人工智慧報告 by 9817059李 揚

**The statistics of the optimal solution depth. How does this depend on the number of tiles?**

一般來說越多tiles solution depth會越深。

以下我各tile數隨機取測資測五次取平均

1 : 1.4層 4: 6.8層 7: 17.2層

2 : 3.8層 5: 9.8層 8: 22.5層(共測11次其中6次無解)

3 : 5.6層 6: 14.4層

除了深度之外，針對搜尋時間作探討的話，branch factor也會有很大的影響，尤其是在有四個或五個的case之下，branch factor格外的大。同樣是用A\*search 搜尋14層的case，”123405678”只要花0.165秒，但”050243010”卻需要2.431秒。相差超過一個數量級。

**Comparison of IDS and A\*(with different heuristics on the search cost.) Are your results consistent with what are in the text book? How are the results for different number of tiles?**

IDS跟A\*的比較方面跟課本大致一樣，IDS雖然可以省掉很多空間，但跑起來速度遠慢於A\*。至於h1和h2的比較方面，我random產生了幾個測資去讓他們跑跑看，一般情形發現h2比h1快一點，但如果是在深度就深的情形效果就顯著很多。

A\* vs IDS(因IDS有些測資無法跑完!故做測資比較而非隨機產生)

“631482075” 0.014(s) for A\*with h2 0.592(s) for IDS

“305628147” 0.015(s) for A\*with h2 1.903(s) for IDS

“123405678” 0.015(s) for A\* with h2 7.831(s) for IDS

For the 8-tile case, the goal state is unreachable from half of the possible initial states. How do you handle this? Can you check for this before the search starts?

再8-tile的情況之下:是可以在搜尋前判斷能否達到goal state的。主要是用一個公式，看input有幾個descending pair(位置在前面，值卻比較小)

Ex:”012345678” 🡪 0 descending pair

“087654321” --> 28 descending pairs

再把這數量mod 2，如果是偶數就有可reach到的goal state，奇數則無解。(在我程式中，此判斷函示為precheck())

而在少於8-tiles的情況之下則絕對有解。

Performance difference with tree or graph based search.

根據我的測試，tree search跟graph search用的時間是差非常多的，尤其是搜尋深度較深的時候。

同樣是用A\*(with h2)search

“0123426587”這個case，tree search 要跑15.319(s)，但graph search只要跑0.218(s)。(此case的cost為20)

另外跑課本上的case:"724506831"(cost為26)，tree search甚至無法跑出來，但graph search只需要15.288秒。

可見得雖然在這次的程式中，想要implement graph search要多花9!\*3\*3的char的空間去紀錄每個state，不過可以避免掉一直搜尋同個state的時間，而這省下的時間是很可觀的。且用graph search不見得會比較花空間，如果是在較為複雜的case，graph case避免去new一些重複的node，這省下來的空間是很可觀的。如果是用tree search的話，有些測資甚至會把記憶體花完而跑不出來。