1η Εργασία Τεχνητής Νοημοσύνης

Παιχνίδι Δοχείων (Beaker Game)

Κωνσταντίνος Μπαζιόπουλος

3635

# Γενική Περιγραφή

Το παραδοτέο πρόγραμμα σχεδιάστηκε για να «παίζει» το παιχνίδι των δοχείων με τα χρωματιστά υγρά. Από την πρώτη του περιγραφή το σχέδιο για την υλοποίησή του ήταν το εξής:

1. Υλοποίηση της κλάσης Beaker που αναπαριστά τα δοχεία με τα χρωματιστά υγρά
2. Υλοποίηση της κλάσης State η οποία αναπαριστά ένα σύνολο από αυτά τα δοχεία
3. Υλοποίηση του αλγορίθμου BFS ο οποίος θα επιδρά πάνω σε αντικείμενα της κλάσης State

## Δοχεία (Beakers)

Σε πρώτη φάση έπρεπε να σχεδιαστούν τα δοχεία. Για λόγους απλότητας κάθε υγρό αναπαρίσταται από έναν ακέραιο (διάφορο του μηδενός, καθώς το μηδέν δουλεύει πολύ βολικά για την αναπαράσταση κενών θέσεων).

Ο τρόπος με τον οποίο εισάγονται και εξάγονται τα περιεχόμενα κάθε δοχείου γίνεται με την μορφή LIFO, οπότε ήταν ξεκάθαρο πως έπρεπε να υλοποιούνται με στοίβα. Οι συναρτήσεις εισαγωγής (**add()**) και εξαγωγής (**remove()**) ήταν εύκολα υλοποιήσιμες και η μόνη πραγματική διαφορά τους από τις «έτοιμες» συναρτήσεις push() και pop() είναι πως επιτρέπουν μόνο 4 ακεραίους στη δομή, και επιστρέφουν true ή false ανάλογα με την επιτυχία της διαδικασίας.

Οι υπόλοιπες λειτουργείες των δοχείων σε αυτή τη φάση ήταν κυρίως βοηθητικές (**print()**, **empty()**,**full()**) και δεν νομίζω πως χρειάζονται επεξήγηση.

Η πιο σημαντική όμως λειτουργεία των δοχείων ήταν να επιτρέπεται η μεταφορά υγρών από ένα δοχείο σε ένα άλλο με βάση τις *ιδιαίτερες προδιαγραφές* του προβλήματος. Η μεταφορά αυτή θα συμπεριληφθεί στην κλάση State, αλλά για να λειτουργήσει χρειάστηκαν οι συναρτήσεις **topLiquid()** και **topAmmount()** που μας λένε ποιο υγρό βρίσκεται στην κορυφή του δοχείου και πόσο από αυτό βρίσκεται σε συνεχόμενες θέσεις, αντίστοιχα (ή επιστρέφουν -1 σε περίπτωση άδειου δοχείου).

Ακόμα υλοποιούμε την συνάρτηση **isGood()** η οποία επιστρέφει true αν το δοχείο είναι γεμάτο με το ίδιο υγρό, και false διαφορετικά.

Τέλος, ο κονστράκτορας που χρησιμοποιείται είναι κενός, καθώς τα δεδομένα κάθε δοχείου στην πράξη, ορίζονται από άλλα κομμάτια του προγράμματος

Με αυτά τα εργαλεία μπορούμε να προχωρήσουμε στην επόμενη φάση σχεδίασης.

## Καταστάσεις (States)

Σύμφωνα με τις προδιαγραφές του προβλήματος, κάθε κατάσταση αποτελείται από **n** δοχεία , και αφού το nmax είναι 20, στην πραγματικότητα κάθε κατάσταση αποτελείται από έναν πίνακα 20 δοχείων, και έναν ακέραιο n που καθοδηγεί το πρόγραμμα για το πόσα από αυτά αξιοποιούνται στην πραγματικότητα.

Ο κονστράκτορας δέχεται μόνο την μεταβλητή n και τα περιεχόμενα των δοχείων της κατάστασης ορίζονται από την συνάρτηση **fillRandomly()** που βάζει 4 ml από n-2 υγρά σε τυχαία δοχεία της κατάστασης.

Η συνάρτηση **move(int from,int to)** επιτρέπει την μετακίνηση υγρών από το δοχείο με index from σε αυτό με index to, εκτελώντας τους απαραίτητους ελέγχους. Επιστρέφει true ή false ανάλογα με το αν είναι επιτρεπτή η μετακίνηση που θέλουμε να εκτελέσουμε.

Ακόμα η συνάρτηση **isFinal()** επιστρέφει true αν όλα τα δοχεία είναι, είτε γεμάτα με το ίδιο υγρό (isGood()), είτε άδεια, δηλαδή όταν το πρόβλημα έχει λυθεί, διαφορετικά επιστρέφει false.

Τέλος η **print()** απλά εμφανίζει τα δοχεία της κατάστασης στο terminal

Σε αυτό το σημείο έχουμε όλα τα απαραίτητα συστατικά για να υλοποιήσουμε το παιχνίδι με ένα απλό loop στη main όπου ο χρήστης καλείται να διαλέγει τις ανταλλαγές υγρών.

## BFS

Μέχρι τώρα έχουν εσκεμμένα παραληφθεί τα κομμάτια του προγράμματος που αφορούν την αυτόματη επίλυση του προβλήματος, καθώς θεώρησα πως θα κατανοηθούν πιο εύκολα σε αυτή την παράγραφο με μία top-down σχεδιαστική λογική, όπου βλέπουμε τι χρειάζεται ο BFS για να λειτουργήσει και το προσθέτουμε σταδιακά στα παρακάτω επίπεδα.

Ως προσχέδιο χρησιμοποιούμε το σετ διαφανειών «Υλοποίηση Αλγορίθμων Αναζήτησης» του μαθήματος.

### Αναπαράσταση Κατάστασης

Ότι αφορά σε αυτό το μέρος του προβλήματος έχει ήδη αναλυθεί παραπάνω.

### Μονοπάτι από τη Ρίζα

Για να γνωρίζουμε τα βήματα που εκτελέσθηκαν προσθέτουμε στην κλάση State δύο πεδία, ένα **pointer σε State** που ονομάζεται **parent** και ένα **string** που ονομάζεται **instruction.** To πρώτο δείχνει από ποιο State παράγεται το συγκεκριμένο, ενώ το δεύτερο εξηγεί με ποιόν τρόπο παράγεται (με την μορφή της πρότασης «από χ προς ψ», δηλαδή αναπαριστά τα ορίσματα της move()). Η αρχική κατάσταση παίρνει τις ειδικές τιμές **nullptr** και «**initial**» για αυτές της τις μεταβλητές αντίστοιχα. Για να αξιοποιηθούν σωστά αυτές οι μεταβλητές αποδεχόμαστε πως οι αρχικές καταστάσεις παράγονται από την συνάρτηση **fillRandomly()** η οποία αναθέτει τις παραπάνω τιμές στις αντίστοιχες μεταβλητές όταν καλείται. Επίσης αποδεχόμαστε πως οι καταστάσεις παιδιά παράγονται με τον εξής τρόπο:

1. Δημιουργούμε ένα αντίγραφο της παρούσας κατάστασης με έναν νέο κονστράκτορα, ο οποίος δέχεται ως όρισμα pointer στην παρούσα κατάσταση, τον οποίο αναθέτει και ως τιμή στη μεταβλητή parent.
2. Εκτελούμε την συνάρτηση move(χ, ψ) η οποία τώρα, πέρα της αρχικής της λειτουργείας, αναθέτει και το string “from χ to ψ” στην μεταβλητή instruction.

### Συνάρτηση Σύγκρισης

Για να συγκρίνουμε καταστάσεις θα μπορούσαμε απλά να βλέπουμε αν όλα τα δοχεία έχουν τα ίδια περιεχόμενα, όμως θα επιβαρύναμε το πρόγραμμα με καταστάσεις που είναι ισοδύναμες (δηλαδή όπου το ίδιο δοχείο με τα ίδια περιεχόμενα βρίσκεται απλά σε άλλη θέση). Για να βελτιώσουμε τον αλγόριθμο μπορούμε να ταξινομούμε τα δοχεία με βάση τα περιεχόμενά τους.

Ο τρόπος με τον οποίο υλοποιείται αυτό το μέρος του προγράμματος είναι ίσως μπερδεμένος…

Προσθέτουμε στην κλάση την μεταβλητή **code**, έναν πίνακα από 20 ακεραίους, ο οποίος αρχικοποιείται με μηδενικά. Σε κάθε θέση του πίνακα αυτού τοποθετείται ένας ακέραιος, που αναπαριστά τα περιεχόμενα των δοχείων. Για να έχουμε αυτούς τους ακεραίους, ορίζουμε επίσης την συνάρτηση **toInt()** στην κλάση Beaker. Με την συνάρτηση **generateCode()** παίρνουμε όλους αυτούς τους τετραψήφιους, τους ταξινομούμε και τους εισάγουμε στην **code** η οποία μέχρι τώρα είχε αρχικοποιηθεί με μηδενικά.

Όπως και πριν σιγουρευόμαστε πως ο κωδικός παράγεται για κάθε νέα κατάσταση, δηλαδή όταν καλούμε την fillRandomly() και όταν καλούμε την move().

### Επέκταση Κατάστασης

Παραλείπω τους *Τελεστές Μετάβασης* και τον *Έλεγχο Εφαρμοσιμότητας* καθώς και τα δύο υλοποιούνται στην συνάρτηση move() η οποία έχει αναλυθεί παραπάνω.

Για την επέκταση μίας κατάστασης ορίζουμε την συνάρτηση **expand()**η οποία καλεί την move() με όλους τους πιθανούς συνδυασμούς δοχείων και επιστρέφει ένα vector με pointers σε όλες τις έγκυρες νέες καταστάσεις.

### Υλοποίηση BFS

Έχουμε πλέον (σχεδόν) όλα τα απαραίτητασυστατικά για να υλοποιήσουμε τον αλγόριθμο BFS. Αυτό το κάνουμε πάλι με προσχέδιο τις διαφάνειες του μαθήματος, με δύο τρόπους: χρησιμοποιώντας vector και unordered\_map για την αναπαράσταση του κλειστού συνόλου.

Η λεπτομέρεια που χρειάζεται για να λειτουργήσει το unordered\_map είναι η συνάρτηση κατακερματισμού, η οποία δεν υλοποιείται (ή τουλάχιστον δεν κατάφερα εγώ να την υλοποιήσω) με πίνακα ακεραίων, οπότε αντί για τον code της κατάστασης χρησιμοποιούμε ένα string με τα περιεχόμενα του code και μηδενικά στις απαραίτητες κενές θέσεις για να αποφύγουμε λάθη.

Τέλος ορίζουμε και την συνάρτηση **printPath()** στην main του προγράμματος, η οποία παίρνει την τελική κατάσταση που βρήκαμε, ακολουθεί τους parents μέχρι την αρχική, και έπειτα εκτυπώνει τις καταστάσεις με ανάποδη σειρά, έτσι ώστε να ξεκινάει από την αρχική και να τελειώνει στην τελική κατάσταση.

## Σημειώσεις

Πριν δώσω τα παραδείγματα εκτέλεσης του αλγορίθμου θέλω να αναφέρω τα εξής:

1. Η χρήση δυναμικά δημιουργούμενων αντικειμένων και δεικτών σε αυτά τα αντικείμενα αποτέλεσαν εμπόδιο και κάποια κομμάτια του προγράμματος που αφορούν στην αυτόματη επίλυση του προβλήματος είναι ως αποτέλεσμα «μπαλωμένα», όπως η διαφορά της συνάρτησης κατακερματισμού από την ήδη υπάρχουσα μεταβλητή code που θα έπρεπε ίσως να ταυτίζονται.
2. Το πρόγραμμα είναι τεχνικά **ατελές** καθώς οι τετραψήφιοι αριθμοί που χρησιμοποιούνται για την αναπαράσταση των δοχείων θα προκαλούσαν σφάλματα όταν χρειαζόμαστε παραπάνω από δύο ψηφία για την αναπαράσταση υγρών (n>=12) . Όμως παραλείπω την διόρθωση σε αυτό το κομμάτι, η οποία θα απαιτούσε μεγάλη προσπάθεια λόγω της 3ης σημείωσης.
3. Το πρόγραμμα αδυνατεί να λειτουργήσει για περισσότερα από 10 δοχεία. Ενώ δεν υπάρχει απαραίτητα κάποιο λάθος μέχρι να χρησιμοποιήσουμε 12 δοχεία, το φόρτο εργασίας είναι τόσο μεγάλο και οι απαιτούμενες αναζητήσεις τόσες πολλές, που με την παρούσα υλοποίηση ισοδυναμούν με ατέρμονα βρόγχο. Θα μπορούσαμε μάλλον να υλοποιήσουμε μικρές βελτιώσεις και να μειώσουμε την πολυπλοκότητα σε διάφορα κομμάτια του κώδικα, αλλά υπάρχει μάλλον κάποιο όριο το οποίο είναι μικρότερο από 20 δοχεία και το οποίο για να ξεπεραστεί θα έπρεπε να χρησιμοποιήσουμε διαφορετική λογική επίλυσης (πχ. ευρετικό αλγόριθμο)
4. Η ιδέα να ταξινομήσω τα περιεχόμενα των καταστάσεων μου δόθηκε από τον Σταύρο Νικολαΐδη, και τον ευχαριστώ!
5. Ο αριθμός δοχείων μπορεί να αλλάξει από την σταθερά **Ν** στη main

# Παραδείγματα Χρήσης

Χρησιμοποιούμε την μέθοδο με unordered\_map αλλά συμπεριλαμβάνεται και η μέθοδος με vector στη main

## 4 Δοχεία

initial

+++++++++++++++

0 | 0001

1 | 0001

2 | 2211

3 | 0022

+++++++++++++++

from 0 to 1

+++++++++++++++

0 | 0000

1 | 0011

2 | 2211

3 | 0022

+++++++++++++++

from 2 to 3

+++++++++++++++

0 | 0000

1 | 0011

2 | 0011

3 | 2222

+++++++++++++++

from 1 to 2

+++++++++++++++

0 | 0000

1 | 0000

2 | 1111

3 | 2222

+++++++++++++++

depth = 4, Mem: 18, Examined: 8

Process finished with exit code 0

## 6 Δοχεία

initial

+++++++++++++++

0 | 3322

1 | 0042

2 | 0004

3 | 3111

4 | 0432

5 | 0041

+++++++++++++++

from 1 to 2

+++++++++++++++

0 | 3322

1 | 0002

2 | 0044

3 | 3111

4 | 0432

5 | 0041

+++++++++++++++

from 4 to 2

+++++++++++++++

0 | 3322

1 | 0002

2 | 0444

3 | 3111

4 | 0032

5 | 0041

+++++++++++++++

from 0 to 4

+++++++++++++++

0 | 0022

1 | 0002

2 | 0444

3 | 3111

4 | 3332

5 | 0041

+++++++++++++++

from 0 to 1

+++++++++++++++

0 | 0000

1 | 0222

2 | 0444

3 | 3111

4 | 3332

5 | 0041

+++++++++++++++

from 3 to 0

+++++++++++++++

0 | 0003

1 | 0222

2 | 0444

3 | 0111

4 | 3332

5 | 0041

+++++++++++++++

from 4 to 0

+++++++++++++++

0 | 3333

1 | 0222

2 | 0444

3 | 0111

4 | 0002

5 | 0041

+++++++++++++++

from 1 to 4

+++++++++++++++

0 | 3333

1 | 0000

2 | 0444

3 | 0111

4 | 2222

5 | 0041

+++++++++++++++

from 5 to 2

+++++++++++++++

0 | 3333

1 | 0000

2 | 4444

3 | 0111

4 | 2222

5 | 0001

+++++++++++++++

from 3 to 5

+++++++++++++++

0 | 3333

1 | 0000

2 | 4444

3 | 0000

4 | 2222

5 | 1111

+++++++++++++++

depth = 10, Mem: 335, Examined: 257

Process finished with exit code 0

## 9 Δοχεία

initial

+++++++++++++++

0 | 0742

1 | 6211

2 | 0061

3 | 6533

4 | 0062

5 | 7431

6 | 0754

7 | 7532

8 | 0054

+++++++++++++++

from 0 to 6

+++++++++++++++

0 | 0042

1 | 6211

2 | 0061

3 | 6533

4 | 0062

5 | 7431

6 | 7754

7 | 7532

8 | 0054

+++++++++++++++

from 1 to 2

+++++++++++++++

0 | 0042

1 | 0211

2 | 0661

3 | 6533

4 | 0062

5 | 7431

6 | 7754

7 | 7532

8 | 0054

+++++++++++++++

from 3 to 2

+++++++++++++++

0 | 0042

1 | 0211

2 | 6661

3 | 0533

4 | 0062

5 | 7431

6 | 7754

7 | 7532

8 | 0054

+++++++++++++++

from 8 to 3

+++++++++++++++

0 | 0042

1 | 0211

2 | 6661

3 | 5533

4 | 0062

5 | 7431

6 | 7754

7 | 7532

8 | 0004

+++++++++++++++

from 0 to 8

+++++++++++++++

0 | 0002

1 | 0211

2 | 6661

3 | 5533

4 | 0062

5 | 7431

6 | 7754

7 | 7532

8 | 0044

+++++++++++++++

from 0 to 1

+++++++++++++++

0 | 0000

1 | 2211

2 | 6661

3 | 5533

4 | 0062

5 | 7431

6 | 7754

7 | 7532

8 | 0044

+++++++++++++++

from 2 to 0

+++++++++++++++

0 | 0666

1 | 2211

2 | 0001

3 | 5533

4 | 0062

5 | 7431

6 | 7754

7 | 7532

8 | 0044

+++++++++++++++

from 4 to 0

+++++++++++++++

0 | 6666

1 | 2211

2 | 0001

3 | 5533

4 | 0002

5 | 7431

6 | 7754

7 | 7532

8 | 0044

+++++++++++++++

from 1 to 4

+++++++++++++++

0 | 6666

1 | 0011

2 | 0001

3 | 5533

4 | 0222

5 | 7431

6 | 7754

7 | 7532

8 | 0044

+++++++++++++++

from 1 to 2

+++++++++++++++

0 | 6666

1 | 0000

2 | 0111

3 | 5533

4 | 0222

5 | 7431

6 | 7754

7 | 7532

8 | 0044

+++++++++++++++

from 5 to 1

+++++++++++++++

0 | 6666

1 | 0007

2 | 0111

3 | 5533

4 | 0222

5 | 0431

6 | 7754

7 | 7532

8 | 0044

+++++++++++++++

from 5 to 8

+++++++++++++++

0 | 6666

1 | 0007

2 | 0111

3 | 5533

4 | 0222

5 | 0031

6 | 7754

7 | 7532

8 | 0444

+++++++++++++++

from 6 to 1

+++++++++++++++

0 | 6666

1 | 0777

2 | 0111

3 | 5533

4 | 0222

5 | 0031

6 | 0054

7 | 7532

8 | 0444

+++++++++++++++

from 3 to 6

+++++++++++++++

0 | 6666

1 | 0777

2 | 0111

3 | 0033

4 | 0222

5 | 0031

6 | 5554

7 | 7532

8 | 0444

+++++++++++++++

from 5 to 3

+++++++++++++++

0 | 6666

1 | 0777

2 | 0111

3 | 0333

4 | 0222

5 | 0001

6 | 5554

7 | 7532

8 | 0444

+++++++++++++++

from 2 to 5

+++++++++++++++

0 | 6666

1 | 0777

2 | 0000

3 | 0333

4 | 0222

5 | 1111

6 | 5554

7 | 7532

8 | 0444

+++++++++++++++

from 6 to 2

+++++++++++++++

0 | 6666

1 | 0777

2 | 0555

3 | 0333

4 | 0222

5 | 1111

6 | 0004

7 | 7532

8 | 0444

+++++++++++++++

from 6 to 8

+++++++++++++++

0 | 6666

1 | 0777

2 | 0555

3 | 0333

4 | 0222

5 | 1111

6 | 0000

7 | 7532

8 | 4444

+++++++++++++++

from 7 to 1

+++++++++++++++

0 | 6666

1 | 7777

2 | 0555

3 | 0333

4 | 0222

5 | 1111

6 | 0000

7 | 0532

8 | 4444

+++++++++++++++

from 7 to 2

+++++++++++++++

0 | 6666

1 | 7777

2 | 5555

3 | 0333

4 | 0222

5 | 1111

6 | 0000

7 | 0032

8 | 4444

+++++++++++++++

from 7 to 3

+++++++++++++++

0 | 6666

1 | 7777

2 | 5555

3 | 3333

4 | 0222

5 | 1111

6 | 0000

7 | 0002

8 | 4444

+++++++++++++++

from 4 to 7

+++++++++++++++

0 | 6666

1 | 7777

2 | 5555

3 | 3333

4 | 0000

5 | 1111

6 | 0000

7 | 2222

8 | 4444

+++++++++++++++

depth = 23, Mem: 6996, Examined: 6795

Process finished with exit code 0