10월 3주차 스터디 - 1조

멤버 : 김석원, 김중재, 우정연, 허범

발표일시 : 2021.10.22 21시

문제

- SWEA 4724 견우와 직녀
  https://swexpertacademy.com/main/code/userProblem/userProblemDetail.do?contestProbId=AWSHOpR6f_sDFARw&
- 개인적으로 풀어보고 싶은 문제 1개

풀이

김석원

10026번 적록색약

10026번: 적록색약 (acmicpc.net)

```
import java.io.BufferedReader; import java.io.IOException; import
java.io.InputStreamReader; import java.util.LinkedList; import
java.util.Queue; import java.util.concurrent.LinkedBlockingDeque; public
class Main { static int N; static char[][] board; static boolean[][]
visit; static char alph; static int[] dx = {0,1,0,-1}; static int[] dy =
{1,0,-1,0}; static void pb() { for(char[] a : board) { for(char b : a) {
System.out.print(b); } System.out.println(); } } static void printvisit()
{ for(boolean[] vis : visit) { for(boolean v : vis) { int tmp = v == true
? 1 : 0; System.out.print(tmp); } System.out.println(); } } public static
void main(String[] args) throws NumberFormatException, IOException {
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
N = Integer.parseInt(br.readLine()); board = new char[N][N]; visit = new
boolean[N][N]; for(int i = 0; i < N; i++) { board[i] =
br.readLine().toCharArray(); } int count = 0; for(int i = 0; i < N; i++)
{ for(int j = 0; j < N; j++) { if(!visit[i][j]) { count++; bfs(i,j); } }
} System.out.print(count + " "); count = 0; visit = new boolean[N][N];
for(int i = 0; i < N; i++) { for(int j = 0; j < N; j++) { if(!visit[i]
[j]) { count++; bfs2(i,j); // printvisit(); } } }
System.out.println(count); } static void bfs(int x,int y) { alph =
board[x][y]; visit[x][y] = true; Queue<point> queue = new
LinkedList<point>(); queue.offer(new point(x,y)); while(!queue.isEmpty())
{ point cur = queue.poll(); for(int d = 0; d < 4; d++) { int nx = dx[d] +
cur.x; int ny = cur.y + dy[d]; if(nx >= 0 && nx < N && ny >= 0 && ny < N)
{ if(board[nx][ny] == alph && !visit[nx][ny]) { visit[nx][ny] = true;
queue.offer(new point(nx, ny)); } } } } } //R G 는 같고 B만 다르다. static
void bfs2(int x,int y) { int flag = 0; alph = board[x][y]; if(alph ==
'B') flag = 1; visit[x][y] = true; Queue<point> queue = new
LinkedList<point>(); queue.offer(new point(x,y)); while(!queue.isEmpty())
{ point cur = queue.poll(); for(int d = 0; d < 4; d++) { int nx = dx[d] +
cur.x; int ny = cur.y + dy[d]; if(nx >= 0 && nx < N && ny >= 0 && ny < N)
{ if(flag == 1 && board[nx][ny] == 'B' && !visit[nx][ny]) { //
System.out.print(board[nx][ny]); visit[nx][ny] = true; queue.offer(new
point(nx, ny)); }else if(flag == 0 && board[nx][ny] != 'B' && !visit[nx]
[ny]) { // System.out.print(board[nx][ny]); visit[nx][ny] = true;
queue.offer(new point(nx, ny)); } } } } } static class point{ int x,y;
public point(int x, int y) { super(); this.x = x; this.y = y; } } }
```

▼ 풀이과정

BFS , DFS를 사용하는 문제중 RGB각각 구분하는것과 RG, B로 구분만 해주는 로직만 추가된 문제

김중재

백준 1197 - 최소 스패닝 트리

- 최소 스패닝 트리 = 최소 신장 트리

> 💡 최소 신장 트리란, 모든 정점이 이어져 있고 간선(**가중치**)의 합이 최소인 트리를 말함.

- 특징 : 간선의 갯수(E) = 정점의 수(V) - 1
- 관련 알고리즘 : 1. 크루스칼 , 2. 프림

1. **크루스칼 알고리즘 (간선 중심)**
    a. 간선(**가중치**)을 기준으로 정렬. → 이 과정에서 **우선순위 큐**를 사용해도 좋음.
    b. 정렬된 간선을 연결하면서 사이클 체크. → union-find 알고리즘 활용.
    c. 간선의 갯수(E) = 정점의 수(V) - 1 되는 시점에 종료.
    d. 최소 신장 트리 완성
2. **프림 알고리즘 (정점 중심)**
    a. 시작점을 랜덤으로 잡고 시작.
    b. BFS 처럼 인접 정점 탐색하고 간선의 **가중치**가 가장 작은 정점 선택.
    → 이 과정에서 **우선순위 큐** 사용 권장.
    c. b의 과정 반복.
    d. 간선의 갯수(E) = 정점의 수(V) - 1 되는 시점에 종료.
    e. 최소 신장 트리 완성

- **풀이**

1. 크루스칼 알고리즘을 활용했음.
2. 입력으로 들어온 간선의 정보를 클래스(Node)를 만들어 저장했고, Comparable 인터페이스로 정렬했을 시 가중치를 기준으로 하게함.
3. 정렬된 간선을 연결하면서 사이클 체크. + 가중치를 더해줌.→ union-find 알고리즘 활용.
4. 간선의 갯수(E) = 정점의 수(V) - 1 되는 시점에 종료.
5. 더한 가중치를 출력

▶ **코드**

우정연

- 견우와 직녀
  - Thinking) 원숭이, 벽뚫기 문제와 비슷하나, 쉬는시간때는 못지나가는게 다르네
    - 고려사항 : 오작교 두번 연속으로 못감
      오작교 쉬는시간 T일 때 0, T, 2T... 이 때만 지나갈 수 있음
      다리 한 번 놓을 수 있음
    - t % T == 0일 때만 해당 칸으로 갈 수 있고, 아니면 다시 now를 queue에 넣음
    - visit에 다리 만들 수 있는 횟수 큰값 저장되도록(-1(초기값), 0(다리 만들었음), 1(다리 아직 안만듬))
  - Method : BFS(이동하는 최소 시간)
  - Error1 : now가 여러번 들어감 -> now칸에서 상하좌우에 대해 오작교나 절벽(다리 아직 안만들어서 건너갈 수 있을 때)이 여러개면 now 여러번 들어가면서 쉬는시간동안 증식...
  - Error2 : 런타임 에러 -> 또 증식중... -> visit[ni][nj] >= now.cnt 같을 때도 패스해야 함(다시 방문하지 않도록)
  - Error3 : 각 오작교 쉬는시간 다 끝나지도 않았는데 지나가버림 -> 오작교인 경우, (time + 1) % map[ni][nj] == 0로 해야 함(% M 했었음..)
  - Error4 : 오작교 연속으로 두번 못건넘

허범

- 나이트의 이동
  - 이 문제는 나이트의 최소 이동 경로를 찾는 문제.
  - 최소 이동경로를 찾아야 하기 때문에 BFS를 이용해서 해결.
  - 현재 위치를 저장 후, 모든 방향(8방향)으로 이동하며 탐색.
  - 각 정점에 도달하면 배열에 이동횟수를 저장하고, 목표지점에 도달하면 이동횟수 출력.