電腦對局HW1 Sokoboru

b05902045 宋哲寬

1. Implementation

Environment and Run

我的環境為 Ubuntu 20.04, intel i7-7

```
> make
> ./solver < [testdata file]
```

我主要實作了2種search algorithm分別為BFS跟A*

A*

重點在於實作h(x)上面，而比較好的方法是估計minimum pushing effort因為主角每次最多推一個東西，而估計每個球到箱子以及箱子到球的minimum effort然後再利用完美匹配（匈牙利演算法）來估計距離，又分為admissable跟nonadmissable，這題因為球可以一次滾很長的距離所以如果要保證admissable的話，很容易造成低估讓h(x)的值太小，但是如果使用一些比較大的值就不容易找到最佳解。

我主要使用了了兩種

第1種 是用球跟箱子的曼哈頓距離來算出每個之間所需要的cost，之後再利用完美匹配來估出來值，這個會很接近admissable（只有球能完美滾動時才會高估）。不過這個演算法時間仍然沒有很快，small的測資就差不多一分鐘了，感覺很難過medium或是large

第2種我就加上了尚未解出來的箱子的數量*3，這會讓他變成不是admissable但是速度會快非常的多，因為會heauristic會傾向於走向快找到解的盤面，有點像dfs的感覺

壓縮盤面

盤面需要存進hashtable來判定是否會走過，所以希望可以愈小愈好，我使用了兩個unsigned long long 來表達 ball 跟box以及一個short來存player的位置，如下

```
struct BOARD{
    unsigned long long ball;
    unsigned long long box;
    short player;
};
```

完美匹配

在A*;上面為了估出更好的hearuistic的值，我使用了匈牙利演算法來進行完美匹配，當中因為要實作O(n^3)的方法十分複雜有上網參考別人的實作方式。

死角偵測

這題因為箱子跟球都可以動，所以就算有球跑到死角或是箱子跑到死角，還是可以靠移動球或是箱子解，除非剩下的球跟箱子都全卡死了，所以我認為可以刪去的盤面並沒有很多，實作過後覺得對效能影響不大，所以後來沒有繼續進行完整的實作

2. Experiment

我主要分析了三種不同的搜尋演算法，分別是BFS, A-*admisable跟* A-nonadmissable。除了時間之外，走過的盤面數量我也有紀錄，來分析是否能有效找到解。

當中time, nodes, penalty都是一組測資的全部加總

Algorithm	Test Data	time	nodes	penalty
BFS	tiny.in	1.815s	536342	259
BFS	small.in	Not solved		
A*-admis	tiny.in	1.534s	160059	259
A*-admiss	small.in	1m6.705	4536788	303
A*-admiss	medium.in	Not solved		
A*-noadmiss	tiny.in	0.307s	31742	278
A*-noadmiss	small.in	2.195s	184377	335
A*-nadmiss	medium.in	4.369s	305968	406

可以看出來每個時間複雜度都差的非常的多，主要跟尋找的node的數量成正比，A*可以有效優先選認為有用的盤面

3.Discussion

The complexity of Sokoboru puzzle

一個sokoboru puzzle，如果單純沒有WALL的話幾乎不會有完全地死角，所以Branching factor會很大。例如small和medium的puzzle，通常沒有什麼WALL所以解也不會到很長，但是如果用爆搜的話大概就過不了，反而如tiny的puzzle雖然有一些解很長，但是因為限制很多，反而可以阻止exponential的增長

The complexity of different search algorithms

在這裡使用e為branching factor，d為解的深度

BFS

很好保證最佳解的演算法，但是因為banching factor接近於4, 所以時間複查度會指數成長 O(e^d)

Bi-direction

因為球跟箱子都可以動，所以最終盤面的情況非常的多，會是C(N*M, #boxs), 會是排列成長，所以太多了。所以不考慮使用

A*

重點在於h(x)是估計的值的差異，如果把值調大，很容易可以找到解，不過也會找不太到最佳解，可以看出在實驗中，non admissable的版本雖然速度很快，但是penalty也比別人大上不少

我的puzzle

我的puzzle長的如下

```
5 6
------
-00-$$
--#---
--#-#-
--0@$#
```

這個並不難，大概23步左右就可以解出來，因為我不是很會玩推箱子，所以我也沒有創出很難的puzzle不過我寫出了一兩種跟計算hearuistic上容易造成誤差的pattern，不過因為解的範圍很有限，所以搜尋起來也很快就可以找到解