X = fallimento (soluzione non individuata entro 15 minuti)

Cella vuota = labirinto non testato

In verde il fatto che la soluzione trovata sia ottima

**Algoritmo A\***

**Labirinto 5x5**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Pochi | Medi | Tanti |
| 2 |  | << 1 s |  |
| 3 |  |  |  |
| 4 |  |  |  |

**Labirinto 10x10**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Pochi | Medi | Tanti |
| 2 | << 1 s | << 1 s | << 1 s |
| 3 |  |  |  |
| 4 |  |  |  |

**Labirinto 30x30**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Pochi | Medi | Tanti |
| 2 | << 1 s | << 1 s |  |
| 3 | << 1 s | << 1 s |  |
| 4 |  |  | << 1 s |

**Labirinto 50x50**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Pochi | Medi | Tanti |
| 2 | << 1 s | << 1 s |  |
| 3 | << 1 s | << 1 s |  |
| 4 |  |  |  |

**Labirinto 80x80**

(Ricontrollare, mi sembra che non abbiano senso)

|  |  |  |  |
| --- | --- | --- | --- |
|  | Pochi | Medi | Tanti |
| 2 | 0.93 s | < 1 s | 4.43 s |
| 3 | 0.60 s | 1.46 s | 2.98 s |
| 4 | < 1 s | < 1 s | 2.30 s |

**Labirinto 100x100**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Pochi | Medi | Tanti |
| 2 | < 1 s | 5.07 s |  |
| 3 |  |  | 2.70 s |
| 4 | < 1 s |  |  |

**Labirinto 200x200**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Pochi | Medi | Tanti |
| 2 | 19 s |  | 24 s |
| 3 | 3 min 38 s (\*) | 3 s |  |
| 4 |  |  |  |

**(\*) nessun ostacolo**

**Labirinto 300x300**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Pochi | Medi | Tanti |
| 2 | 8 min 40 s |  | 1 min 20 s |
| 3 |  | 14 min |  |
| 4 |  |  |  |

**Uscite non raggiungibili**

|  |  |  |  |
| --- | --- | --- | --- |
| Numero uscite | Dimensioni | Numero Ostacoli | Tempo |
| 1 | 10x10 | 10 | << 1 s |
| 2 | 30x30 | 30 | 0.9 s |

**Vedere perché IDA si rompa su lambirinti 10x10 nei casi 4 e 5**

**Provare ancora con qualche labirinto che porti ad una sequenza di mosse più complessa della L (ad es che costringa a tornare “indietro”)**

**Labirinti con uscite non raggiungibili**

Su un labirinto 10x10 con una sola uscita bloccata da una colonna di muri, l’assenza di soluzione viene rilevata in meno di un secondo.

Su un labirinto 40x40 con due uscite bloccate da due recinti di muri, l’assenza di soluzione viene rilevata in più di un secondo (fare un rilevamento preciso).

**Labirinti non quadrati**

Trova sempre la soluzione in meno di un secondo, ma ha senso che sia così, sono solo esempi giocattolo, che servono a mostrare che l’algoritmo funziona anche in questi casi, poi per le prestazioni dovrebbero valere le stesse riflessioni fatte in precedenza per quelli quadrati.

**Algoritmo IDA\***

**Labirinto 5x5**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Pochi | Medi | Tanti |
| 2 |  | << 1 s |  |
| 3 |  |  |  |
| 4 |  |  |  |

**Labirinto 10x10**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Pochi | Medi | Tanti |
| 2 | << 1 s | 0.54 s | 2.31 m |
| 3 |  |  |  |
| 4 |  |  |  |

**Labirinto 30x30**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Pochi | Medi | Tanti |
| 2 | << 1 s | 4 s |  |
| 3 | << 1 s | << 1 s (\*) |  |
| 4 |  |  | 4.30 s |

**(\*) conferma che gli elevati tempi di esecuzione si presentano con soluzioni che non hanno la classica struttura ad L. Soprattutto, ad essere problematici sono gli ostacoli a scala**

**Labirinto 50x50**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Pochi | Medi | Tanti |
| 2 | << 1 s | Killed (\*) |  |
| 3 | Stack Overflow (\* \*\*) | Killed (\*) |  |
| 4 |  |  |  |

**Labirinto 300x300**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Pochi | Medi | Tanti |
| 2 | < 1 s |  |  |
| 3 |  |  |  |
| 4 |  |  |  |

**Uscite non raggiungibili**

|  |  |  |  |
| --- | --- | --- | --- |
| Numero uscite | Dimensioni | Numero Ostacoli | Tempo |
| 1 | 10x10 | 10 | << 1 s |
| 2 | 30x30 |  |  |
| 4 |  |  |  |

**(\*\*) con stack size impostato ad 1 GB**

Osservazioni generali

1. A rigor di logica, IDA dovrebbe essere più veloce su labirinti con pochi ostacoli in cui ci sono soluzioni ad L, soprattutto se molto grandi, perché espande meno nodi
2. Come osservato, è più lento su labirinti con ostacoli diagonali e soluzioni non ad L, perché la discesa in profondità ed il non ordinare i successori in base al costo lo porta a seguire percorsi che non portano alla soluzione, visitando anche più volte gli stessi sottoalberi (spiegare meglio)