ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

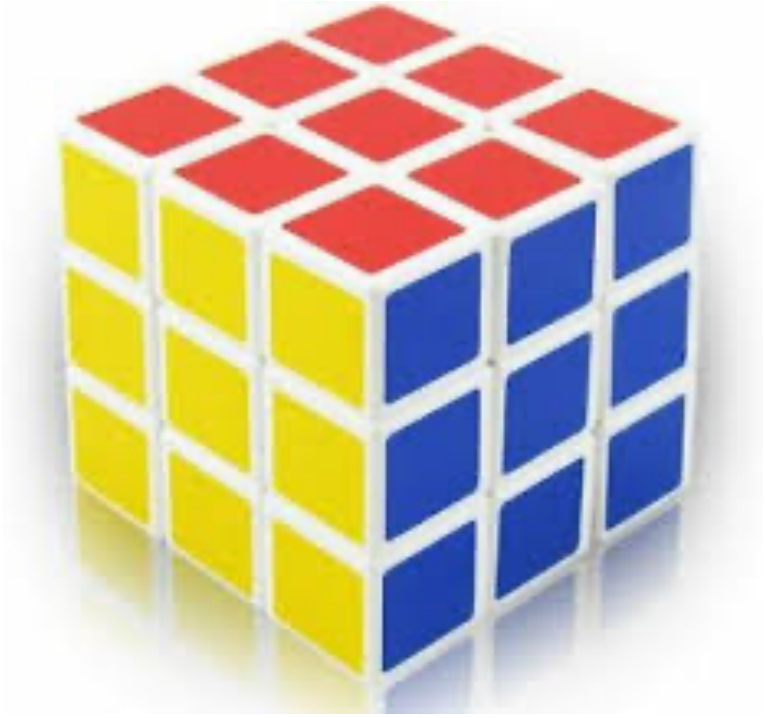
ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ & ΤΕΧΝΟΛΟΓΙΑΣ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
SCHOOL OF INFORMATION SCIENCES & TECHNOLOGY

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
DEPARTMENT OF INFORMATICS

Τεχνητή Νοημοσύνη

Διδάσκων: Ι. Ανδρουτσόπουλος

1η Εργασία



Φοιτητές:
Αθανάσιος Τριφώνης, p3200298
Χριστόφορος Παπαποστόλου, p3150208

Ακαδημαϊκό έτος: 2022–23

Οδηγίες Εκτέλεσης:

Στην κλάση Main ορίζεται μία αρχική κατάσταση-κύβος μέσω του κατασκευαστή που έχει όρισμα το K (ελάχιστος αριθμός πλευρών ίδιου χρώματος για να θεωρηθεί τελική κατάσταση)

```
Cube initialState =  new  Cube(6); // Least number of faces with the same color to solve the cube. [1,6]
```

Δημιουργείται ο κύβος και εν συνεχεία μία γεννήτρια τυχαίων αριθμών παίρνει ως όρισμα τα όρια του πλήθους των τυχαίων κινήσεων που θα γίνουν για να «μπερδευτεί» ο κύβος. Το κάτω όριο πρέπει να είναι μη αρνητικός αριθμός και το πάνω όριο να είναι μεγαλύτερο του κάτω ορίου, αφού αυτά τα δύο δηλώνουν ένα σύνολο [α ,β).

```
initialState.randomize(5,10); // randomize the cube with α to β-1 moves.
```

Η μέθοδος AstarClosedSet δέχεται ως πρώτο όρισμα την αρχική κατάσταση για να ξεκινήσει αναζήτηση, καθώς και την ευρετική που θα χρησιμοποιηθεί. Έχουμε υλοποιήσει δύο διαφορετικές για σύγκριση, αριθμημένες με 0 και 1.

```
Cube terminalState = astar.AstarClosedSet(initialState, 1); // 0 for corner-edges heuristic, 1 for total distance heuristic.
```

Οι εκτυπώσεις του κύβου στο τερματικό γίνονται με την παρακάτω μορφή:

```
--------------------
         Top
        5 5 5
        5 5 5
Left    5 5 5   Right   Back
4 4 4   1 1 1   2 2 2   3 3 3
4 4 4   1 1 1   2 2 2   3 3 3
4 4 4   1 1 1   2 2 2   3 3 3
        6 6 6
        6 6 6
        6 6 6
        Bottom
--------------------
```

Εισαγωγή:

Στην παρούσα εργασία υλοποιήσαμε τον κύβο του Rubik στην γλώσσα Java. Ο κύβος χρησιμοποιεί την διεπαφή RubikCube, η οποία ορίζει τις 12 δυνατές κινήσεις, τον έλεγχο του κύβου για το αν είναι λυμένος καθώς και την εκτύπωση του στο τερματικό. Οι κινήσεις είναι 2 για κάθε πλευρά από τις 6 του κύβου, μία δεξιόστροφη και μία αριστερόστροφη.

```
// Rotate the front face of the cube anticlockwise.
public void F_l();

// Rotate the front face of the cube clockwise.
public void F_r();

// Rotate the back face of the cube anticlockwise.
public void B_l();

// Rotate the back face of the cube clockwise.
public void B_r();

// Rotate the top face of the cube anticlockwise.
public void U_l();

// Rotate the top face of the cube clockwise.
public void U_r();

// Rotate the bottom face of the cube anticlockwise.
public void D_l();

// Rotate the bottom face of the cube clockwise.
public void D_r();

// Rotate the left face of the cube anticlockwise.
public void L_l();

// Rotate the left face of the cube clockwise.
public void L_r();

// Rotate the right face of the cube anticlockwise.
public void R_l();

// Rotate the right face of the cube clockwise.
public void R_r();

// Print the cube to the console.
public void printCube();

// Check if the cube is solved.
public  boolean isSolved();

// Check if K faces are solved.
public boolean isKSolved();
```

Ο αλγόριθμος επίλυσης που επιλέχθηκε είναι ο A* με κλειστό σύνολο καθώς από κάποιους κόμβους προκύπτουν άπειρα κλαδιά και θα ήταν χρήσιμο να κρατάμε στη μνήμη τις καταστάσεις που έχουμε εξερευνήσει ώστε να μην παράγουμε εκ νέου απογόνους για αυτές.

Υλοποίηση Κλάσης Κύβου

Ο κύβος έχει υλοποιηθεί μέσω ενός 3D πίνακα ακεραίων, διαστάσεων 6x3x3 όπου ο πρώτος αριθμός αναπαριστά την πλευρά, και οι επόμενοι δύο τους άξονες x και y. Εντός του πίνακα βρίσκεται ένας αριθμός ανά πλακίδιο στο διάστημα [1,6], για την αναπαράσταση του αριθμού της πλευράς που ανήκει στην τελική κατάσταση (αντί χαρακτήρα που θα έδειχνε χρώμα).

- Τα πεδία που περιλαμβάνει ένας κύβος είναι το score που του αποδίδει η ευρετική:

```
// Heuristic score.
private double score;
```

- Το πεδίο K συμβολίζει τον ελάχιστο αριθμό των πλευρών που θέλουμε να είναι ίδιου χρώματος για να θεωρηθεί σε τελική κατάσταση:

```
// Number of correct faces needed to solve the cube.
private int K;
```

- Καθώς και τον γονέα από όπου παράχθηκε:

```
// The parent state.
private Cube parent = null;
```

- Επίσης εντός της ίδιας κλάσης υπάρχουν και δύο στατικές δομές, το faces, που χρησιμοποιήθηκε κυρίως σε κάποια στάδια του debugging:

```
// Map with the faces of the cube from 1 to 6.
private static final Map<Integer, String> faces = new HashMap<>();
```

- και τέλος ο 3D πίνακας solved ο οποίος περιλαμβάνει τις τιμές ενός κύβου τελικής κατάστασης.

Ευρετικές:

Επιλέξαμε να χρησιμοποιήσουμε δύο διαφορετικές ευρετικές και να τις συγκρίνουμε μεταξύ τους για να δούμε ποια προσεγγίζει καλύτερα το πραγματικό κόστος.

Η απόσταση που θεωρούμε στην παρούσα εργασία ισούται με 2 μονάδες για πλακίδιο που βρίσκεται στην ανάποδη πλευρά από την τελική του καθώς θέλει τουλάχιστον δύο κινήσεις για να κάνει την μετάβαση αυτή, αντίθετα ένα πλακίδιο που βρίσκεται σε «διπλανή» πλευρά, πλευρά δηλαδή η οποία εφάπτεται με την πλευρά της θέσης που θα είχε στην τελική κατάσταση, λογίζεται ως μία μονάδα απόστασης καθώς μεταβαίνει σε αυτή με τουλάχιστον 1 κίνηση *(πχ ένα πλακίδιο που βρίσκεται στην αριστερή πλευρά και η τελική κατάσταση το έχει στην δεξιά πλευρά, λογίζεται ως 2 μονάδες απόσταση. Ένα πλακίδιο που είναι στην πάνω πλευρά και πρέπει να πάει στη μπροστινή θεωρούμε πως έχει απόσταση 1. Εντός της κλάσης Astar, κατά την αναζήτηση υπάρχει ένας δείκτης ο οποίος αυξάνεται κάθε φορά που γίνεται εξερεύνηση ενός κόμβου).*

- Η ευρετική countTotalDistance() υπολογίζει το άθροισμα των αποστάσεων για κάθε πλακίδιο από την θέση που έχει μέχρι την θέση που βρίσκεται σε τελική κατάσταση.

```
public void countTotalDistance()
```

- Η ευρετική countTotalDistanceCornersEdges() υπολογίζει ξεχωριστά τις αποστάσεις των γωνιών και τις αποστάσεις των ακμών, η λογική που επιλέχθηκε βασίζεται στο ότι επειδή ο αριθμός των κομματιών που αντιστοιχούν σε γωνίες είναι μικρότερος του αριθμού ακμών σε έναν κύβο (8 γωνίες, 12 ακμές).
-

```
public void countTotalDistanceCornersEdges()
```

Παρακάτω παραθέτονται κάποια αποτελέσματα δοκιμών με διαφορετικούς συνδυασμούς ευρετικής, τιμής K, καθώς και πλήθους τυχαίων κινήσεων για το μπέρδεμα του κύβου.

- Η ευρετική countTotalDistance για K = 6 έβρισκε εντατικά λύσεις μέχρι τις 6 κινήσεις «μπερδέματος», έπειτα σταμάτησε να βρίσκει λύσεις και άρχισε να κολλάει σε διαρκές ψάξιμο. Κάποιες φορές βρήκε λύση καθώς διαιρέσαμε την αθροιστική απόσταση με κάποιους αριθμούς.

- Οι αριθμοί ενδέχεται να παρουσίασαν αυτό το φαινόμενο τυχαία, καθώς δεν το εμφάνιζαν κατά εξακολούθηση με εκ νέου εκτέλεση ή με άλλες περιπτώσεις, γεγονός που ίσως δείχνει πως σε κάποιες περιπτώσεις που είχαμε υπερεκτιμήσει το κόστος η εν λόγω διαίρεση βοηθούσε στο να το υποεκτιμήσουμε αρκετά ώστε να είμαστε κοντά στο πραγματικό χωρίς να το υπερβαίνουμε.

- Μετά τις 7 κινήσεις σταμάτησε τελείως να βρίσκει λύση για K = 6 και έτσι μειώσαμε το K, αλλά το αποτέλεσμα παρέμεινε το ίδιο, έβρισκε λύση αρκετά σπάνια.

#	K	Scramble Moves	Time Searching	Moves to Solve
1	6	1	0.001sec	1
2	6	2	0.001sec	2
3	6	3	0.001sec	1
4	6	4	0.001sec	2
5	6	5	0.001sec	3
6	6	6	0.002sec	4
7	6	7	0.04sec	5
8	6	8	0.04sec	4
9	4	7	0.054sec	3
10	1	7	0.75sec	4

Ακολουθούν κάποιες ενδεικτικές εκτελέσεις του countTotalDistance. Είναι οι εκτελέσεις 6, 7, 8 και 9 αντίστοιχα από αριστερά προς τα δεξιά. Κάθε εκτέλεση εκτείνεται κάθετα, ξεκινάει με εκτύπωση του γονέα και συνεχίζει εκτυπώνοντας κάθε παιδί της διαδρομής προς την λύση.

Path:

Top

3 2 6

3 5 6

Left		Right	Back
5 5 5	2 1 3	6 2 2	1 3 4
4 4 4	5 1 3	6 2 2	1 3 6
4 4 4	5 1 3	6 2 2	1 3 6

1 6 2

1 6 5

1 4 5

Bottom

Top

3 2 3

3 5 3

Left		Right	Back
5 5 5	2 1 2	6 6 6	4 3 4
4 4 4	5 1 5	2 2 2	6 3 6
4 4 4	5 1 5	2 2 2	6 3 6

1 6 1

1 6 1

1 4 1

Bottom

Top

3 2 2

3 5 5

Left		Right	Back
5 5 5	2 1 1	2 2 6	3 3 4
4 4 4	5 1 1	2 2 6	3 3 6
4 4 4	5 1 1	2 2 6	3 3 6

1 6 6

1 6 6

1 4 4

Bottom

Top

2 2 2

5 5 5

Left		Right	Back
5 4 4	1 1 1	2 2 6	3 3 3
5 4 4	1 1 1	2 2 6	3 3 3
5 4 4	1 1 1	2 2 6	3 3 3

6 6 6

6 6 6

4 4 4

Bottom

Top

5 5 5

5 5 5

Left		Right	Back
4 4 4	1 1 1	2 2 2	3 3 3
4 4 4	1 1 1	2 2 2	3 3 3
4 4 4	1 1 1	2 2 2	3 3 3

6 6 6

6 6 6

6 6 6

Bottom

Search time:0.002 sec.

Top

1 5 5

1 5 5

Left		Right	Back
5 5 3	4 1 1	6 2 2	3 3 2
4 4 4	1 1 1	6 2 2	3 3 3
1 2 2	3 3 3	4 4 4	1 5 5

6 6 6

6 6 6

4 4 6

Bottom

Top

1 5 5

1 5 5

Left		Right	Back
5 5 3	4 1 1	6 2 2	3 3 2
4 4 4	1 1 1	6 2 2	3 3 3
1 5 5	1 2 2	3 3 3	4 4 4

4 6 6

4 6 6

6 6 6

Bottom

Top

1 5 5

1 5 5

Left		Right	Back
5 5 3	4 1 1	6 2 2	3 3 2
4 4 4	1 1 1	6 2 2	3 3 3
4 4 4	1 5 5	1 2 2	3 3 3

6 4 4

6 6 6

6 6 6

Bottom

Top

3 5 5

3 5 5

Left		Right	Back
4 4 5	1 1 1	6 2 2	3 3 6
4 4 5	1 1 1	6 2 2	3 3 6
4 4 3	5 5 5	1 2 2	3 3 6

4 4 4

1 6 6

1 6 6

Bottom

Top

3 5 5

3 5 5

Left		Right	Back
4 4 4	5 1 1	2 2 2	3 3 6
4 4 4	5 1 1	2 2 2	3 3 6
4 4 4	5 1 1	2 2 2	3 3 6

1 6 6

1 6 6

1 6 6

Bottom

Top

5 5 5

5 5 5

Left		Right	Back
4 4 4	1 1 1	2 2 2	3 3 3
4 4 4	1 1 1	2 2 2	3 3 3
4 4 4	1 1 1	2 2 2	3 3 3

6 6 6

6 6 6

6 6 6

Bottom

Top

2 5 6

3 5 3

Left		Right	Back
3 4 4	5 1 5	2 2 1	4 3 6
3 4 4	5 1 5	2 2 1	4 3 6
3 5 5	2 4 2	6 6 1	4 2 6

1 6 1

1 6 1

4 6 5

Bottom

Top

5 5 6

5 5 3

Left		Right	Back
4 4 5	1 1 5	2 2 1	4 3 3
4 4 5	1 1 5	2 2 1	4 3 3
3 3 3	4 4 2	6 6 1	4 2 2

6 6 1

6 6 1

6 6 5

Bottom

Top

5 5 5

5 5 5

Left		Right	Back
4 4 5	1 1 1	6 2 2	3 3 3
4 4 5	1 1 1	6 2 2	3 3 3
3 3 3	4 4 5	1 1 1	6 2 2

6 6 4

6 6 4

6 6 4

Bottom

Top

5 5 5

5 5 5

Left		Right	Back
4 4 5	1 1 1	6 2 2	3 3 3
4 4 5	1 1 1	6 2 2	3 3 3
4 4 5	1 1 1	6 2 2	3 3 3

4 4 4

6 6 6

6 6 6

Bottom

Top

5 5 5

5 5 5

Left		Right	Back
4 4 4	1 1 1	2 2 2	3 3 3
4 4 4	1 1 1	2 2 2	3 3 3
4 4 4	1 1 1	2 2 2	3 3 3

6 6 6

6 6 6

6 6 6

Bottom

Top

5 5 5

5 5 5

Left		Right	Back
4 4 6	1 1 1	2 2 2	3 3 3
4 4 3	6 1 1	2 2 2	3 3 3
4 4 3	6 4 4	5 1 1	2 2 3

2 1 1

6 6 6

6 6 6

Bottom

Top

3 5 5

3 5 5

Left		Right	Back
4 4 4	5 1 1	2 2 2	3 3 6
4 4 4	5 1 1	2 2 2	3 3 6
3 3 6	4 4 4	5 1 1	2 2 2

1 1 1

6 6 6

6 6 6

Bottom

Top

3 5 5

3 5 5

Left		Right	Back
4 4 4	5 1 1	2 2 2	3 3 6
4 4 4	5 1 1	2 2 2	3 3 6
4 4 4	5 1 1	2 2 2	3 3 6

1 6 6

1 6 6

1 6 6

Bottom

Top

5 5 5

5 5 5

Left		Right	Back
4 4 4	1 1 1	2 2 2	3 3 3
4 4 4	1 1 1	2 2 2	3 3 3
4 4 4	1 1 1	2 2 2	3 3 3

6 6 6

6 6 6

6 6 6

Bottom

Search time:0.054 sec.

- Η ευρετική countTotalDistanceCornersEdges υπολογίζει 2 διαφορετικές τιμές κόστους, μία για τις πλευρές και μία για τις γωνίες, ξεχωριστά, και στη συνέχεια διαιρεί την κάθε τιμή με το πλήθος των πλακιδίων-πλευρών και γωνιών αντίστοιχα.

#	K	Scramble Moves	Time Searching	Moves to Solve
1	6	4	0.002sec	2
2	6	5	0.001sec	3
3	6	6	0.003sec	4
4	6	7	0.001sec	3
5	6	8	0.02sec	6
6	4	7	0.001sec	3
7	3	7	0.126sec	3
8	2	7	0.001sec	2
9	1	7	0.002sec	4
10	1	6	0.008sec	3

Παρακάτω δίνονται ενδεικτικά κάποια αποτελέσματα εκτελέσεων(συγκεκριμένα των 6,7,9 και 10 αντίστοιχα από αριστερά προς δεξιά).

```
        Top
        5 5 5
        5 5 5
Left    4 4 4   Right   Back
2 2 5   3 3 3   6 4 4   1 1 1
6 4 4   1 1 1   2 2 5   3 3 3
6 4 4   1 1 1   2 2 5   3 3 3
        6 6 6
        6 6 6
        2 2 2
        Bottom
--------------------


--------------------
        Top
        4 5 5
        4 5 5
Left    4 5 5   Right   Back
3 3 3   6 4 4   1 1 1   2 2 5
6 4 4   1 1 1   2 2 5   3 3 3
6 4 4   1 1 1   2 2 5   3 3 3
        6 6 6
        6 6 6
        2 2 2
        Bottom
--------------------


--------------------
        Top
        4 4 4
        5 5 5
Left    5 5 5   Right   Back
6 4 4   1 1 1   2 2 5   3 3 3
6 4 4   1 1 1   2 2 5   3 3 3
6 4 4   1 1 1   2 2 5   3 3 3
        6 6 6
        6 6 6
        2 2 2
        Bottom
--------------------


--------------------
        Top
        5 5 5
        5 5 5
Left    5 5 5   Right   Back
4 4 4   1 1 1   2 2 2   3 3 3
4 4 4   1 1 1   2 2 2   3 3 3
4 4 4   1 1 1   2 2 2   3 3 3
        6 6 6
        6 6 6
        6 6 6
        Bottom
--------------------


Search time:0.001 sec.
```

```
        Top
        1 5 5
        1 5 5
Left    4 5 5   Right   Back
4 4 3   6 1 1   2 2 2   3 3 5
4 4 3   6 1 1   2 2 2   3 3 5
6 4 4   1 1 1   2 2 5   4 4 3
        6 6 6
        6 6 6
        2 3 3
        Bottom
--------------------


--------------------
        Top
        1 5 5
        1 5 5
Left    4 5 5   Right   Back
4 4 3   6 1 1   2 2 2   3 3 5
4 4 3   6 1 1   2 2 2   3 3 5
4 4 3   6 4 4   1 1 1   2 2 5
        2 6 6
        3 6 6
        3 6 6
        Bottom
--------------------


--------------------
        Top
        5 5 5
        5 5 5
Left    5 5 5   Right   Back
4 4 4   1 1 1   2 2 2   3 3 3
4 4 4   1 1 1   2 2 2   3 3 3
3 3 3   4 4 4   1 1 1   2 2 2
        6 6 6
        6 6 6
        6 6 6
        Bottom
--------------------


--------------------
        Top
        5 5 5
        5 5 5
Left    5 5 5   Right   Back
4 4 4   1 1 1   2 2 2   3 3 3
4 4 4   1 1 1   2 2 2   3 3 3
4 4 4   1 1 1   2 2 2   3 3 3
        6 6 6
        6 6 6
        6 6 6
        Bottom
--------------------


Search time:0.126 sec.
```

```
        5 2 2
        5 5 6
Left    6 6 6   Right   Back
1 4 2   1 1 1   4 2 5   1 5 4
5 4 2   1 1 1   4 2 5   1 3 3
5 4 3   4 3 3   6 6 3   2 3 3
        6 2 2
        6 6 3
        4 4 5
        Bottom
--------------------


--------------------
        Top
        5 2 2
        5 5 6
Left    4 4 6   Right   Back
1 4 6   1 1 3   2 2 5   1 5 4
5 4 6   1 1 3   2 2 5   1 3 3
5 4 6   1 1 4   6 6 3   2 3 3
        2 2 3
        6 6 3
        4 4 5
        Bottom
--------------------


--------------------
        Top
        5 2 2
        5 5 1
Left    4 4 1   Right   Back
1 4 6   1 1 2   5 5 3   5 5 4
5 4 6   1 1 6   2 2 6   3 3 3
5 4 6   1 1 6   2 2 6   3 3 3
        2 2 3
        6 6 3
        4 4 4
        Bottom
--------------------


--------------------
        Top
        5 5 1
        5 5 1
Left    4 4 1   Right   Back
4 4 6   1 1 2   5 5 5   4 3 3
4 4 6   1 1 6   2 2 2   5 3 3
4 4 6   1 1 6   2 2 2   5 3 3
        2 2 3
        6 6 3
        6 6 3
        Bottom
--------------------


--------------------
        Top
        5 5 5
        5 5 5
Left    4 4 4   Right   Back
4 4 6   1 1 1   5 2 2   3 3 3
4 4 6   1 1 1   5 2 2   3 3 3
4 4 6   1 1 1   5 2 2   3 3 3
        2 2 2
        6 6 6
        6 6 6
        Bottom
--------------------


Search time:0.002 sec.
```

```
        Top
        5 5 4
        5 5 4
Left    4 4 5   Right   Back
1 1 6   1 1 2   3 3 3   6 4 4
6 4 6   1 1 2   3 2 5   3 3 3
1 4 1   5 5 5   4 2 3   6 6 6
        2 2 3
        6 6 1
        2 2 2
        Bottom
--------------------


--------------------
        Top
        5 5 4
        5 5 4
Left    3 3 4   Right   Back
1 1 5   2 2 5   3 3 3   6 4 4
6 4 4   1 1 5   2 2 5   3 3 3
1 4 4   1 1 5   2 2 3   6 6 6
        6 6 1
        6 6 1
        2 2 2
        Bottom
--------------------


--------------------
        Top
        4 4 4
        5 5 3
Left    5 5 3   Right   Back
6 4 4   1 1 5   2 2 5   3 3 3
6 4 4   1 1 5   2 2 5   3 3 3
1 4 4   1 1 5   2 2 3   6 6 6
        6 6 1
        6 6 1
        2 2 2
        Bottom
--------------------


--------------------
        Top
        5 5 3
        5 5 3
Left    5 5 3   Right   Back
4 4 4   1 1 5   2 2 2   6 3 3
4 4 4   1 1 5   2 2 2   6 3 3
4 4 4   1 1 5   2 2 2   6 3 3
        6 6 1
        6 6 1
        6 6 1
        Bottom
--------------------


Search time:0.08 sec.
```

Οι ευρετικές που χρησιμοποιήθηκαν δεν υπολόγιζαν ακριβώς το κόστος με αποτέλεσμα να μην βρίσκουμε πάντα λύση όταν αυξάνονταν οι κινήσεις που χρειάζονταν για την επίλυση. Ο κύριος λόγος που υπήρχε αυτό το ζήτημα πιστεύουμε πως ήταν εξαιτίας της υλοποίησης του κύβου που κάναμε. Το πρόβλημα της είναι πως απεικονίζει έναν κύβο με τον τρόπο που θα το σκεφτόταν ένας άνθρωπος και όχι όπως ένας υπολογιστής. Πιο συγκεκριμένα μία υλοποίηση που θα απεικόνιζε τον κύβο ως μία οντότητα που αποτελείται από κομμάτια δύο κατηγοριών, γωνίες και ακμές θα ήταν ενδεχομένως αρκετά ανώτερη. Σε μία τέτοια υλοποίηση θα ξέραμε πως κάθε γωνία έχει 3 χρώματα και 3 δυνατά orientation, τα οποία μεταβάλονται ανάλογα με την φορά της περιστροφής. Στο orientation 0 η γωνία θα ήταν στο «σωστό» orientation. Αντίστοιχα για τις ακμές θα υπήρχαν 2 χρώματα και 2 orientation (ένα για κάθε χρώμα).

Με την προαναφερόμενη υλοποίηση θα γνωρίζαμε ακριβώς το κόστος για να πάει ένα κομμάτι στη σωστή του θέση και με το σωστό orientation. Προσπαθήσαμε να κάνουμε την εν λόγω υλοποίηση αλλά δυστυχώς δεν καταφέραμε να υλοποιήσουμε την αλλαγή του orientation στις ακμές.