Back 2 Back Solver

Jordan Turley

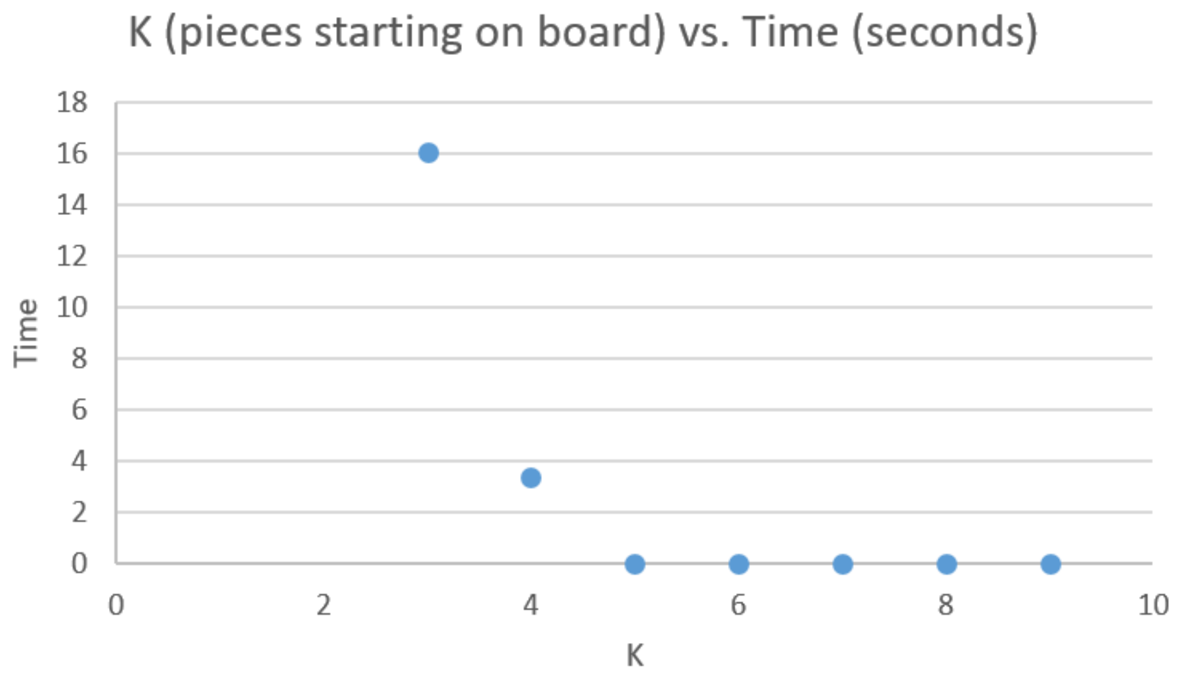
For Assignment Two of the Back 2 Back project, I implemented the A* search algorithm to solve Back 2 Back and tell the user where to place each piece on the board, given one of the start states from the Back 2 Back instructions.

For this problem, I defined the cost of placing a piece as how many spaces it fills. So, for example, if one cylinder of a piece goes all the way through to the other side, it gets a score of two; if it only covers one side of the board, it gets a score of one. I sum all of these up to get the cost of the piece. The path cost of a node is the cost of performing the action of that node plus the cost of all the actions before that node.

My heuristic is that for every hole on the board, a hole that is filled on both sides is worth 2 points, filled on one side is 1.5, and empty holes are 0. The heuristic score of an empty board is zero, and of a fully solved board is 60. This heuristic is not admissible, because it can overestimate the cost to the goal, but it does not really matter for this problem.

To use the code, you create a new Board object. This will create an empty board. Then, to set one of the initial states from the book, call setInitialState(int) on the board, passing it the number of the challenge in the book. This number must be between 1 and 60. However, because it was very tedious to code in each start layout, I actually did not finish this part. Ideally, you would be able to set the game to any start state in the book, but in this program, you can set start states 1-8, 11, 25, 26, 35, and 44-60. This should not really be a problem, however, because I do have at least one layout for every number of pieces on the board. Then, you call solve() on the board, and a list of Action objects will be returned, which you can just print out to show the steps to the solved board. When you print an Action object, it will tell you the piece, orientation, x-y location, and side. Piece and orientation are clear, the x-y location is where the top-left of the piece goes, even if the top left of the piece is empty. Also, (1,1) is the top-left cell on the board. The front of the board is the side that is facing forward in the challenge book.

I used Java for this assignment. I didn't use any third party packages; I used Java's Priority Queue, Lists, and HashMap. I did not implement an explored set, but the program seems to work fine without it. I worked with Will Edwards but did the majority of the coding. We used my code from Assignment 1, because Will was unable to finish his, but I also wrote most of the code for this assignment. There are no bugs that I know of; everything seems to work fine.



For my experiment, I ran the program 3 times for each starting layout (that I had programmed in), and averaged them, then averaged all the results for each K value. As we can see, as K decreases, Time increases exponentially. One thing that I did notice is how the time can change within each K value. For example, for problem 48 (K = 3), it took about 10 seconds to solve. However, for problem 47 (also K = 3), it took almost 30 seconds. I definitely could have done my experiment better; I should have had all the layouts available and tried all of them, but put it off until too late. I also didn't test any problems that took very long (more than a few minutes), because I ran out of time. However, even without all starting states, we can still very clearly see how the time increases drastically as K decreases.

From this assignment, I learned how to implement the A* search algorithm. We learned how to in class, and saw pseudo-code, but to actually implement it for a real world problem gave me a much better understanding of it and how it works.

Example output for solving challenge 1 from Back 2 Back book (from running Solver.java):

```
Current side: front
|P|P|R|T|U|U|
|P|E|R|U|U|B|
|E|G|G|T|Y|B|
| | |G| |Y|O|
| | |G|O|O|O|

Current side: back
|U|U|T|R|P|P|
|B|B|T|R|E|E|
|B|Y|T|R|G|E|
|Y|Y| |g| | |
|O|Y|O|G| | |

Solving took: 0.014 seconds

Solution:
Place Color: Pink
2 0
2 1
at (1, 4) on the front

Place Color: Light Blue
2 1 2
0 0 1
at (3, 4) on the back
```