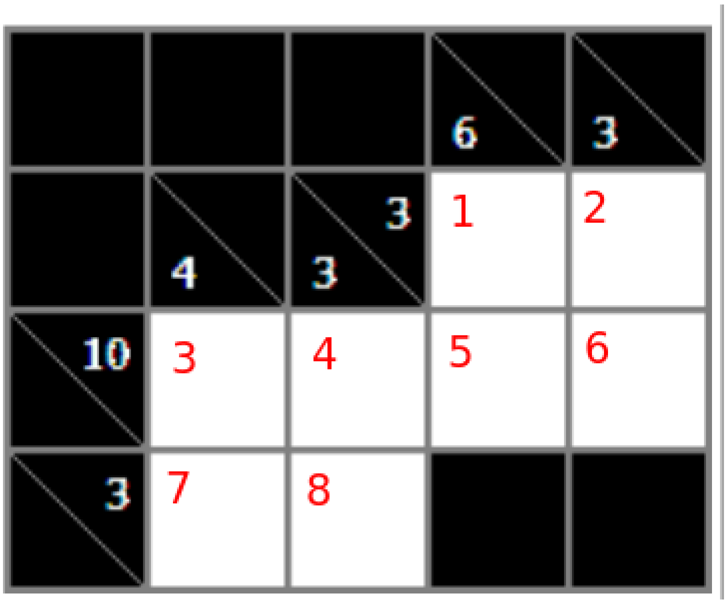
Ονοματεπώνυμο: Ορφανίδης Ελευθέριος
Α.Μ.: 1115201400133

Πρόβλημα 1:

1)Έχουμε 3 είδη κομβων:

-Το μαύρο κουτι στο οποίο δεν μπορούμε να γράψουμε τίποτα.
-Το λευκό κουτί το οποίο μπορεί να πάρει τιμές απο το 1 μέχρι το 9.
-Το μαύρο κουτί περιορισμού, το οποιο περιοριζει είτε τα λευκα κουτια που βρισκονται δεξια απο αυτό(περιορισμος γραμμης), ειτε τα λευκα κουτια που βρισκονται κατω απο αυτο(περιορισμος στηλης), ειτε και τα 2.
Σημ: Ο περιορισμός ισχύει μόνο για τα συνεχόμενα λευκά κουτιά.



Μεταβλητές:

X_i με i ∈ [1,8]

X_1 = {1}, X_2 = {2}
X_3 = {3}, X_4 = {4}
X_5 = {5}, X_6 = {6}
X_7 = {7}, X_8 = {8}

Πεδία ορισμού:

D_i = {1,2,3,4,5,6,7,8,9} με i ∈ [1,8]

Περιορισμοί:

- $X_1 + X_2 = 3$
- $X_3 + X_4 + X_5 + X_6 = 10$
- $X_7 + X_8 = 3$
- $X_3 + X_7 = 4$
- $X_4 + X_8 = 3$
- $X_1 + X_5 = 6$
- $X_2 + X_6 = 3$

Επίσης κάθε σετ μεταβλητών που υπόκειται σε αθροιστικό περιορισμό πρέπει να έχει διαφορετικές τιμές. Δηλ:

- X_1 , X_2 all different
- X_3 , X_4 , X_5 , X_6 all different
- X_7 , X_8 all different
- X_3 , X_7 all different
- X_4 , X_8 all different
- X_1 , X_5 all different
- X_2 , X_6 all different

2) Στο αρχείο kakuro.py έχω μοντελοποιήσει την κλαση kakuro η οποια δέχεται ενα grid ως εισοδο και επεξεργαζεται τα δεδομενα του grid με την μορφη που ζηταει η κλαση CSP. Συγκεκριμενα:
- Καθε σημειο (x,y) του grid που έχει την τιμη '_' ειναι μια variable και οι συντεταγμενες του σημειου γινονται append στην λιστα self.variables.
- Καθε σημειο (x,y) του grid που εχει την τιμη '*' αγνοειται γιατι ειναι το μαυρο κουτι που ανεφερα πιο πάνω.
- Τα υπολοιπα σημεια του grid αναφερονται σε κομβους περιορισμου, οποτε ελέγχω αν εχουν περιορισμο γραμμης , στήλης ή και τους 2 και στην συνέχεια για καθε group συνεχόμενων λευκων κελιων ενημερώνω τα λεξικά neighbors (που χρησιμοποιει η κλαση csp), down_constraint και right_constraint (που χρησιμοποιω στην constraint function που δημιουργησα). <u>Παραδειγμα:</u> Στο grid της εικονας παραπανω, για την δεξιοτερη στηλη η μορφη του down_constraint θα ηταν: down_constraint[(1,4)] = {(2,4)} και down_constraint[(2,4)] = {(1,4)}. Όμως το neighbors[(1,4)] = {(2,4), (1,3)} καθώς περιέχει όλους του γειτονες του (1,4) που βρισκονται σε καποιον περιορισμο μαζι. Το right_constraint λειτουργει με τον ιδιο τροπο που λειτουργει και το down_constraint απλα για τους περιορισμους γραμμης.
Τα domains όλων των μεταβλητών αρχικοποιούνται σε [1,2,3,4,5,6,7,8,9].

<u>Constraint_Function:</u>

<u>Περιορισμός του domain A,B</u>

Η πρώτη constraint Function που δημιουργησα δεχεται 2 μεταβλητες και 1 τιμη για καθε μεταβλητη. Αν η μεταβλητη Α βρισκεται στο down_constraint τοτε με ενα while βρισκω τον κομβο περιορισμού και κρατάω στην μεταβλητη Sum το αθροισμα που πρεπει να εχει αυτο το group μεταβλητων. Στην συνέχεια διατρέχω τις μεταβλητές που βρισκονται στο

down_constraint[A] (δηλαδη τις μεταβλητες που ανηκουν στον ιδιο καθετο περιορισμο).
Ενώ διατρέχω την στήλη:
-Ελέγχω αν καποια απο αυτές τις μεταβλητές εχει γίνει ήδη assign από τον αλγόριθμο:

- Αν εχει γινει assign, αφαιρώ την τιμή που έχει κάνει assign ο αλγοριθμος απο τις πιθανές τιμές που μπορεί να παρει η μεταβλητη A και μειώνω το Sum (το αθροισμα που πρεπει να εχει η στηλη) κατα την τιμη αυτή. (Έχω δηλαδή μικρότερο αθροισμα με λιγότερες μεταβλητές)

-Αν η μεταβλητη δεν εχει γινει assign την προσθέτω 1 στον αριθμο των μεταβλητών που υποκεινται σε αυτόν τον περιορισμό.

Μόλις βρω το συνολο των μεταβλητων του συγκεκριμενου περιορισμου υπολογίζω τις MaxValue και MinValue που μπορει να παρει η μεταβλητη A και αφαιρω απο τις πιθανες τιμες της οποια τιμη δεν ανηκει στο διαστημα [MinValue, Maxvalue].
Εκτελώ την ίδια διαδικασία για το right_constraint και έτσι περιορίζω πληρως τις τιμές του A. Στην συνέχεια εκτελώ το ίδιο για το B.
Αν οι value_a βρισκεται στις διαθέσιμες τιμές του A και value_b στις διαθέσιμες τιμές του B και value_a != value_b τοτε επιστρέφει True η συνάρτηση.
<u>Περί MaxValue και MinValue:</u>

Έστω οτι εχουμε n μεταβλητες που υπόκεινται στον ιδιο περιορισμο και οτι η τιμη του περιορισμου ειναι Sum. Τότε, επιλέγωντας οποιαδήποτε τιμή για τον n-οστο κόμβο , η μικρότερη δυνατή τιμη που μπορουν να εχουν οι υπόλοιποι κομβοι είναι: 1+2+..+(n-1).

$$\text{Sum} = \sum_{k=1}^{n-1} k = n*(n-1)/2$$

Άρα η MaxValue του συγκεκριμενου κομβου ειναι = Sum – n*(n-1)/2

Με τον ιδιο τροπο βγαινει η MinValue καθώς αν επιλέξω οποιαδήποτε τιμη για τον n-οστο κόμβο η μέγιστη τιμη που μπορουν να εχουν οι υπόλοιποι κομβοι ειναι 9+8+...+(n-1).

$$\text{Sum} = \sum_{k=1}^{n-1} (10-k) = (20 - n)*(n - 1) / 2$$

Άρα MinValue = Sum - (20 – n)*(n – 1) / 2

Στο προγραμμα, δινεται η επιλογη αναμεσα σε 4 puzzle κλιμακουμενης δυσκολίας, τα puzzle0, puzzle1 ειναι δυσκολίας 0 , το puzzle 2 ειναι δυσκολίας 1 και το puzzle3 ειναι δυσκολίας 2.

<u>Αλγόριθμοι:</u>
Τα puzzle του kakuro τα λύνω με την χρήση 5 διαφορετικών αλγορίθμων:

- Backtracking
- Forward-Checking
- FC-MRV
- Mac

- Minimum-conflicts

Το backtracking ειναι ο βασικός αλγοριθμός λύσης csp προβλημάτων (και ο πιο αναποτελεσματικός) , ενώ το forward-checking, το FC-MRV και το Mac ειναι βελτιώσεις του Backtracking αλγορίθμου. Η επιλογή αυτων των 4 αλγορίθμων έγινε ώστε να ελεγξω αν υπαρχει βελτιωση απο τον απλο backtracking αλγοριθμο , και αν αυτή ειναι αισθητή.

3)Για έναν πιο ολοκληρωμένο έλεγχο του κάθε αλγόριθμου , στις μετρήσεις μου λαμβάνω υπόψιν , τόσο τον χρόνο εκτέλεσης, όσο και τον αριθμό assignments που γίνονται απο τον αλγόριθμο , καθώς και το πόσες φορές καλείται η constraint function απο τον εκάστοτε αλγόριθμο.
Μεσα στην κλαση kakuro εχω βαλει μια μεταβλητη self.function_calls την οποια αρχικοποιω σε 0 και την κανω +1 καθε φορα που καλειται η constraint_function. Στους παρακατω πίνακες γραφω τον αριθμο που εχει κληθει η συναρτηση απο κάθε αλγόριθμο και σε παρένθεση τον χρόνο εκτέλεσης. Με κόκκινο είναι ο αριθμός των assignments (csp.nassigns) που εγιναν απο τον αλγόριθμο στην εκάστοτε εκτέλεση.
(Σημ: για τους FC-MRV , Mac και Min-con οι αριθμοι αποτελουν έναν μέσο ορο απο πολλαπλες εκτελέσεις)

	Backtracking	Forward-Checking	FC- MRV	Mac	Min-con
Puzzle1	108 8	132 8	182 8	413 8	419 15
Puzzle2	45.853 (1 s) 911	3.027 (0.06 s) 167	1.863(0.035 s) 45	6.271 (0.125 s) 79	15.066 (0.27 s)
Puzzle3	181.537 (5.13 s) 9.269	8.734 (0.26 s) 600	3.727 (0.13 s) 177	19.550 (0.5 s) 193	50.848 (1.25 s) 339

‘Οπως παρατηρουμε στο puzzle3, το οποιο ειναι το puzzle μεγαλυτερης δυσκολιας, ο πιο αποδοτικός αλγόριθμος ειναι ο FC-MRV με μεγάλη διαφορά απο τους υπολοιπους (εξαιρουμένου του FC ο οποιος έχει τις διπλάσιες κλήσεις της constraint function, που αποτελει την δευτερη καλύτερη επίδοση). Ο αλγοριθμος BT, στο puzzle3 κυμαίνεται σε απαγορευτικά επίπεδα καλώντας την constraint function 181.537 φορές.
Επίσης παρατηρουμε ότι στο puzzle1 που ειναι ενα grid πολυ μικρης δυσκολιας, ο backtracking ειναι ο πιο αποδοτικός αλγόριθμος. Μόλις αυξηθει η δυσκολια του grid ομως, τότε γινονται εμφανη τα μειονεκτηματα του backtracking καθώς οι χρονοι εκτελεσης του ειναι στην καλύτερη περίπτωση 6 φορές μεγαλύτεροι απο τον 2ο πιο αργο αλγόριθμο. Ο αλγόριθμος forward-checking ειναι σταθερά πολύ πιο αποδοτικός απο τους υπόλοιπους , ειδικά οταν γίνεται και χρήση του mrv ταυτόχρονα ώστε να επιλέγονται οι μεταβλητές με

τις λιγότερες διαθέσιμες τιμές κάθε φορά.

<u>Πρόβλημα 2</u>

1)

<u>Χρονική Πληροφορία</u>

<u>Αποστάσεις-Χρόνος:</u>

1. Αίθουσα συνεδρίου – Δωμάτιο : 5-10 λεπτά
2. Αίθουσα συνεδρίου – Χρηματοκιβώτιο: 20-30 λεπτά
3. Παραβίαση χρηματοκιβώτιου: 45-90 λεπτά

<u>Χρόνος</u> 9:00 9:30 10:00 10:30 11:00

<u>Άτομα</u>				
Γιάννης (X_1)				
Μαρία (X_2)				
Όλγα (X_3)				
Μήτσος				

Τα κόκκινα κελια του παραπάνω πίνακα δείχνουν ότι για εκείνο το χρονικό διάστημα το συγκεκριμένο άτομο βρισκόταν μέσα στην αίθουσα.

<u>Μεταβλητές:</u> X_{11} , X_{12} , X_{13} , X_{21} , X_{22} , X_{23} , X_{31} , X_{32} , X_{33}

<u>Περιορισμοί:</u>

- $X_{11} + X_{12} + X_{13} \leq 90$ λεπτών
- $X_{21} + X_{22} + X_{23} \leq 60$ λεπτών
- $X_{31} + X_{32} + X_{33} \leq 30$ λεπτών

<u>Πεδία ορισμού</u>:

- $D_{ij} = \{1,2,3\}\ \forall\ i , j \in [1,3]$, όπου 1,2,3 οι <u>αποστάσεις-χρόνος</u> απο πάνω

2) Από τις 11:00, που επέστρεψαν όλοι στην αίθουσα χρείαστηκαν τουλάχιστον 85 λεπτά για την διάρρηξη του χρηματοκιβώτιου.(20 λεπτα για να πάει στο χρηματοκιβώτιο – 20 λεπτά να γυρίσει και 45 λεπτά για να το παραβιάσει). Άρα το αργότερο που θα μπορούσε να φύγει κάποιος από την αίθουσα θα ήταν στις 9:35. Η Μαρία εκείνη την στιγμή παρουσίαζε το βιβλίο της ενώ η Όλγα δεν το είχε παρουσιάσει ακόμα (η παρουσίαση της ξεκίναγε στις 10:00) . Οπότε ο μοναδικός που πληρεί τις προϋποθέσεις είναι ο Γιάννης.

Ο αστυνόμος κατάλαβε ότι για να διαρρήξει κάποιος, που βρίσκεται στην αίθουσα, το χρηματοκιβώτιο χρειάζεται τουλάχιστον 85 λεπτά συνεχόμενης απουσίας από την αίθουσα, άρα πρέπει:

- $X_{i1} + X_{i2} + X_{i3} \geq 85$ λεπτών , για κάποιο $i \in [1,3]$

Η μόνη μεταβλητή που ικανοποιεί την από πάνω συνθήκη είναι η X_1 δηλαδή ο Γιάννης.

3) Ο αστυνόμος μπορεί να αναθέτει τυχαία τιμές στις μεταβλητές, να ελέγχει σε κάθε αναθεση αν τηρούνται οι περιορισμοί και μετά να ελέγχει αν αυτές οι τιμές καλύπτουν τον στόχο που αναφέρεται απο πάνω, δηλαδη να έχουν άθροισμα ≥ 85.

<u>Πρόβλημα 5</u>

Logic operators: $\wedge$ $\vee$, $\Rightarrow$, $\Leftrightarrow$

1. Έστω:
   P : Θα πας στο γήπεδο
   Q : Θα βρέχει
   Z : Θα έρθω μαζί σου

Τότε:

$$(P \wedge \neg Q) \Leftrightarrow T$$

2. Έστω:
   P : ο καιρος θα ειναι κακός
   Q: θα είμαι άρρωστος
   Z: θα έρθω σχολείο

Τότε:

$$P \vee Q \Leftrightarrow \neg Z$$

3. Έστω:
   P: Θα έρθει η Μαρία στο πάρτυ
   Q: Θα έρθει η Ελένη στο πάρτυ

Τότε:

$$Q \Rightarrow (P \vee \neg P)$$

4. Έστω:
   P: Θα βελτιώσεις τις γνώσεις σου στον προγραμματισμό
   Q: Θα αρχίσεις να διαβάζεις παραπάνω
   Z: Θα πάρεις πτυχίο

Τότε:

$\neg(P \wedge Q) \Leftrightarrow \neg Z$

5.Έστω:

P: Υπάρχουν εξωγήινοι
Q: Βρίσκονται εξωγήινοι στην Γη
Z: Η Γη δεν είναι ενδιαφέρον τουριστικός προορισμός

Τότε:

$P \Leftrightarrow(Q \vee \neg Z)$

Πρόβλημα 4

α)

$(A \wedge B \wedge C \Rightarrow D) \Leftrightarrow (A \Rightarrow (B \Rightarrow (C \Rightarrow D)))$

$(\neg(A \wedge B \wedge C) \vee D) \Leftrightarrow(\neg A \vee \neg B \vee \neg C \vee D)$

$(\neg A \vee \neg B \vee \neg C \vee D) \Leftrightarrow(\neg A \vee \neg B \vee \neg C \vee D)$

Θέτω $X = (\neg A \vee \neg B \vee \neg C \vee D)$ άρα καταλήγω στο:

$X \Leftrightarrow X$

X	$X \Leftrightarrow X$
F	T
T	T

Η παραπάνω σχέση είναι πάντα αληθής, όπως βλέπουμε και στον πίνακα αλήθειας, οπότε είναι ταυτολογία άρα είναι και έγκυρη και ικανοποιήσιμη, από την στιγμή που έχει τουλάχιστον ένα μοντέλο. Επίσης είναι και σε μορφή Horn.

β)

$A \wedge (A \Rightarrow B) \wedge (A \Rightarrow \neg B)$

$A \wedge (\neg A \vee B) \wedge (\neg A \vee \neg B)$

$(A \wedge \neg A) \vee (A \vee B) \wedge (\neg B \vee \neg A)$

$(A \wedge \neg A)$:False

$(A \vee B) \wedge (\neg B \vee \neg A)$: False

Άρα καταλήγουμε στο False OR False = False. Οπότε η συγκεκριμένη πρόταση δεν ειναι έγκυρη, ούτε ικανοποιήσιμη, ούτε ταυτολογία. Επίσης δεν έχει κανένα μοντέλο και δεν είναι πρόταση σε μορφή Horn.

γ)

$(A \vee B) \wedge (\neg A \vee C) \wedge \neg B \wedge \neg C$

$(A \vee B) \wedge (\neg A \wedge \neg B) \vee (\neg B \wedge C) \wedge \neg C$

$(A \vee B) \wedge (\neg A \wedge \neg B)$: False

$(\neg B \wedge C) \wedge \neg C$: False

Άρα καταλήγουμε στο False OR False = False. Οπότε και αυτή η πρόταση δεν ειναι έγκυρη, ούτε ικανοποιήσιμη, ούτε ταυτολογία. Επίσης δεν έχει κανένα μοντέλο και δεν είναι πρόταση σε μορφή Horn.

δ)

$(A \vee B) \wedge (\neg A \vee C) \wedge (B \vee C)$

A	**B**	**C**	**Πρόταση**
F	F	F	F
F	F	T	F
F	T	F	T
F	T	T	T
T	F	F	F
T	F	T	T
T	T	F	F
T	T	T	T

Από ότι παρατηρουμε από τον πίνακα αλήθειας η συγκριμένη πρόταση έχει τουλάχιστον ένα μοντέλο , είναι ικανοποιήσιμη και έγκυρη, αλλά δεν είναι ταυτολογία.

Πρόβλημα 7

1. Είναι καλά ορισμένη πρόταση
2. Δεν ειναι καλά ορισμένη πρόταση. (Στις διαφάνειες χρησιμοποιούμε αυτό $\Rightarrow$)
3. Είναι καλά ορισμένη πρόταση
4. Δεν είναι καλά ορισμένη.
5. Δεν είναι καλά ορισμένη πρόταση.

Πρόβλημα 6

Πρόταση 1: $A \wedge (B \Leftrightarrow C)$

CNF πρότασης 1: $A \wedge (\neg B \vee C) \wedge (B \vee \neg C)$ (1)

A (2)

$(\neg B \vee C)$ (3)

$(B \vee \neg C)$ (4)

Πρόταση 2: $(A \wedge B) \Leftrightarrow (A \wedge C)$

Αρκεί η άρνηση αυτής της πρότασης να καταλήξει σε κενό μέσω της ανάλυσης.

CNF της άρνησης: $A \wedge (\neg B \vee \neg C) \wedge (B \vee C)$ (5)

A (6)

$(\neg B \lor \neg C)$ (7)

$(B \lor C)$ (8)

<u>Ανάλυση:</u>

Από τις (8), (3): C (10)

Από τις (7), (4) : $\neg C$ (11)

Από τις (10),(11): κενό

Άρα η πρόταση 1 καλύπτει λογικά την πρόταση 2-