Τεχνητή Νοημοσύνη

Άσκηση 1η

Ρετσίνας Γιώργος Α.Μ.: 03107029
Τσούτσουρας Βασίλης Α.Μ.: 03107182

1.

Σκοπός της άσκησης, είναι η κατασκευή ενός προγράμματος, το οποίο κάνοντας χρήση ενός αλγορίθμου εύρεσης βέλτιστης λύσης με ευριστικές τεχνικές, θα καθοδηγεί τις κινήσεις 2 ρομπότ ώστε να μεταβούν με τον ελάχιστο αριθμό κινήσεων από μία αρχική θέση του χώρου σε μία τελική. Την επίλυση του προβλήματος, δυσχεραίνει το γεγονός, ότι στον χώρο υπάρχουν εμπόδια τα οποία τα ρομπότ πρέπει να αποφύγουν. Ακόμα, επειδή τα ρομπότ θεωρητικά κινούνται ταυτόχρονα, πρέπει να φροντίσουμε να μην συγκρουστούν, δηλαδή να μην βρίσκονται ταυτόχρονα στην ίδια θέση.

Για την επίλυση του προβλήματος, αρχικά έπρεπε να δημιουργήσουμε ένα μοντέλο για τα δεδομένα εισόδου. Επιλέχτηκε, αυτό να είναι ένα αρχείο εισόδου, στο οποίο δίνονται σε καρτεσιανές συντεταγμένες, οι διαστάσεις NxM του χώρου στον οποίο κινούνται τα ρομπότ, οι αρχικές τους θέσεις και η θέση στην οποία πρέπει να φτάσουν. Έπειτα ακολουθεί ένα κείμενο Ν γραμμών και Μ στηλών που αναπαριστά κάθε σημείο του χώρου και σε αυτό σημειώνεται Χ αν υπάρχει κάποιο εμπόδιο ή Ο αν η θέση είναι κενή. Οι συντεταγμένες x και y αριθμούνται από 1..N και 1 έως 1..M αντίστοιχα και η πρώτη γραμμή του κειμένου αντιστοιχεί στα σημεία με x = 1 και y από 1 έως Μ.

Ο αλγόριθμός εύρεσης βέλτιστης λύσης που χρησιμοποιείται είναι ο A* σε δύο διαφορετικές του εκδόσεις, μία με υπό-εκτιμητή και μία με υπέρ-εκτιμητή. Για το δέντρο των πιθανών κινήσεων, το οποίο κάνει χρήση ο αλγόριθμος, έχει δημιουργηθεί μια ειδική δομή η οποία θα αναλυθεί λεπτομερώς παρακάτω. Πρακτικά, η ταυτόχρονη εξέταση των κινήσεων των δύο ρομπότ είναι δύσκολη κ είναι πιθανό ότι θα οδηγήσει σε εσφαλμένο αποτέλεσμα. Έτσι καθορίζουμε την πορεία του πρώτου ρομπότ και με βάση την θέση του σε κάθε βήμα, υπολογίζουμε και την πορεία του δεύτερου. Για την υλοποίηση του A*, έγινε χρήση μιας ουράς προτεραιότητας ελαχίστου ώστε σε κάθε βήμα να εξετάζεται η κίνηση που ελαχιστοποιεί τον εκτιμητή. Για την εύρεση των σημείων του μονοπατιού του κάθε ρομπότ, όταν ολοκληρωθεί η αναζήτηση, διασχίζουμε το δέντρο από το φύλλο του, που αντιστοιχεί στο τελικό σημείο, έως την ρίζα του, σημειώνοντας τις συντεταγμένες των κόμβων που συναντάμε κατά την διάσχιση.

2.

Η ουρά προτεραιότητας του δέντρου αναζήτησης του A*, είναι ένας δυαδικός σωρός ο οποίος υλοποιήθηκε με ένα πίνακα. Ο πίνακας ουσιαστικά αναπαριστά το αντίστοιχο δυαδικό δέντρο του σωρού. Για το στοιχείο i του πίνακα, το i/2 συμβολίζει τον πατέρα του, το 2 i το αριστερό παιδί του και το 2i + 1 το δεξί παιδί του. Στον σωρό ισχύει ότι κάθε πατέρας έχει μικρότερη τιμή χαρακτηριστικού από τα παιδία του. Υπάρχουν τρεις

συναρτήσεις που σχετίζονται με την ουρά. Η insert η οποία εισάγει ένα νέο στοιχείο στην ουρά, η combine η οποία τοποθετεί το στοιχείο στην σωστή του θέση στον πίνακα και η q_min η οποία επιστρέφει το ελάχιστο των στοιχείων και φροντίζει για την προσκομιδή του επόμενου μικρότερου στην κορυφή της ουράς. Η τοποθέτηση των στοιχείων του πίνακα, γίνεται με βάση την τιμή F(S) του A*. η τιμή του κάθε στοιχείου του πίνακα, είναι η διεύθυνση του κόμβου της αντίστοιχης θέσης στο δέντρο αναζήτησης του A*.

Για το δέντρο αναζήτησης, έγινε χρήση της ακόλουθης δομής:

```
typedef int state_type;
typedef int g_type;
typedef double h_type;

typedef struct Node{
        struct Node *parent;
        int x;
        int y;
        state_type state;
        g_type g; //g_type = g_type of par + 1
        h_type h;
        h_type dist;
        int next; //0 open / 1 closed
        int pos;
        struct Node *r_sibling;
        struct Node *child1;
} SearchGraphNode;
```

Κάθε κόμβος αναπαριστά ένα σημείο του χώρου. Οι τιμές x και y αντιστοιχούν στις συντεταγμένες του σημείου αυτού. Ο δείκτης parent δείχνει στον πατέρα του κάθε κόμβου στο δέντρο. Η μεταβλητή state είναι απλά ένα μοναδικό όνομα του κόμβου. Αν τι για A,B κλπ επιλέχτηκαν ακέραιοι σε αύξουσα σειρά για να έχουμε ευκολότερη αύξηση της τιμής. Τα g και h συμβολίζουν την απόσταση από την αρχική και την τελική θέση αντίστοιχα. Η dist είναι το άθροισμα των δύο προηγούμενων τιμών και χρησιμοποιείται για την ταξινόμηση του κόμβου στην ουρά προτεραιότητας. Η next είναι 0 όταν ο κόμβος μπορεί να χρησιμοποιηθεί για περαιτέρω επέκταση το δέντρου και 1 όταν ο κόμβος διακόπτεται από το σκέλος δυναμικού προγραμματισμού του A*. Η μεταβλητή pos δείχνει σε ποιο στοιχείο του πίνακα της ουράς προτεραιότητας, αντιστοιχεί ο εν λόγω κόμβος, για να μπορούμε να το βρούμε γρήγορα όταν πρέπει να του αλλάξουμε θέση στον πίνακα. Τέλος, οι δείκτες r_sibling και r_child1, δείχνουν ο πρώτος στον επόμενο κόμβο του δέντρου στο ίδιο ύψος και ό δεύτερος στο πρώτο παιδί του κόμβου.

Τέλος, αξίζει να σημειωθεί η ύπαρξη δύο πινάκων των dyn_rob1 και dyn_rob2 οι οποίοι κρατούν πληροφορία για τον ελάχιστο αριθμό κινήσεων με τον οποίον τα ρομπότ 1 και 2 αντίστοιχα, έφτασαν στην αντίστοιχη θέση. Έτσι, υλοποιείται το σκέλος του δυναμικού προγραμματισμού του αλγορίθμου. Οι πίνακες αυτοί αρχικοποιούνται με -1 που συμβολίζει ότι το ρομπότ δεν έχει εξετάσει ποτέ την αντίστοιχη θέση.

3.

Για να μπορεί να υλοποιηθεί ένα μεγάλο κομμάτι του κώδικα επαναληπτικά, αντί για τέσσερεις συναρτήσεις τελεστές, μία για κάθε πιθανή κίνηση του ρομπότ από μια δεδομένη θέση, δημιουργήθηκε μια συνάρτηση η οποία παίρνει σαν όρισμα τον τύπο της κίνησης και την τρέχουσα θέση και επιστρέφει τις συντεταγμένες της επόμενης θέσης , σύμφωνα με τον τύπο της κίνησης. Η συνάρτηση αυτή είναι:

```
void move(int type,int cur_x,int cur_y,int *new_x,int *new_y){
	if (type == 0){//up
		*new_x = cur_x-1;
		*new_y = cur_y;
	}

	if (type == 1){//down
		*new_x = cur_x+1;
		*new_y = cur_y;
	}

	if (type == 2){//left
		*new_x = cur_x;
		*new_y = cur_y-1;
	}

	if (type == 3){//right
		*new_x = cur_x;
		*new_y = cur_y+1;
	}
}
```

Με cur_x, cur_y συμβολίζουμε τις συντεταγμένες της θέσης που βρίσκεται το ρομπότ και με new_x, new_y, τις συντεταγμένες που θα εξετάσει το ρομπότ για πιθανή του μετάβαση.

3.

Ως υπό-εκτιμητής χρησιμοποιήθηκε η ευκλείδεια απόσταση καθώς συμπίπτει με την πραγματική απόσταση ενός σημείου από το σημείο εξόδου. Ως υπέρ-εκτιμητή χρησιμοποιήσαμε το άθροισμα της ευκλείδειας απόστασης και του δεκαπλασίου της απόστασης Manhattan. Εφόσον η απόσταση Manhattan είναι μη αρνητική, το άθροισμα τους θα είναι μεγαλύτερο από την ευκλείδεια παντού εκτός από την τελική θέση, γεγονός που το καθιστά υπέρ-εκτιμητή.

4.

Ο αλγόριθμος παρατίθεται με την μορφή ψευδοκώδικα:

Αρχικοποίηση των δομών δεδομένων.
Δημιουργία της ρίζας του δέντρου αναζήτησης του πρώτου ρομπότ.

```
Όσο τρέχων_κόμβος <> τερματικού {
        Φτιάξε μια καινούρια θέση με τη συναρτήσεις μετάβασης για όλες τις πιθανές
        μεταβάσεις
        Αν δεν βγαίνει εκτός του χώρου και δεν πέφτει σε εμπόδιο
                Πρόσθεσε την στο δέντρο αναζήτησης

        Αν η καινούρια θέση πληροί τα κριτήρια του δυναμικού προγραμματισμού
                Σημείωσε  next=0
                Ανανέωσε τα δεδομένα του δυναμικού προγραμματισμού
                Πρόσθεσε την στην ουρά ελαχίστου
        Αλλιώς
                Σημείωσε  next=1

        Εξαγωγή από την ουρά του κόμβου min με την ελάχιστη εκτιμώμενη απόσταση
        τρέχων_κόμβος  = new
}
```

Ακολούθα του κόμβους του δέντρου, με αρχή αυτό που δείχνει ο τρέχων_κόμβος, μέχρις ότου φτάσεις στην ρίζα του δέντρου, για να βρεις το ακριβές μονοπάτι της κίνησης.

Αρχικοποίησε εκ νέου την ουρά ελαχίστου
Δημιουργία της ρίζας του δέντρου αναζήτησης του δεύτερου ρομπότ.

```
Όσο τρέχων_κόμβος <> τερματικού {
        Φτιάξε μια καινούρια θέση με τις συναρτήσεις μετάβασης για όλες τις πιθανές
        μεταβάσεις

        Αν δεν βγαίνει εκτός του χώρου και δεν πέφτει σε εμπόδιο και δεν βρίσκεται το
                ρομπότ 1 σε αυτή την θέση την δεδομένη στιγμή

                Πρόσθεσε την στο δέντρο αναζήτησης

        Αν η καινούρια θέση πληροί τα κριτήρια του δυναμικού προγραμματισμού
                Σημείωσε  next=0
                Ανανέωσε τα δεδομένα του δυναμικού προγραμματισμού
                Πρόσθεσε την στην ουρά ελαχίστου
        Αλλιώς
                Σημείωσε  next=1

        Εξαγωγή από την ουρά του κόμβου min με την ελάχιστη εκτιμώμενη απόσταση
        τρέχων_κόμβος = new
}
```

Ακολούθα του κόμβους του δέντρου, με αρχή αυτό που δείχνει ο τρέχων_κόμβος, μέχρις ότου φτάσεις στην ρίζα του δέντρου, για να βρεις το ακριβές μονοπάτι της κίνησης.

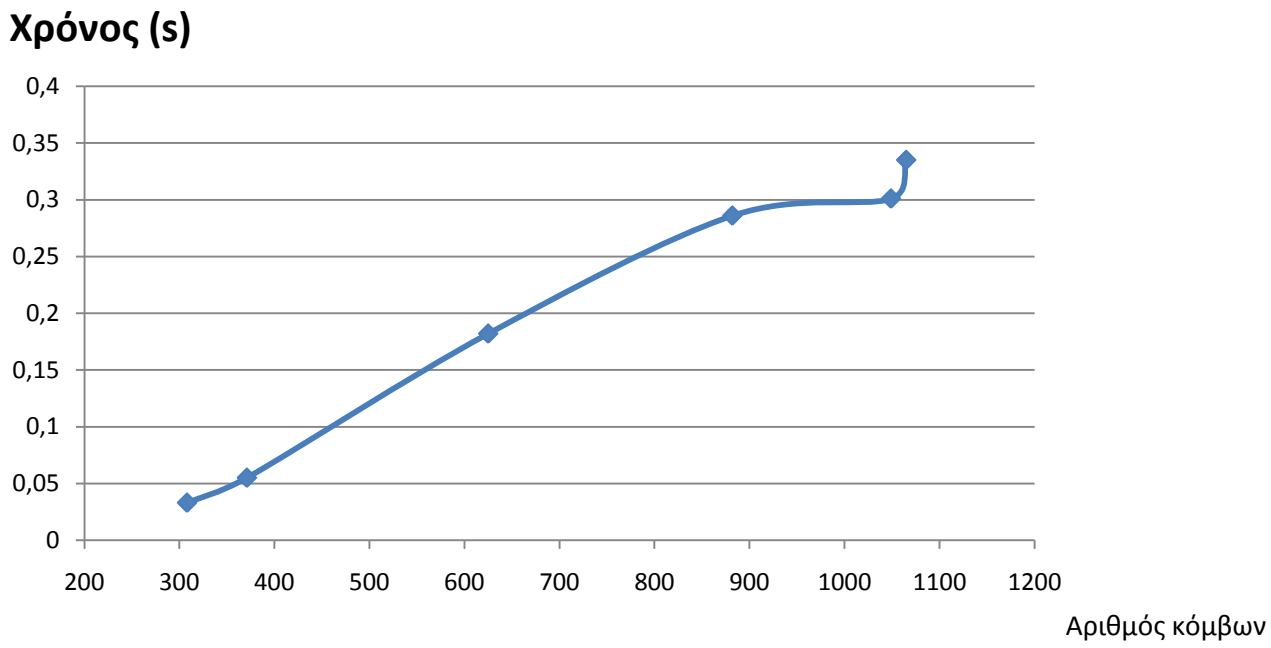
Τέλος_αλγορίθμου

5.
Ο χρόνος μετρήθηκε με την συνάρτηση time του λειτουργικού συστήματος linux. Τα αρχεία εισόδου κατευθύνθηκαν στο standard input με την εντολή:

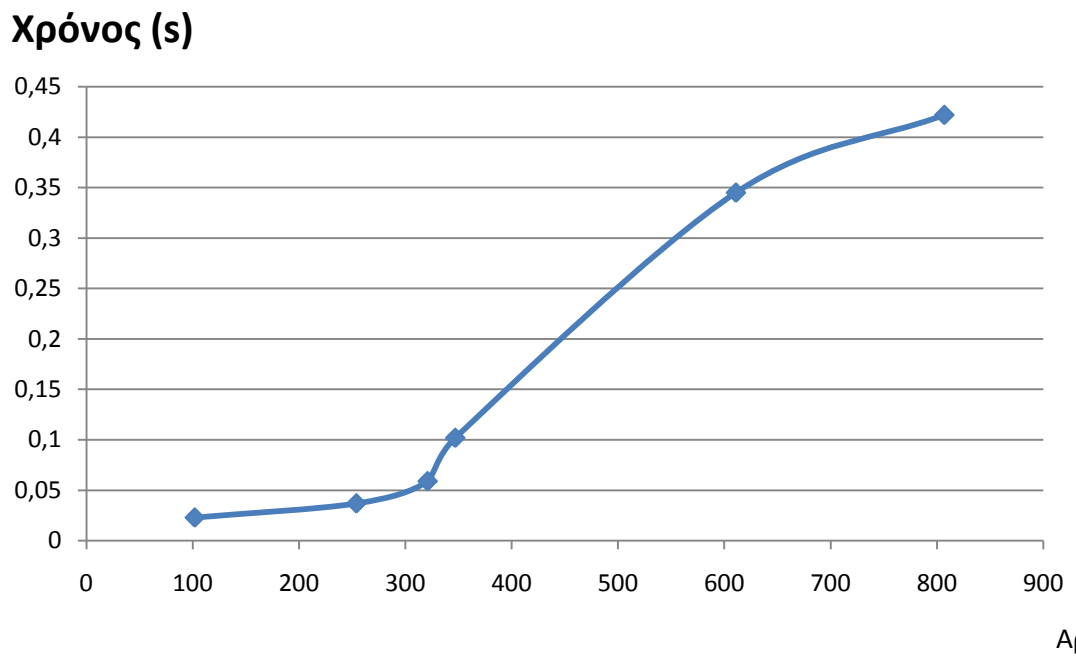
time ./robot.out < in1.txt

Κατά την μέτρηση του χρόνου, δεν αφαιρέθηκαν τα μηνύματα εξόδου.

Έτσι προκύπτει το διάγραμμα:



Παρατηρούμε ότι αύξηση του αριθμού των κόμβων του δέντρου, συνοδεύεται από αύξηση του χρόνου, πολλές φορές και γραμμική.



Βλέπουμε ότι με τον υπέρ-εκτιμητή, το πρόγραμμα εξετάζει λιγότερους κόμβους και τελειώνει πιο γρήγορα, αυτό συμβαίνει γιατί η μεγάλη τιμή του εκτιμητή, οξύνει τις διαφορές και εισάγει λιγότερους κόμβους στο δέντρο. Αυτό θα μπορούσε να αλλάξει κάνοντας χρήση ενός μεταβλητού εκτιμητή, οποίο όμως δεν θα είχε σχέση με την απόσταση, όπως π.χ. το τετράγωνο του αριθμού των βημάτων που έχουν γίνει.

6.

Παρατίθενται τα μηνύματα εξόδου για το αρχείο εισόδου 4 (φαίνεται στο τέλος της αναφοράς). Όμως λόγω του πολύ μεγάλου μεγέθους τους, παρατίθενται μόνο για τις κινήσεις του δεύτερου ρομπότ, πού έχει όλες τις περιπτώσεις σύγκρουσης και μάλιστα για ένα μικρό σχετικά τμήμα των κινήσεων αυτών.

<Robot 2 considering new position <6,1> at step 1
<Robot 2 considering new position <8,1> at step 1
<Robot 2 considering new position <7,0> at step 1
<Robot 2 ignoring position <7,0> because it is out of bounds.
<Robot 2 considering new position <7,2> at step 1
<Robot 2 moving to new position <7,2> at step 1
<Robot 2 considering new position <6,2> at step 2
<Robot 2 considering new position <8,2> at step 2
<Robot 2 considering new position <7,1> at step 2
<Robot 2 considering new position <7,3> at step 2
<Robot 2 moving to new position <7,3> at step 2
<Robot 2 considering new position <6,3> at step 3
<Robot 2 considering new position <8,3> at step 3
<Robot 2 considering new position <7,2> at step 3
<Robot 2 considering new position <7,4> at step 3
<Robot 2 moving to new position <7,4> at step 3

<Robot 2 considering new position <6,4> at step 4
<Robot 2 considering new position <8,4> at step 4
<Robot 2 considering new position <7,3> at step 4
<Robot 2 considering new position <7,5> at step 4
<Robot 2 moving to new position <7,5> at step 4
<Robot 2 considering new position <6,5> at step 5
<Robot 2 considering new position <8,5> at step 5
<Robot 2 considering new position <7,4> at step 5
<Robot 2 considering new position <7,6> at step 5
<Robot 2 moving to new position <7,6> at step 5
<Robot 2 considering new position <6,6> at step 6
<Robot 2 considering new position <8,6> at step 6
<Robot 2 considering new position <7,5> at step 6
<Robot 2 considering new position <7,7> at step 6
<Robot 2 moving to new position <7,7> at step 6
<Robot 2 considering new position <6,7> at step 7
<Robot 2 ignoring position <6,7> because robot 1 is currently there
<Robot 2 considering new position <8,7> at step 7
<Robot 2 considering new position <7,6> at step 7
<Robot 2 considering new position <7,8> at step 7
<Robot 2 ignoring position <7,8> because of an obstacle.
<Robot 2 moving to new position <8,3> at step 3
<Robot 2 considering new position <7,3> at step 4
<Robot 2 considering new position <9,3> at step 4
<Robot 2 considering new position <8,2> at step 4
<Robot 2 considering new position <8,4> at step 4
<Robot 2 moving to new position <8,7> at step 7
<Robot 2 considering new position <7,7> at step 8
<Robot 2 ignoring position <7,7> because robot 1 is currently there
<Robot 2 considering new position <9,7> at step 8
<Robot 2 considering new position <8,6> at step 8
<Robot 2 considering new position <8,8> at step 8
<Robot 2 moving to new position <8,5> at step 5
<Robot 2 considering new position <7,5> at step 6
<Robot 2 considering new position <9,5> at step 6
<Robot 2 considering new position <8,4> at step 6
<Robot 2 considering new position <8,6> at step 6
<Robot 2 moving to new position <8,8> at step 8
<Robot 2 considering new position <7,8> at step 9
<Robot 2 ignoring position <7,8> because of an obstacle.
<Robot 2 considering new position <9,8> at step 9
<Robot 2 considering new position <8,7> at step 9
<Robot 2 ignoring position <8,7> because robot 1 is currently there
<Robot 2 considering new position <8,9> at step 9
<Robot 2 ignoring position <8,9> because of an obstacle.

<Robot 2 moving to new position <8,4> at step 4

7.

Για το ίδιο αρχείο εισόδου παρατίθενται οι πορείες των δύο ρομπότ.

```
Path of robot 1:
Initial position = <1,9>
Position at step 1 = <2,9>
Position at step 2 = <3,9>
Position at step 3 = <4,9>
Position at step 4 = <5,9>
Position at step 5 = <5,8>
Position at step 6 = <6,8>
Position at step 7 = <6,7>
Position at step 8 = <7,7>
Position at step 9 = <8,7>
Position at step 10 = <8,8>
Position at step 11 = <9,8>
Position at step 12 = <9,9>
Position at step 13 = <10,9>
Position at step 14 = <10,10>
Position at step 15 = <10,11>
Position at step 16 = <10,12>
Position at step 17 = <10,13>
Position at step 18 = <10,14>
Position at step 19 = <11,14>
Position at step 20 = <12,14>
Position at step 21 = <12,15>
Position at step 22 = <12,16>
Position at step 23 = <11,16>
Position at step 24 = <10,16>
Position at step 25 = <9,16>
Position at step 26 = <8,16>
Position at step 27 = <8,17>

Path of robot 2:
Initial position = <7,1>
Position at step 1 = <7,2>
Position at step 2 = <7,3>
Position at step 3 = <7,4>
Position at step 4 = <7,5>
Position at step 5 = <7,6>
Position at step 6 = <7,7>
Position at step 7 = <8,7>
Position at step 8 = <8,8>
Position at step 9 = <9,8>
Position at step 10 = <9,9>
Position at step 11 = <10,9>
Position at step 12 = <10,10>
Position at step 13 = <10,11>
Position at step 14 = <10,12>
Position at step 15 = <10,13>
Position at step 16 = <10,14>
Position at step 17 = <11,14>
```

```
Position at step 18 = <12,14>
Position at step 19 = <12,15>
Position at step 20 = <12,16>
Position at step 21 = <11,16>
Position at step 22 = <10,16>
Position at step 23 = <9,16>
Position at step 24 = <8,16>
```

Αρχεία Εισόδου:

1ο

```
13 20
1 9
13 9
8 17
XOOOOOOOOOOOOOOOOOXX
XOOOOOOOOOOOOOOOOOXX
XOOOOOOOOOOOOOOOOOXX
XOOOOOOOOOOOOOOOOOXX
OOOOOOOOOXOOOOOOOOOO
OOOOOOOOXXXOOOOOOOOX
OOOOOOOXXXXXOOOOOOOX
OOOOOOOOXXXOOOOOOOOX
OOOOOOOOOXOOOOOOOOOO
XOOOOOOOOOOOOOOOOOOO
XOOOOOOOOOOOOOOOOOXX
XOOOOOOOOOOOOOOOOOXX
XOOOOOOOOOOOOOOOOOXX
```

2ο

```
13 20
1 9
13 9
8 17
XOOOOOOOOOOOOOXOOOXX
XOOOOOOOOOOOOOOOOOXX
XOOOOOOOOOOOOOXOOOXX
XOOOOOOOOOOOOOXOOOXX
OOOOOOOOOXOOOOXOOOOO
OOOOOOOOXXXOOOXOOOOX
OOOOOOOXXXXXOOXOOOOX
OOOOOOOOXXXOOOXOOOOX
OOOOOOOOOXOOOOXOOOOO
XOOOOOOOOOOOOOXOOOOO
XOOOOOOOOOOOOOXOOOXX
XOOOOOOOOOOOOOOOOOXX
XOOOOOOOOOOOOOXOOOXX
```

3ο

```
13 20
1 9
13 9
8 17
XOOOOOOOOOOOOOXOOOXX
XOOOOOOOOOOOOOXOOOXX
XOOOOOOOOOOOOOXOOOXX
```

XOOOOOOOOOOOOOXOOOXX
OOOOOOOOOXXXXXXOOOOO
OOOOOOOOXXXOOOXOOOOX
OOOOOOOXXXXXOOXOOOOX
OOOOOOOOXXXOOOXOOOOX
OOOOOOOOOXOOOOXOOOOO
XOOOOOOOOOOOOOXOOOOO
XOOOOOOOOOOOOOXOOOXX
XOOOOOOOOOOOOOOOOOXX
XOOOOOOOOOOOOOXOOOXX

4º

13 20
1 9
7 1
8 17
XOOOOOOOOOOOOOXOOOXX
XOOOOOOOOOOOOOXOOOXX
XOOOOOOOOOOOOOXOOOXX
XOOOOOOOOOOOOOXOOOXX
OOOOOOOOOXXXXXXOOOOO
OOOOOOOOXXXOOOXOOOOX
OOOOOOOXXXXXOOXOOOOX
OOOOOOOOXXXOOOXOOOOX
OOOOOOOOOXOOOOXOOOOO
XOOOOOOOOOOOOOXOOOOO
XOOOOOOOOOOOOOXOOOXX
XOOOOOOOOOOOOOOOOOXX
XOOOOOOOOOOOOOXOOOXX

5º

13 20
1 9
7 1
8 17
XOOOOOOOOOOOOOXOOOXX
XOOOOOOOOOOOOOXOOOXX
XOOOOOOOOOOOOOXOOOXX
XOOOOOOOOOOOOOXOOOXX
OOOOOOOOOXXXXXXOOOOO
OOOOOOOOXXXOOOXOOOOX
OOOOOOOXXXXXOOXOOOOX
OOOOOOOOXXXOOOXOOOOX
OOXXXXXXXXOOOOXOOOOO
XOOOOOOOOOOOOOXOOOOO
XOOOOOOOOOOOOOXOOOXX
XOOOOOOOOOOOOOOOOOXX
XOOOOOOOOOOOOOXOOOXX

6º

13 20
1 9
7 1
8 17

XOOOOOOOOOOOOXOOOXX
XOOOOOXOOOOOOOXOOOXX
XOOOOXOXOOOOOOXOOOXX
XOOOXOOOXOOOOOXOOOXX
OOOXOOOOOXXXXXXOOOOO
OOXOOOOOXXXOOOXOOOOX
OXOOOOOXXXXXOOXOOOOX
OOOOOOOOXXXOOOXOOOOX
OOXXXXXXXXXOOOOXOOOOO
XOOOOOXOOOOOOOXOOOOO
XOOOOOXOOOOOOOXOOOXX
XOOOOOXOOOOOOOOOOOXX
XOOOOOOOOOOOOOXOOOXX