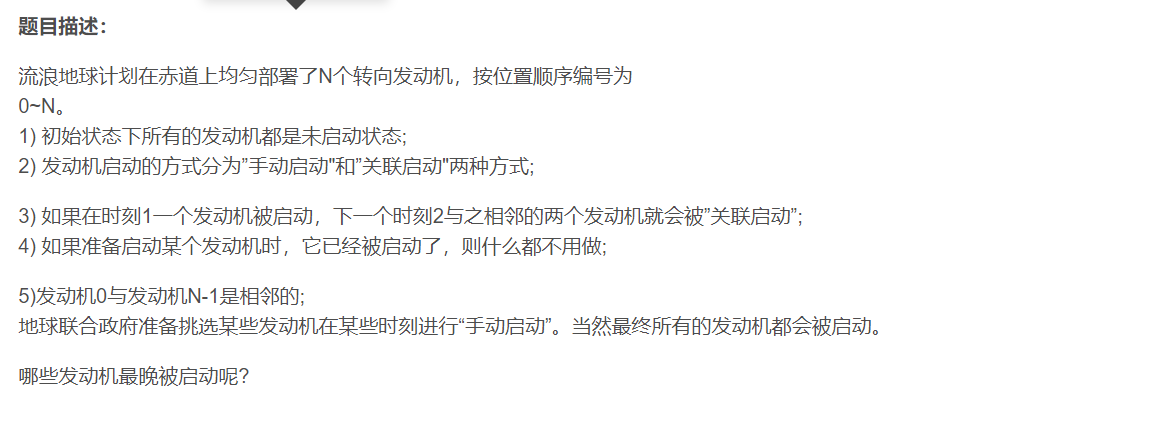
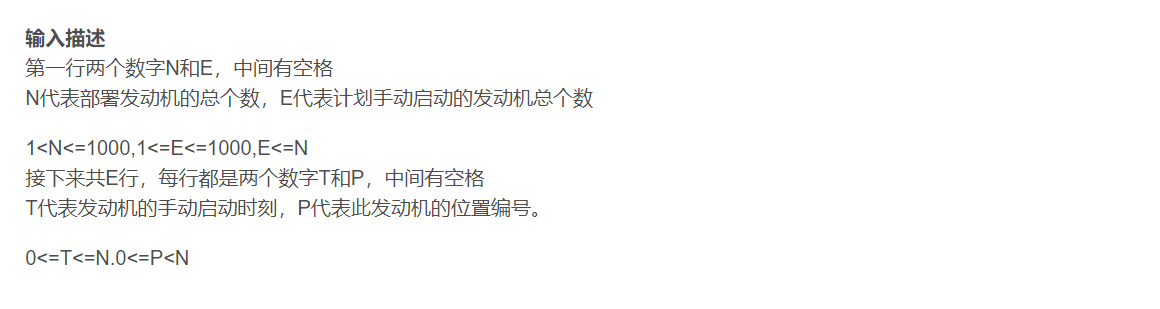
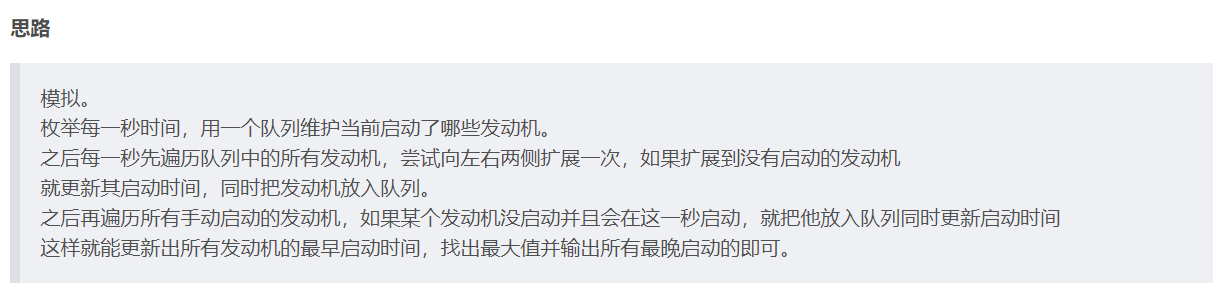
# **E卷-流浪地球[100分]（ Java | Python3 | C++ | C语言 | JsNode | Go）**











import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

int e = sc.nextInt();

int[] ans = new int[n];// 存每个发动机最早启动时间

Arrays.fill(ans, -1);

List<int[]> a = new ArrayList<>();

for (int i = 0; i < e; i++) {

int x = sc.nextInt();

int y = sc.nextInt();

a.add(new int[] { y, x });// 存输入的信息

}

Queue<Integer> q = new LinkedList<>();// 用一个队列维护当前启动了哪些发动机。

for (int t = 0; t <= 5000; ++t) {// 枚举每一秒时间

int sz = q.size();// 当前队列大小

for (int j = 0; j < sz; ++j) {// 循环队列大小次

int f = q.poll();// 出队

int x = (f - 1 + n) % n;// 左侧相邻发动机

int y = (f + 1) % n;// 右侧相邻发动机

if (ans[x] == -1) {// 未启动则启动并放入队列

q.add(x);

ans[x] = ans[f] + 1;

}

if (ans[y] == -1) {// 未启动则启动并放入队列

q.add(y);

ans[y] = ans[f] + 1;

}

}

for (int i = 0; i < a.size(); ++i) {// 遍历所有手动启动的发动机

int P = a.get(i)[0], T = a.get(i)[1];

if (T == t && ans[P] == -1) {// 如果某个发动机没启动并且会在这一秒启动，就把他放入队列同时更新启动时间

q.add(P);

ans[P] = t;

}

}

}

int c = 0;

int max = 0;

for (int i = 0; i < n; i++) {

max = Math.max(max, ans[i]);// 找最晚时间

}

for (int i = 0; i < n; i++) {

if (max == ans[i]) {// 找符合最晚时间的个数

c += 1;

}

}

System.out.println(c);// 输出个数

int f = 0;

for (int i = 0; i < n; i++) {

if (ans[i] == max) {// 输出具体序号

if (f != 0) {

System.out.print(" ");

}

System.out.print(i);

f = 1;

}

}

System.out.println();

}

}

#include <iostream>

#include <vector>

#include <queue>

#include <algorithm>

using namespace std;

int main() {

int n, e;

cin >> n >> e;

vector<int> ans(n, -1);

vector<pair<int, int>> a;

for (int i = 0; i < e; i++) {

int x, y;

cin >> x >> y;

a.push\_back({y, x});

}

queue<int> q;

for (int t = 0; t <= 5000; ++t) {

int sz = q.size();

for (int j = 0; j < sz; ++j) {

int f = q.front();

q.pop();

int x = (f - 1 + n) % n;

int y = (f + 1) % n;

if (ans[x] == -1) {

q.push(x);

ans[x] = ans[f] + 1;

}

if (ans[y] == -1) {

q.push(y);

ans[y] = ans[f] + 1;

}

}

for (int i = 0; i < a.size(); ++i) {

int P = a[i].first, T = a[i].second;

if (T == t && ans[P] == -1) {

q.push(P);

ans[P] = t;

}

}

}

int c = 0;

int mx = 0;

for (int i = 0; i < n; i++) {

mx = max(mx, ans[i]);

}

for (int i = 0; i < n; i++) {

if (mx == ans[i]) {

c += 1;

}

}

cout << c << endl;

int f = 0;

for (int i = 0; i < n; i++) {

if (ans[i] == mx) {

if (f != 0) {

cout << " ";

}

cout << i;

f = 1;

}

}

cout << endl;

return 0;

}

// 8 2

// 0 2

// 0 6

// 2

// 0 4



package main

import (

"fmt"

)

func main() {

var n, e int

fmt.Scan(&n, &e)

ans := make([]int, n)

for i := range ans {

ans[i] = -1

}

a := make([][2]int, e)

for i := 0; i < e; i++ {

fmt.Scan(&a[i][1], &a[i][0])

}

q := make([]int, 0)

for t := 0; t <= 5000; t++ {

sz := len(q)

for j := 0; j < sz; j++ {

f := q[0]

q = q[1:]

x := (f - 1 + n) % n

y := (f + 1) % n

if ans[x] == -1 {

q = append(q, x)

ans[x] = ans[f] + 1

}

if ans[y] == -1 {

q = append(q, y)

ans[y] = ans[f] + 1

}

}

for i := 0; i < e; i++ {

P, T := a[i][0], a[i][1]

if T == t && ans[P] == -1 {

q = append(q, P)

ans[P] = t

}

}

}

c := 0

max := 0

for i := 0; i < n; i++ {

if ans[i] > max {

max = ans[i]

}

}

for i := 0; i < n; i++ {

if ans[i] == max {

c++

}

}

fmt.Println(c)

f := false

for i := 0; i < n; i++ {

if ans[i] == max {

if f {

fmt.Print(" ")

}

fmt.Print(i)

f = true

}

}

fmt.Println()

}



from collections import deque

n, e = map(int, input().split())

ans = [-1] \* n

a = []

for \_ in range(e):

y, x = map(int, input().split())

a.append((x, y))

q = deque()

for t in range(5001):

sz = len(q)

for \_ in range(sz):

f = q.popleft()

x = (f - 1 + n) % n

y = (f + 1) % n

if ans[x] == -1:

q.append(x)

ans[x] = ans[f] + 1

if ans[y] == -1:

q.append(y)

ans[y] = ans[f] + 1

for P, T in a:

if T == t and ans[P] == -1:

q.append(P)

ans[P] = t

max\_val = max(ans)

c = ans.count(max\_val)

print(c)

result = [i for i in range(n) if ans[i] == max\_val]

print(" ".join(map(str, result)))



const rl = require("readline").createInterface({

input: process.stdin,

output: process.stdout,

});

var iter = rl[Symbol.asyncIterator]();

const readline = async () => (await iter.next()).value;

void async function () {

const firstLine = await readline();

const [n, e] = firstLine.trim().split(' ').map(Number);

let ans = Array(n).fill(-1);

let a = [];

for (let i = 0; i < e; i++) {

const line = await readline();

const [x, y] = line.trim().split(' ').map(Number);

a.push([y, x]);

}

let q = [];

for (let t = 0; t <= 5000; t++) {

let sz = q.length;

for (let j = 0; j < sz; j++) {

let f = q.shift();

let x = (f - 1 + n) % n;

let y = (f + 1) % n;

if (ans[x] === -1) {

q.push(x);

ans[x] = ans[f] + 1;

}

if (ans[y] === -1) {

q.push(y);

ans[y] = ans[f] + 1;

}

}

for (let i = 0; i < a.length; i++) {

const [P, T] = a[i];

if (T === t && ans[P] === -1) {

q.push(P);

ans[P] = t;

}

}

}

let max = Math.max(...ans);

let c = ans.filter(v => v === max).length;

console.log(c);

let result = [];

for (let i = 0; i < n; i++) {

if (ans[i] === max) {

result.push(i);

}

}

console.log(result.join(' '));

rl.close();

}();



#include <stdio.h>

#include <stdlib.h>

typedef struct Queue {

int \*data;

int front, rear, size;

} Queue;

void initQueue(Queue \*q, int n) {

q->data = (int \*)malloc(n \* sizeof(int));

q->front = q->rear = -1;

q->size = n;

}

int isEmpty(Queue \*q) {

return q->front == -1;

}

void enqueue(Queue \*q, int value) {

if (isEmpty(q)) {

q->front = 0;

}

q->rear = (q->rear + 1) % q->size;

q->data[q->rear] = value;

}

int dequeue(Queue \*q) {

int value = q->data[q->front];

if (q->front == q->rear) {

q->front = q->rear = -1;

} else {

q->front = (q->front + 1) % q->size;

}

return value;

}

int main() {

int n, e;

scanf("%d %d", &n, &e);

int \*ans = (int \*)calloc(n, sizeof(int));

for (int i = 0; i < n; i++) {

ans[i] = -1;

}

int \*\*a = (int \*\*)malloc(e \* sizeof(int \*));

for (int i = 0; i < e; i++) {

a[i] = (int \*)malloc(2 \* sizeof(int));

scanf("%d %d", &a[i][1], &a[i][0]);

}

Queue q;

initQueue(&q, 500005);

for (int t = 0; t <= 5000; ++t) {

int sz = (q.rear - q.front + q.size) % q.size;

for (int j = 0; j < sz; ++j) {

int f = dequeue(&q);

int x = (f - 1 + n) % n;

int y = (f + 1) % n;

if (ans[x] == -1) {

enqueue(&q, x);

ans[x] = ans[f] + 1;

}

if (ans[y] == -1) {

enqueue(&q, y);

ans[y] = ans[f] + 1;

}

}

for (int i = 0; i < e; ++i) {

int P = a[i][0], T = a[i][1];

if (T == t && ans[P] == -1) {

enqueue(&q, P);

ans[P] = t;

}

}

}

int c = 0;

int mx = 0;

for (int i = 0; i < n; i++) {

if (mx < ans[i]) {

mx = ans[i];

}

}

for (int i = 0; i < n; i++) {

if (mx == ans[i]) {

c++;

}

}

printf("%d\n", c);

int f = 0;

for (int i = 0; i < n; i++) {

if (ans[i] == mx) {

if (f != 0) {

printf(" ");

}

printf("%d", i);

f = 1;

}

}

printf("");

return 0;

}