**★★★☆☆**

**題組：Problem Set Archive with Online Judge**

**題號：11487: Gathering Food**

**解題者：葉杰明**

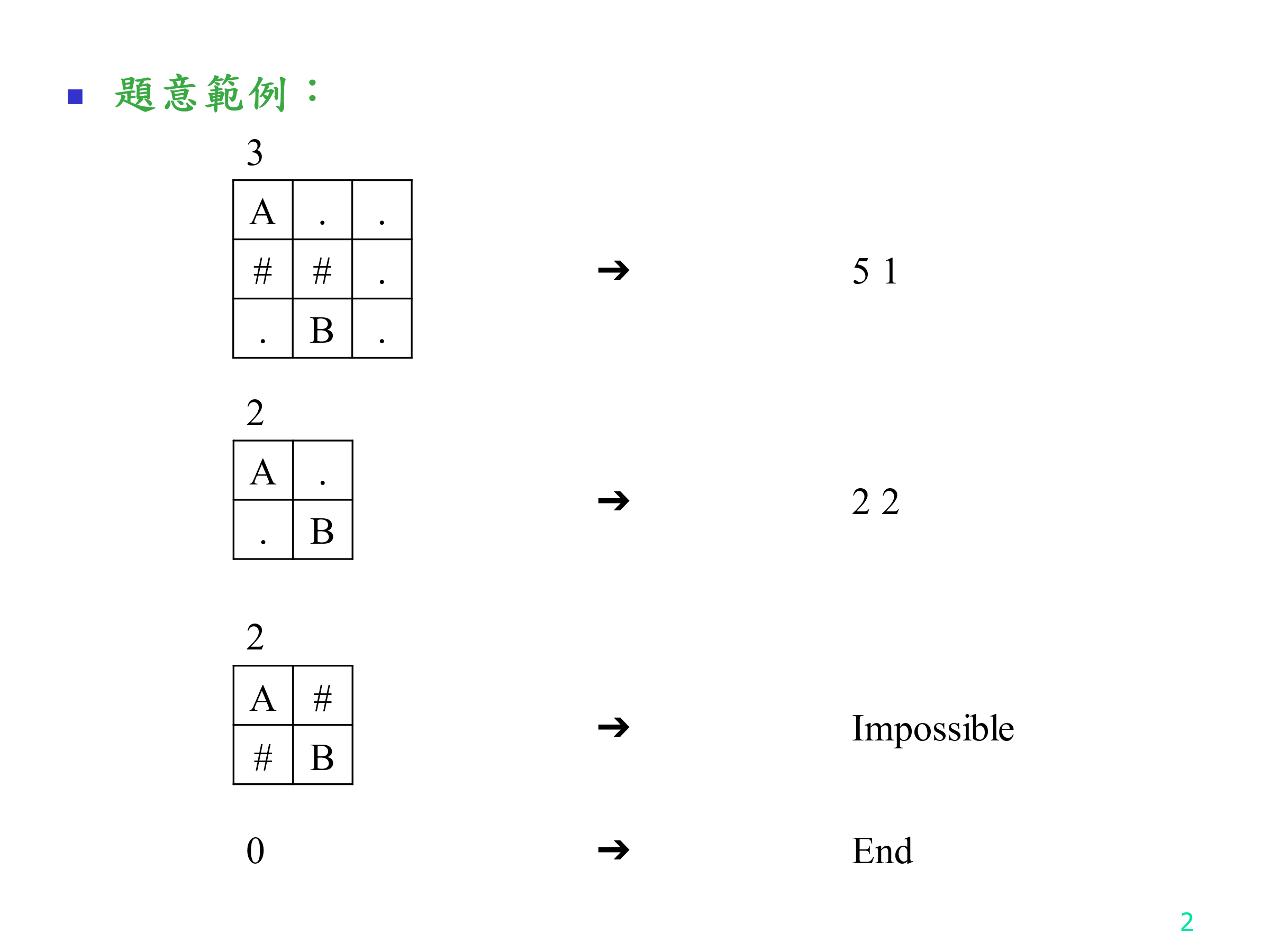
**解題日期：2024年5月16日**

**題意：**

給定的方格，方格內容由‘.’(代表空的空間)、‘#’(代表障礙)、英文字母(A~Z，代表食物)。

移動方向只能為上下左右，若取得食物後則將方格內容改為’.’，起點為字母A，求按照英文字母順序取的食物的最短路徑長以及(最短路徑的數量%20437)。

**題意範例：**



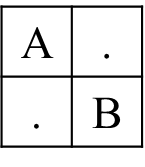
**解法：**

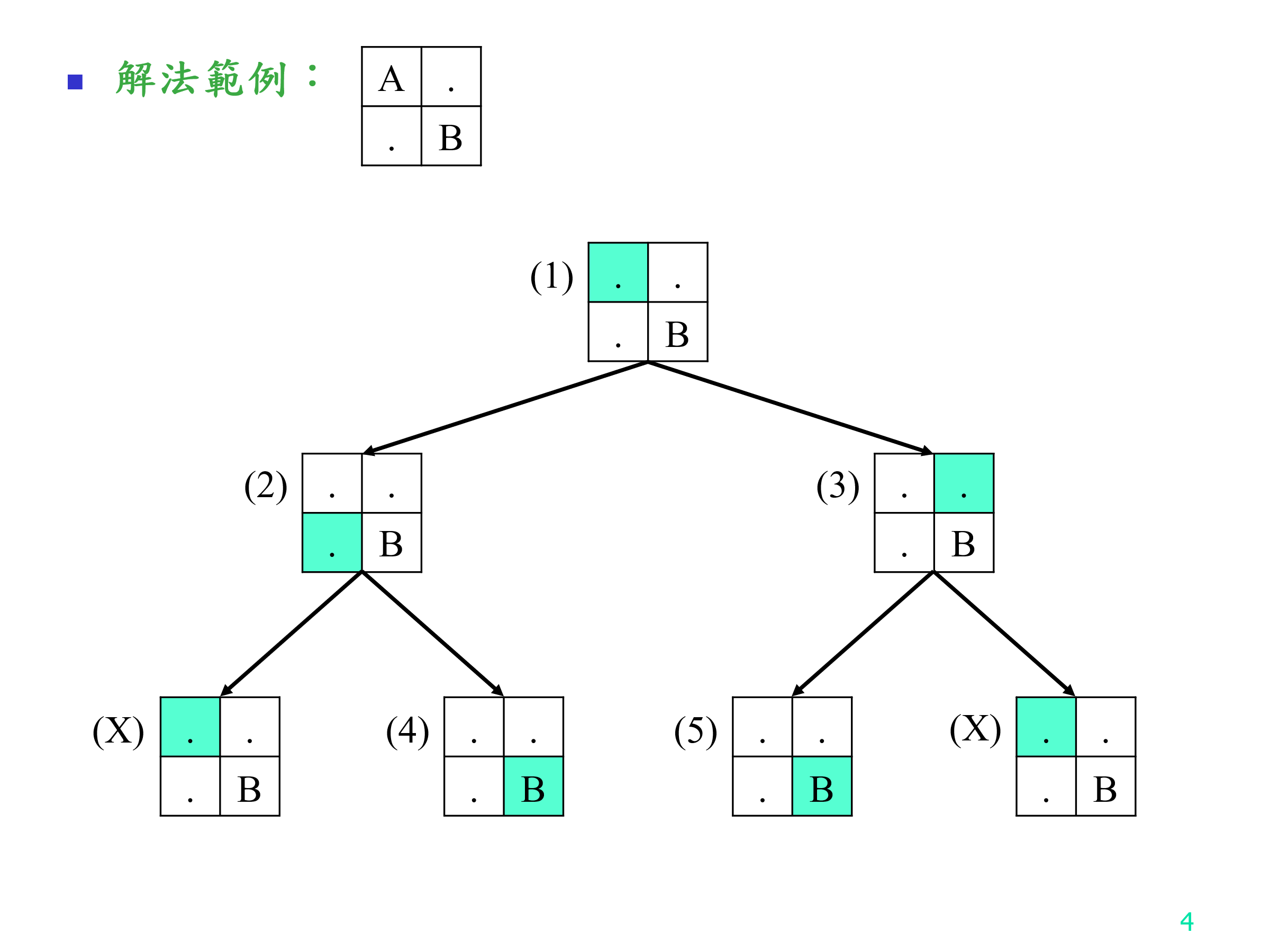
分別找出兩順序字母之間的最短路徑長，以及最短路徑數，由BFS搭配distance[N][N]紀錄從起點到每個點的最短路徑長以及path[N][N]紀錄從起點到每個點的最短路徑數。若，則直接輸出“Impossible”。

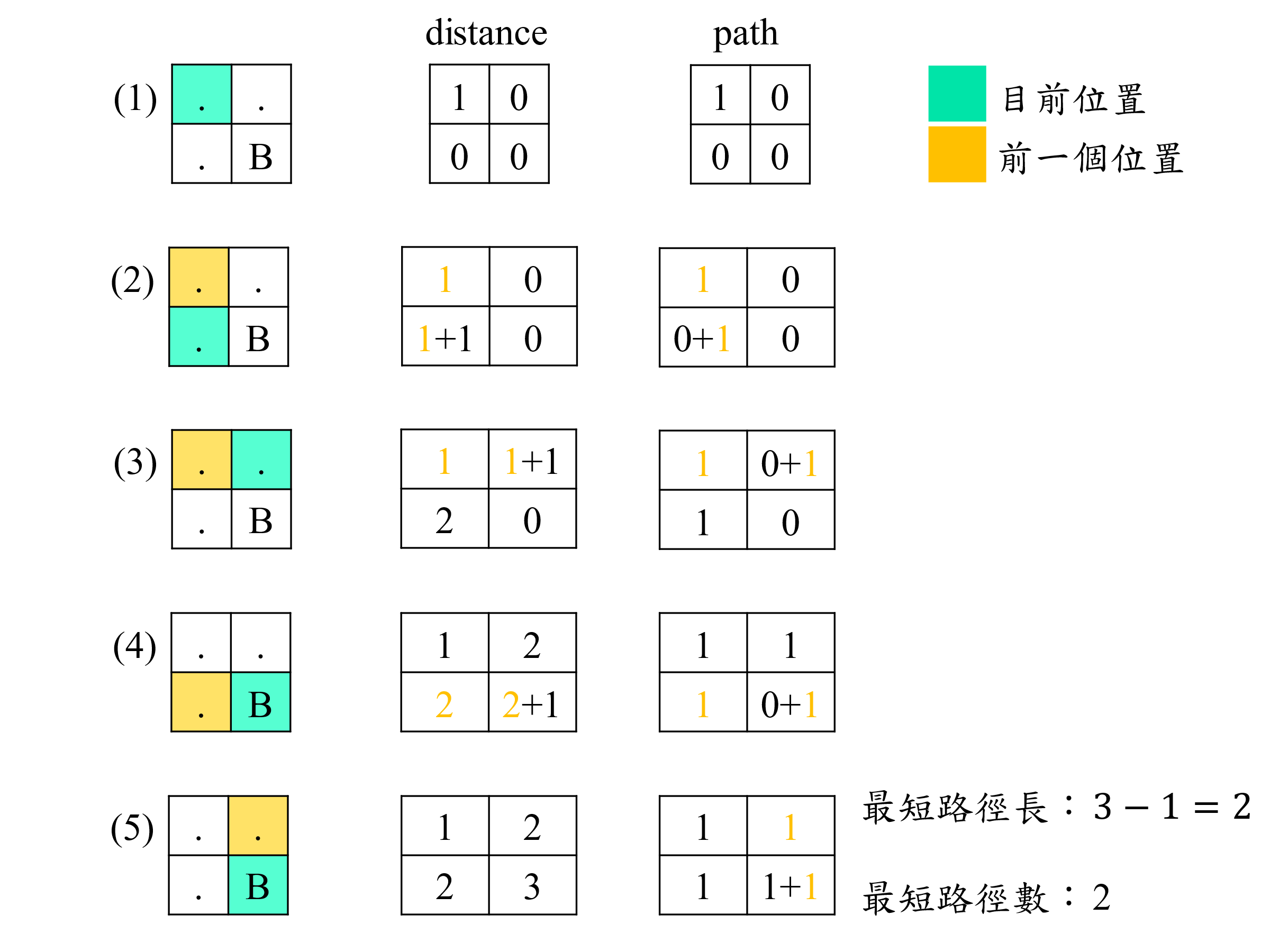
最短路徑長：

最短路徑數：

**解法範例：**

****





**討論：**

若使用DFS，會找出所有可能的路徑，對尋找最短路徑比較沒有效率。

**程式：**

#include <iostream>

#include <cstring>

#include <queue>

#include <cctype>

using namespace std;

#define MOD 20437

struct pos

{ int i, j; };

pos dir[4] = { {-1, 0}, {0, 1}, {1, 0}, {0, -1} };

int main()

{

int N, t = 0;

char grid[10][10] = {};

while (cin >> N && N)

{

pos letter[26] = {};

int letter\_count = 0;

for (int i = 0; i < N; i++)

{

for (int j = 0; j < N; j++)

{

cin >> grid[i][j];

if (isalpha(grid[i][j]))

{

letter[grid[i][j] - 'A'] = {i, j};

letter\_count++;

}

}

}

int total\_dist = 0, total\_path = 1;

bool flag = true;

for (int i = 1; i < letter\_count; i++)

{

int dist[10][10];

memset(dist, 0, sizeof(dist));

int path[10][10];

memset(path, 0, sizeof(path));

dist[letter[i - 1].i][letter[i - 1].j] = 1;

path[letter[i - 1].i][letter[i - 1].j] = 1;

grid[letter[i - 1].i][letter[i - 1].j] = '.';

queue<pos> pos\_Q;

pos\_Q.push(letter[i - 1]);

char next\_alpha = 'A' + i;

while (!pos\_Q.empty())

{

pos cur = pos\_Q.front();

pos\_Q.pop();

for (int j = 0; j < 4; j++)

{

pos next = { cur.i + dir[j].i, cur.j + dir[j].j };

if (next.i < 0 || next.i >= N || next.j < 0 || next.j >= N)

continue;

if (grid[next.i][next.j] == '#')

continue;

if (grid[next.i][next.j] != '.' && grid[next.i][next.j] != next\_alpha)

continue;

if (!dist[next.i][next.j])

{

dist[next.i][next.j] = (dist[cur.i][cur.j] + 1);

if (grid[next.i][next.j] != next\_alpha)

pos\_Q.push(next);

}

if (dist[next.i][next.j] == dist[cur.i][cur.j] + 1)

path[next.i][next.j] = (path[cur.i][cur.j] + path[next.i][next.j]) % MOD;

}

}

if (!dist[letter[i].i][letter[i].j])

{

flag = false;

break;

}

total\_dist += dist[letter[i].i][letter[i].j] - 1;

total\_path = (total\_path \* path[letter[i].i][letter[i].j]) % MOD;

}

cout << "Case " << ++t << ": ";

if (!flag)

cout << "Impossible\n";

else

cout << total\_dist << ' ' << total\_path << endl;

}

return 0;

}