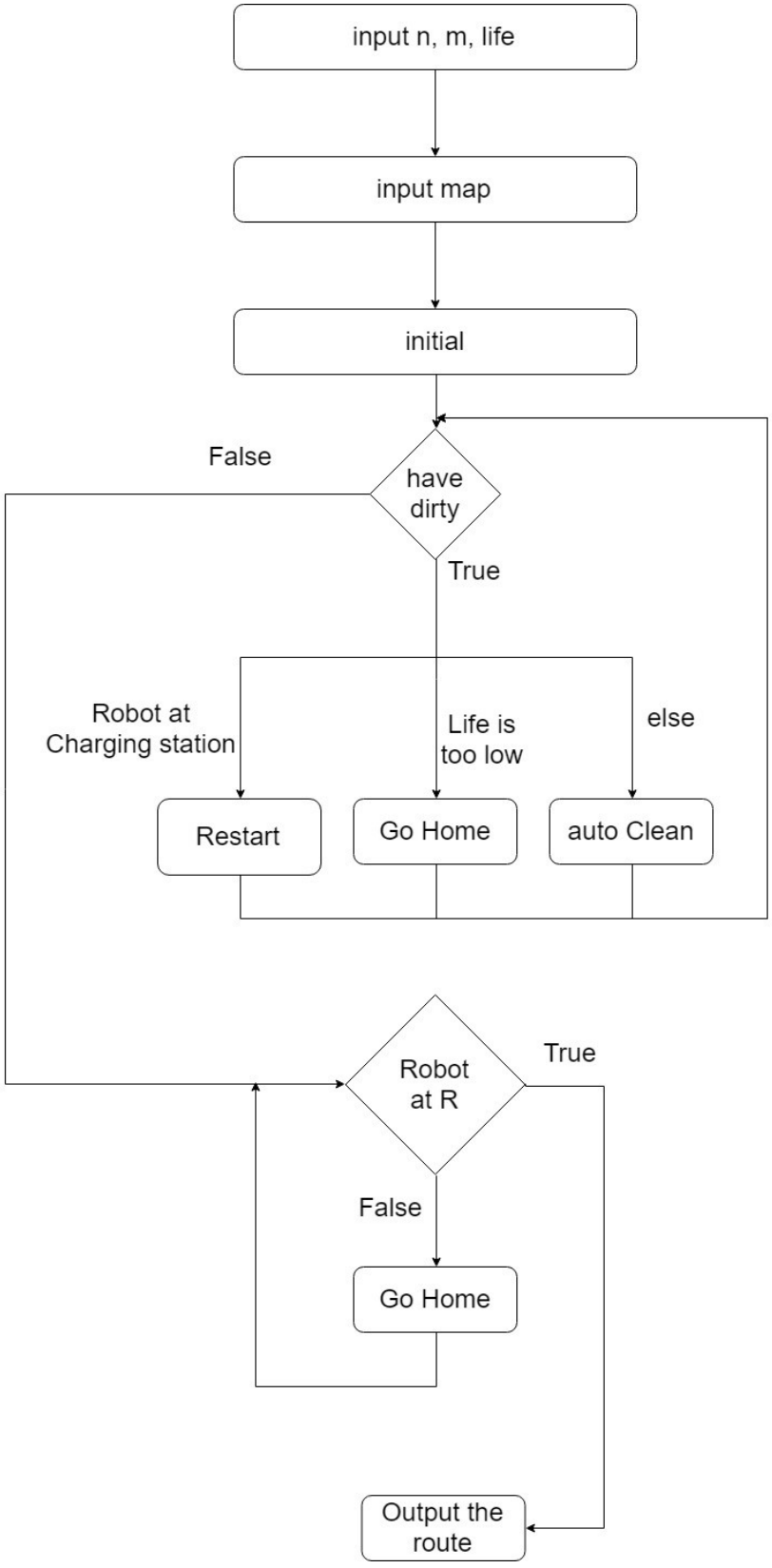
Project 2 Floor Cleaning Robot

106062130 邱詠智

1. Project Description
   1) Flow Chart

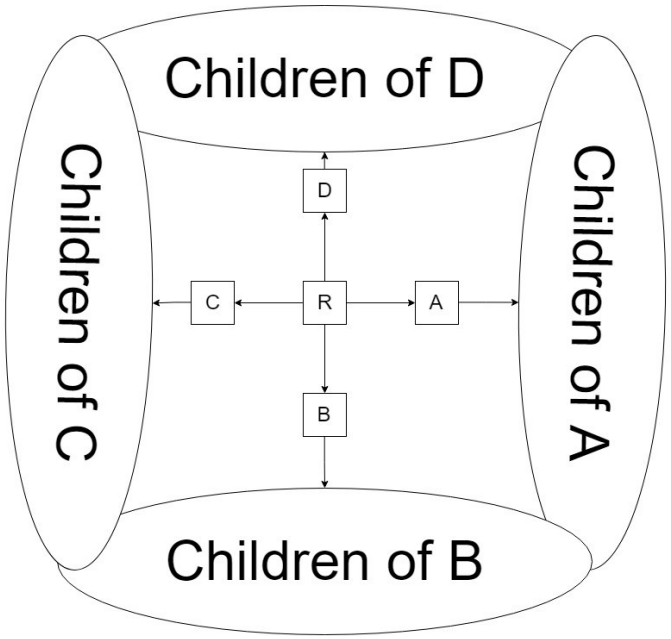


2) Detailed Description

a) 策略概要：

把圖分成 4 個區域

示意圖




1-st 出發：可從任意方向離開 R，任意方向進入 R

k-th 出發：從 (k-1) th 的進入方向離開，清掃完 children 前不更換出入方向。

Ex: 假設 1-st 從 A 回來

children of A 清完後，經由 ABCD 間的最短路更換出入方向 restart

理論上至多更換 3 次出入方向。

b) Initial:

- BFS

  Charging Station ( R ) = 0, Free space ( 0 ) = 0, Obstacle ( 1 ) = 1e9

  第一次 BFS，Free space 的位置會存入到 R 的最短距離。

  以正數表示尚未清理，負數表示已清理。

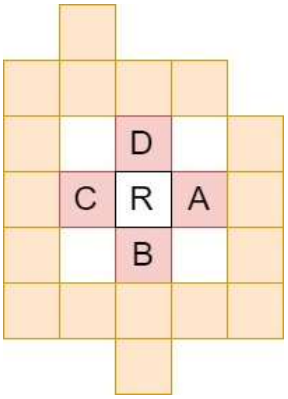
  Ex: 圖中省略了牆壁

| | | | | |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | R | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |

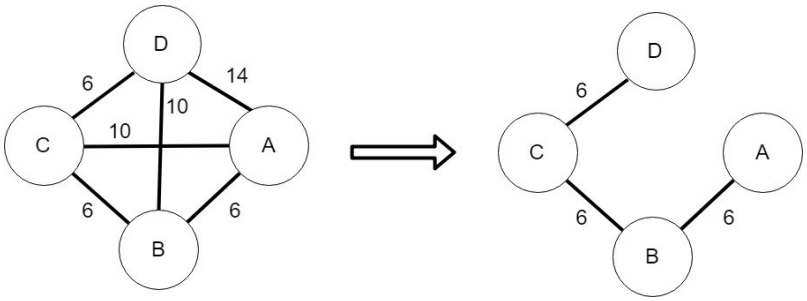
⇨

| | | | | |
|---|---|---|---|---|
| | 4 | | | |
| 4 | 3 | 2 | 3 | |
| 3 | | 1 | | 3 |
| 2 | 1 | 0 | 1 | 2 |
| 3 | | 1 | | 3 |
| 4 | 3 | 2 | 3 | 4 |
| | | 3 | | |

- Shortest path between each entry
  一張圖中有 1 ~ 4 個進出 R 的出入口。
  用 BFS 做出各出入口間的「不充電」的最短路徑，把長度過長（超出電量上限）的路徑排除。
  再以 Floyd-Warshall 計算由出入口 A → B 的最短路（途中可充電）
  Ex: 假設最大電量 = 8,
  D → A: 必須經過 C and B 且中途要充電




( 黃色 = Free Space, 白色 = Obstacle )




c) Robot mode 1: Go home
   從當前位置搜尋相鄰點，走向離 R 較近的地方(1~2 個)。
   若其中 1 個尚未清理，則走向未清理的地方。
   觸發條件：當前 life ≤ 當前位置與 R 的最短距離。

d) Robot mode 2: auto Clean
   走向未清理的相鄰地區，離 R 越遠，優先度越高。
   若四周都已清理過，則執行 Go home。
   觸發條件：當前 life ＞ 當前位置與 R 的最短距離，且不在 R 內。
   ( 不符合 mode 1 or mode 3 時觸發 )

e) Robot mode 3: restart
   以動態規劃的技巧，計算由此地出發，往深處（離 R 較遠）走，能途經「最多未清理區域」的路徑。

定義：

某點的 child = 離 R 較遠的鄰點

$dp[i][j].cnt =$ 由( i, j )往深處走，能途經的未清理區域最大數量。

$dp[i][j].dir =$ 該往哪個方向才能清理較多區域。

對每個座標點 ( i, j ) 可列出以下兩個式子：

$$\text{if } (i,j)\ is\ \text{Obstacle}, dp[i][j].cnt = 0$$
$$dp[i][j].cnt = max(\ dp[child.x][child.y].cnt) + !\,isClean(i,j)$$

$dp[i][j].cnt$ 刷新時，同時更新 $dp[i][j].dir$

為避免過程中出現自環，只允許從起點往 child 方向刷新 dp 陣列，
根據選擇的起點，可能會出現沒有被建表的區域。

Ex：左圖為原圖、右圖為以 R 為中心的最短距離陣列

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 4 | 5 | | | | 11 | 12 |
| 3 | 4 | | 14 | 13 | 12 | 13 |
| 2 | 3 | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |

以 R 為中心 restart: 則會建立出下方兩張表

左表：dp[i][j].cnt

| | | | | | | |
|---|---|---|---|---|---|---|
| 10 | 9 | 8 | 7 | 6 | 5 | 3 |
| 11 | 10 | | | | 4 | 2 |
| 12 | 11 | | 1 | 2 | 3 | 1 |
| 13 | 12 | | | | | |
| 14 | 13 | 5 | 4 | 3 | 2 | 1 |
| 14 | 14 | 6 | 5 | 4 | 3 | 2 |

右表：dp[i][j].dir

| | | | | | | |
|---|---|---|---|---|---|---|
| → | → | → | → | → | ↓ | ↓ |
| → | ↑ | | | | ↓ | ↓ |
| → | ↑ | | X | ← | ← | X |
| → | ↑ | | | | | |
| → | ↑ | → | → | → | → | X |
| → | ↑ | → | → | → | → | ↑ |

藍色 = 清理過的區域、黃色 = 更新 dp 陣列的起點、X = 沒有建表的區域

dp.cnt

| | | | | | | |
|---|---|---|---|---|---|---|
| 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| 3 | 3 | | | | 2 | 2 |
| 3 | 3 | | 0 | 0 | 1 | 1 |
| 3 | 3 | | | | | |
| 5 | 5 | 5 | 4 | 3 | 2 | 1 |
| R | X | X | X | X | X | X |

dp.dir

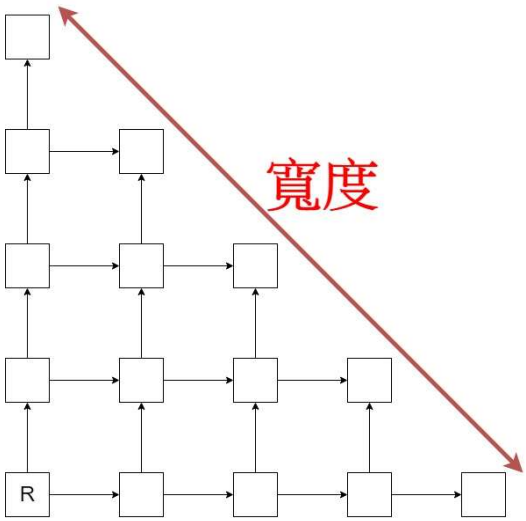
| | | | | | | |
|---|---|---|---|---|---|---|
| → | → | → | → | → | → | ↓ |
| → | ↑ | | | | → | ↓ |
| → | ↑ | | X | X | → | X |
| → | ↑ | | | | | |
| → | → | → | → | → | → | X |
| X | X | X | X | X | X | X |

問題 1: 以 R 為中心無法控制出入方向。( 可能導致不符合 spec )
Solution 1: 第 2 次以後的 restart, 以上次進來的地方為中心 restart 並把活動範圍限制在 dp 陣列有更新的地方(dp.cnt 非 X 的區域)

問題 2: 一個出入方向掃不完。
Solution 2: 更換出入方向。
使用 Initial 時建立的「Shortest path between each entry」

3) 效率及複雜度：
   a) Initial: $O(n * m)$ ( input + BFS )
   b) Go home: $O(1)$
   c) Auto Clean: $O(1)$
   d) restart : $O(n * m)$ ( update dp by DFS )

最糟情況下，每次出發到回來充電都走同一條路
如右圖的情況，restart 次數為最大寬度。
$m * n$ 地圖中，最大寬度約為 $(m + n)$



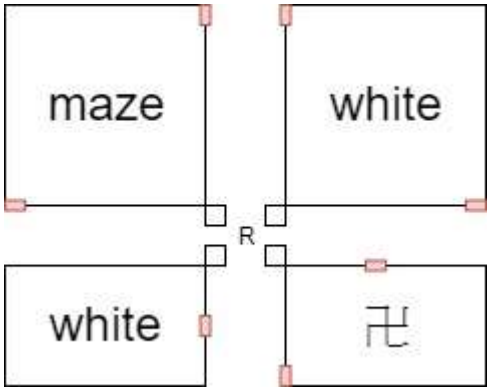

總體時間複雜度：

$$O(\text{restart} * \text{寬度}) = O(m * n(m + n))$$

空間複雜度：
$O$(解答步數 + 出入口間最短路 + dp + 原圖)
$= O(\text{解答步數} + m * n)$
解答步數 $\approx$ 寬度 $*$ 最遠距 + 出入口最短路總和
步數上限 $\approx (m + n)^2$

2. Test Case Design

162 * 200 的矩形，以十字型的道路分成 4 塊。
如右圖所示。 紅色矩形為各區塊的出入口。




a) maze ( 100*100 ) :
各條路寬度為 1, 使用右手定則可以全部走完。
若沒有走到道路底端，且沒有檢查
會產生多餘的步數 or 無法掃完整個迷宮。

b) white ( 100 * 100, 59 * 100 )、卍字型 ( 59 * 100 ) :
基本測試( white )、有點變化的基本測試 ( 卍 )

c) R 的位置、 Battery life = 1009 ( 最遠距離*2 + 1 ):
如果從左邊出來，必須要橫穿整個 maze 才能由上方回去充電。
由於生命值的限制，出入口不能只限制在同一方向。
若為了符合 spec，把出入口限制為一個，那就永遠跑不完。
Ex: 由左方出發，掃完迷宮後必須橫穿迷宮，從上方回去充電。
否則電量不夠支撐到進入左下方的 White.

Battery life =奇數：一趟路徑消耗電量必為偶數，若沒有針對奇數做調整，則走不回來。