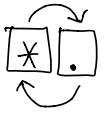
- 문제 · 가로 w, 세로 h ($1 \le w, h \le 20$)
  · 먼지 '*' (최대 10개)
  · 벽 'x', 나머지 '.'
  · 모든 먼지 치우는 최단경로 길이.

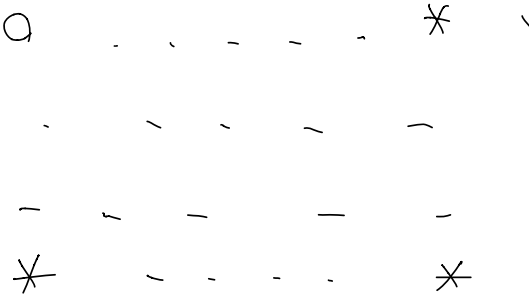
- 접근 · 처음에는 너무어렵게 생각했다..
  3차원 배열 [먼지수][y][x] 사용해서 (visited)
  현재까지 온데미 먼지 0, 1, 2, ... 각각 최소값을 좌표에
  갱신하여 먼지 개수별로 독립적으로 방문수 최소값을 유지하려고 했다.
  ⇒ 문제① 3차원 배열은 먼지 수 기준으로 각 층이라고 하면
  먼저 0층에서 1층으로 도착한 노드가 1층의 좌표값을
  최소값으로 갱신하여 두번째로 1층에 도착한
  노드는 더이상 진행할 수 없게됨 (이것이 최적
  일지라도).

  ② 

  ← 다시 방문했던 먼지로 돌아가서
  새로운 먼지라고 받아들임. (bfs 특성상..)
  (1층 → 2층으로 받아들여서 아직 갱신 x인
  [cnt+1][nx][ny] 로 이동해버림).

  ⇒ 그래서 큐에 들어가는 노드에 먼지 bool로
  방문체크하는 vector도 넣어줄 ... (모듯짓).
  ↳ 문제 ①은 여전히 해결x.

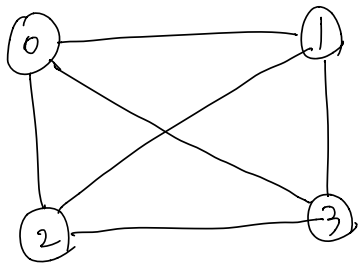
· 간단한 방법이 있었음.



⇒ '0'을 출발점으로 하여
각 '*'을 하나의 노드로 보면
각 노드 기준으로 다른 노드까지
의 상대거리 (bfs를 노드별로
돌리면 최단거리 쉽게 얻음)
을 얻을 수 있다.

⇒ 좌표 최단거리 문제 ⟶ 노드 최단거리 문제로
(지, 나 개념) (그 만큼제)

간단해짐 (각 노드 간의 가중치를 얻을 수 있게됨)
↳ 그래프를 얻은것. (완전 그래프 !!)



이제 dfs 통해 0에서 시작하는 최단 경로 구하면 끝.

* dfs을 재귀 아닌 permutation으로 구현 가능.

```
do {
    sum += w[0][v[0]];
    sum += w[i-1][i];
    :
} while ( next_permutation ( v.begin(), v.end() );
```

↳ v는 {1, 2, 3, ··, dust}
로 초기화하고 시작.

* dfs에서
if (sum > ans) return;
↳ 이 한줄로 10배이상 빨라짐.

* 각 노드별 bfs는 visited 값 -1로 초기화하고
기준점을 0으로 하고 -1만 방문하며 진행하면 너무 간단.

반복문 탈출 후에 아직 ~인 점이 있다면 ~출력 → 종료.