteed that there is at most one edge between any pair of vertices.

Output

Print "NO" (without quotes) in the only line if it's not possible to assign the weights in a required way.

Otherwise, print "YES" in the first line. Next m lines should contain the edges of the resulting graph, with weights assigned to edges which weights were erased. i-th of them should contain three integers u_i, v_i and w_i, denoting an edge between vertices u_i and v_i of weight w_i. The edges of the new graph must coincide with the ones in the graph from the input. The weights that were not erased must remain unchanged whereas the new weights can be any **positive integer** not exceeding 10^{18}.

The order of the edges in the output doesn't matter. The length of the shortest path between s and t must be equal to L.

If there are multiple solutions, print any of them.

[Problem - B - Codeforces]

B. Complete The Graph

一个**n**个点**m**条边的无向图，每条边有**非负**边权**vi**，
如果**边权为0**表示可以修改该条边的边权为任意
正整数，给出起点**s**和终点**t。**
问是否有一种修改边权的方案使得**s**到**t**的最短路
为**L**

$(2 \le n \le 1000,\ 1 \le m \le 10\,000,\ 1 \le L \le 10^9,\ 0 \le s, t \le n-1,\ s \ne t)$

B. Complete The Graph

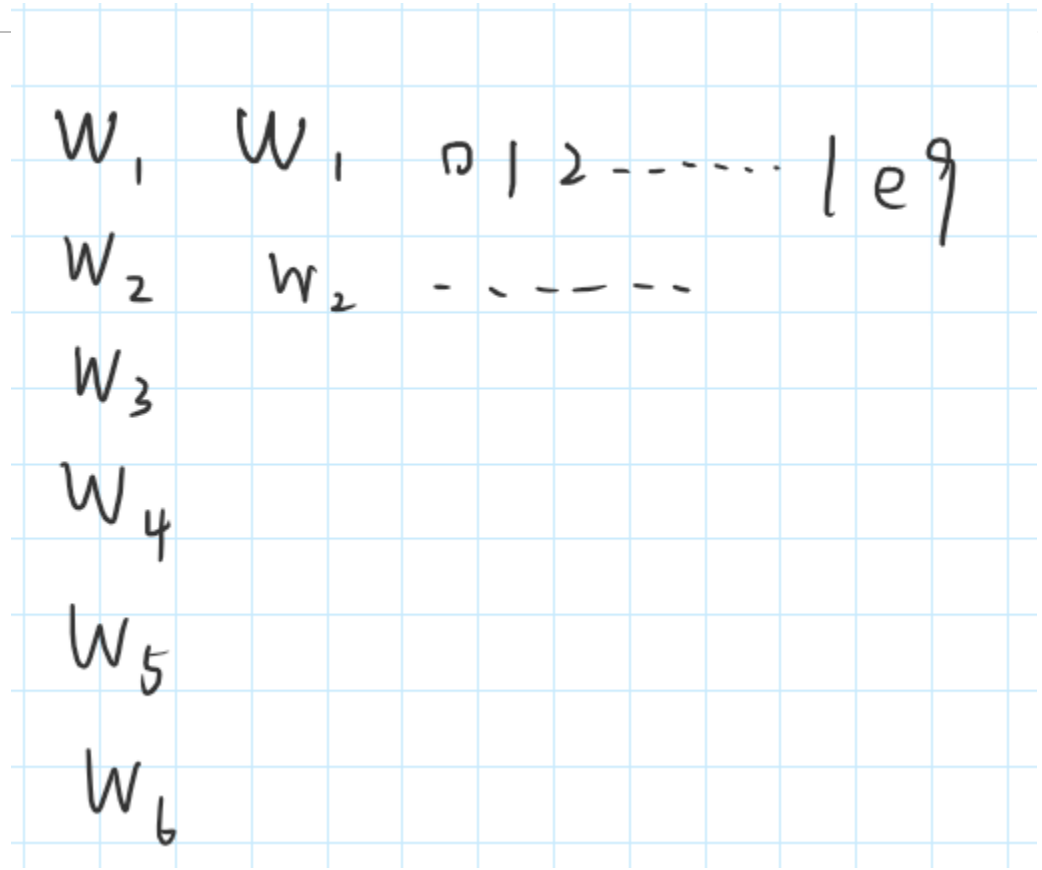
无解的情况：所有0权值边都不动，dis[t]仍大于L或所有0权值边全赋最大值，dis[t]仍小于L。

通过观察，可以看出按某一个顺序增加某一个位置的权值，对于最短路的影响一定是**非减的**

因此可以直接二分我们按某一顺序对这些0权值边操作的总次数。（顺序任意，但选定了就不能改）

每次check里面，跑一次dijkstra，返回dis[t]

复杂度$O(n\log^2 n)$



代码细节：[Submission #203222913 - Codeforces]

J. Elden Ring

International Collegiate Programming Contest

2021 Asia-East Continent Final

Problem J

Elden Ring

Prof. Pang is getting addicted to the game called Elden Ring, in which the world is a connected graph including n vertices indexed from 1 to n and m undirected edges. Players start at vertex 1 and travel across the world to slay the god on vertex n.

However, it's not that easy. For any vertex i except vertex 1, there is exactly one boss whose level is l_i, and the player starts the game with level l_1. For each day, the player can travel to any vertex i from vertex 1 and challenge the boss there. If the current level of the player is greater than the boss, the boss will be eliminated from the world (inactivated) and the level of the player will be increased by A. Notice that traveling through a vertex that has an active boss is forbidden. (In other words, Prof. Pang can travel from vertex 1 to vertex i if there is a path in the graph from vertex 1 to vertex i such that each vertex on this path, except for vertex i, has no active boss.) Meanwhile, at the beginning of each day, all the remaining bosses in the world will also be promoted by B levels.

To finish a playthrough of the game, you need to slay the boss on vertex n (Elden Beast). Given the information of the world, Prof. Pang is wondering how many days he needs at least to do so.

The Player can only challenge one boss each day.

Input

The first line contains a single integer T $(1 \le T \le 10^5)$ denoting the number of test cases.

For each test case, the first line includes four integers n, m, A, B $(1 \le n, m, A, B \le 2 \times 10^5)$. In next m lines, each line contains two integers a_i, b_i $(1 \le a_i, b_i \le n)$, denoting the endpoints of the i-th undirected edge. The last line contains n integers l_i $(1 \le l_i \le 2 \times 10^5)$, representing the initial levels of the player and bosses mentioned above.

It is guaranteed that the sum of n over all test cases will not exceed 10^6 and the sum of m over all test cases will not exceed 10^6.

[Problem - J - Codeforces]

J. Elden Ring

给你一个无向边构成的图，你在1点，终极BOSS在n号点，其他的点上都是小boss，然后会给你每个点的经验值，1号点就是自身经验，否则是boss的经验。

然后你每天可以攻击一个与你占领点相邻的boss，初始占领点只有1号点，然后你如果战胜了i号点，那么这个点也就成为了你的占领点。

如果你的经验严格大于这个boss，你就可以战胜他并获得A的经验，每天开始的时候所有boss都会获得B经验，问你多少天可以消灭终极BOSS，如果永远不可以，输出-1

Input

The first line contains a single integer T $(1 \le T \le 10^5)$ denoting the number of test cases.

For each test case, the first line includes four integers n, m, A, B $(1 \le n, m, A, B \le 2 \times 10^5)$. In next m lines, each line contains two integers a_i, b_i $(1 \le a_i, b_i \le n)$, denoting the endpoints of the i-th undirected edge. The last line contains n integers l_i $(1 \le l_i \le 2 \times 10^5)$, representing the initial levels of the player and bosses mentioned above.

It is guaranteed that the sum of n over all test cases will not exceed 10^6 and the sum of m over all test cases will not exceed 10^6.

J. Elden Ring

分三种情况讨论

A = B 直接跑最短路
A < B 预处理一下，直接跑最短路

*A > B 预处理两下，直接跑最短路

用一个c[i]数组存放杀掉当前怪需要的时间

Submission #100203 - QOJ.ac

Thanks