Theory of Computer Games – HW1

B04705003 資工三 林子雋

Implementation

- How to compile and run:

  $ cd b04705003/source

  $ make

  $ ./main bibfs < [small.in|small2.in|large.in|large2.in]

- Algorithm and heuristic implementation:

  I chose Bidirectional BFS as my final algorithm. Also, I combined some simple methods to prune branch. For example, if a box is pushed to a square which causing it unmovable, I will not search this branch.

  Reverse BFS means that I search from goal states.

Experiment

- Setting:

Environment: CSIE Workstation

small.in

Algorithms	Execution time
Breadth First Search(BFS)	4.9 seconds
BFS with pruning	1.2 seconds
Reverse BFS	0.05 seconds
Bidirectional BFS	2.3 seconds

large.in

Algorithms	Execution time
Breadth First Search(BFS)	Too Long
BFS with pruning	22 mins
Reverse BFS	29 seconds
Bidirectional BFS	5 seconds

large2.in

Algorithms	Execution time
Breadth First Search(BFS)	Too long
BFS with pruning	Too long
Reverse BFS	Too long
Bidirectional BFS	2 mins

Discussion

- The complexity of this game:
  In this homework, our board are restricted into maximum 50 squares. Therefore, its maximum state space is factorial of 50. If we take wall squares into consideration and do some domain knowledge pruning, the search space may dramatically decrease.
- The complexity of each algorithm:
  For BFS, its complexity is O(b^d). After using bidirectional BFS, its complexity decrease to O(b^(d/2).
  In the experiments part, you can find that bidirectional BFS outperforms other algorithms in most of the cases. Only in small.in does the bidirectional BFS perform worse than others. I think it may due to the overhead on hash table and memory.