Απαντήσεις θεωρητικών ερωτήσεων τρίτης εργασίας Τεχνητής Νοημοσύνης

Άγγελος Τσιτσόλη sdi2000200

Νοέμβριος 13, 2022

Πρόβλημα 1

Η απάντηση στο πρώτο υποερώτημα βρίσκεται στο σημείο αρχείο csp.py προς το τέλος του pdf . **Kenken:** Αρχικά δίνεται απο το πρώτο ερώτημα ότι θα πρέπει να μοντελοποιήσουμε το παζλ κενκεν ως πρόβλημα ικανοποίησης περιορισμών . Οπότε για την επίλυση του προβλήματος αυτού θα πρέπει να χρησιμοποιήσουμε αλγορίμθους αναζήτησης οι οποίοι θα ικανοποιούν όσο καλύτερα γίνετε τους περιορισμούς που διαθέτει το πρόβλημα ώστε να βρούμε την λύση του (όπου λύση του προβλήματος θα θεωρηθεί η ανάθεση τιμής σε κάθε μεταβλητή του προβλήματος).Για την λύση του προβλήματος θα μπορούσαμε να χρησιμοποιήσουμε αλγορίθμους όπως ο BFS ή ο DFS , ωστόσο παρουσιάζουν προβλήματα και οι δύο , επομένως θα κοιτάξουμε αλγορίθμους υπαναχώρησης. Στο αρχείο csp.py υπάρχουν ορισμένοι αλγόριθμοι υπαναχώρησης , διάλεξα δύο αλγορίθμους απο τους αλγορίθμους υπαναχώρησης (FC,MAC) τους οποίους θεώρησα ως κατάλληλους σε σχέση με άλλους όπου εξηγώ παρακάτω.

Forward Checking:
Ουσιαστικά χρησιμοποιείται ο αλγόριθμος αυτός ως μια καλύτερη έκδοση του αλγοριθμου backtrack search. Ο αλγόριθμος backtrack αυτό που θα κάνει είναι να αναθέτει τιμές στις μεταβλητές μέχρι να μην μπορεί να αναθέτει τιμές λόγω περιορισμών οπότε και θα γυρίσει πίσω Επίσης ο backtrack θα κάτσει και θα λειτουργήσει με το σκεπτικό να ελέγξει όλες τις τιμές για κάθε μεταβλητή μέχρι να καταλήξει σε κάποια τιμή για αυτές , προφανώς πέφτωντας πάνω σε παραβιάσεις περιορισμών συνεχών καθώς μόνο έτσι θα ξέρει αν θα βάλει ή όχι κάποια τιμή σε μεταβλητή. Προφανώς θα χρειαστούμε κάτι πιο γρήγορο και πιο έξπνο. Μέσω του forward checking μπορούμε να βελτιώσουμε τον backtrack μέσω της διαγραφής των τιμών . Δηλαδή κάθε φορά που βάζουμε μια τιμή σε μιά μεταβλητή τότε διαγράφουμε για τις επόμενες μεταβλητές τιμές που θα παραβιάσουν τους περιορισμούς μας , 'προνοεί' ουσιαστικά τις παραβιάσεις περιορισμών ώστε να μην πέσει πάνω τους.

Make Arc Consistency:
Ο αλγόριθμος αυτός με λίγα λόγια λέει ότι για κάθε τιμή μιας μεταβλητής X του προβλήματος υπάρχει μια τιμή για την μεταβλητή Y για την οποία δεν παραβιάζεται κάποιος περιορισμός. Δηλαδή δίνεται κάποια τιμή σε μια μεταβλητή αλλά σε περίπτωση που η μεταβλητή αυτή δεν είναι συνεπής με κάποια άλλη μεταβλητή τότε μπορεί εύκολα να γίνει αφαιρώντας ορισμένες τιμές που παραβιάζουν περιορισμούς οι οποίες ήταν υποψήφιες να πάρει η μεταβλητή (σύμφωνα πάντα με τον παραπάνω κανόνα που αναφέρθηκε), και αυτό μπορεί να γίνει αναδρομικά.Προφανώς θα είναι καλύτερος ο αλγόριθμος απο τον αλγόριθμο backtrack search σκέτο, καθώς προβλέπει απο νωρίς τυχόν παραβιάσεις των περιορισμών.

Σχέση μεταξύ forward checking και make arc consistency:
Ο αλγόριθμος forward checking μπορεί να προσφέρει καλύτερο χρόνο απο τον αλγόριθμο backtrack search σκέτο , ωστόσο σε σχέση με τον make arc consistency ο αλγόριθμος forward checking υστερεί στο γεγονός ότι δεν μπορεί να 'προλάβει' απο νωρίς κάθε παραβίαση . Δηλαδή απο την μία κάνει αναθέσεις και δίνει σε επόμενες την πληροφορίες ως προς το ποιες τιμές να αποφύγουν παρόλ'αυτα απο την άλλη μπορεί να μην ανιχνεύσει έγκαιρα κάποιο λάθος που μπορεί'να γίνει και να βρεθούμε σε αδιέξοδο.Με λίγα λόγια κάθε φορά που χρησιμοποιούμε forward checking όταν γίνεται μια ανάθεση μπορούμε να βεβαιωθούμε ότι δεν θα γίνει παραβίαση των περιορισμών μόνο για την ακριβώς επόμενη ανάθεση που θα γίνει και όχι για τις υπόλοιπες επόμενες καθώς δεν φτάνει τόσο μακρυά , κάθε φορά δηλαδή μεριμνά μόνο για την επόμενη μεταβλητή . Μέσω της make arc consistency ωστόσο εφόσον αλάζουμε τις τιμές

που μπορούν να δωθούν σε μια μεταβλητή τότε θα πρέπει να αλλάξουμε και τις τιμές των 'γειτόνων' μεταβλητών ή αλλιώς των μεταβλητων με τις οποίες συνδέται δημιουργεί μια ακμή. Άρα μαυτόν τον τρόπο γίνετε πιο εύκολα ανίχνευση κάποιο λάθος και νωρίτερα σε σχέση με τον forward checking.Είναι προφανές ότι ο MAC χρειάζεται περισσότερο χρόνο για να κάνει αυτή την δουλειά διότι κάνει μια ανάθεση σε μια μεταβλητή και ύστερα μεριμνά και για τις υπόλοιπες μεταβλητές ώστε να προλάβει παραβιάσεις . Απο την άλλη γλυτώνονται έτσι οι πολλές αναθέσεις διότι προλαβαίνει τυχόν παραβιάσεις άρα και δεν γίνονται πολλές δοκιμές σε μια μεταβλητή , δηλαδή απο εκεί που ελέγχονταν για παράδειγμα 4 τιμές σε μια μεταβλητή με την μία να είναι σωστή και τις τρεις λάθος με τον MAC θα είχε ανατεθεί μόνο η μία η σωστή , στην μετβαλητή .Οπότε ο MAC προνοεί καλύτερα απο τον FC ωστόσο χρειάζεται λίγο περισσότερο χρόνο κάνοντας ελέγχους και διατηρώντας την συνέπεια σε όλες τις μεταβλητές έτσι ώστε να γλυτώνει παραβιάσεις.

MinConflicts:

Ο αλγόριθμος αυτός ξεκινάει με τυχαία ανάθεση τιμών στις μεταβλητές και σιγα σιγά στην συνέχεια διορθώνει τις αναθέσεις που έκανε .Μετά την αρχική τυχαία ανάθεση ουσιαστικα θα διαλέξει τυχαία μία μεταβλητή , απο τις μεταβλητές που παραβιάζουν έναν ή πολλούς περιορισμούς με την τιμή που διαθέτουν και στην συνέχεια θα της δώσει την τιμή που προκαλέι τις λιγότερες παραβιάσεις περιορισμών .Σε περίπτωση που υπάρχουν παραπάνω απο μία τιμές οι οποίες προκαλούν τους λιγότερο δυνατούς περιορισμούς τότε επιλέγεται τυχαία μία απο αυτές για την μεταβλητή.

Χρήση της ευρετικής MRV:

Με την χρήση της ευρετικής MRV αποσκοπούμε να βελτιώσουμε έναα αλγόριθμο υπαναχώρησης. Συγκεκριμένα ο MRV θα επιλέξει τις μεταβλητές κάθε φορά με λιγότερες νόμιμες τιμές που απομένουν , ελαχιστοποιώντας τον παράγοντα διακλάδωσης . Έλεγξα τα αποτελέσματα της ευρετικής αυτής σε συνδυασμό με τους αλγορίθμους υπαναχώρησης που διάλεξα και έβγαλα κάποια αποτελέσματα παρακάτω και συμπεράσματα σύμφωνα με τις μετρήσεις.

	Backtrack Search with Forward Checking			Backtrack Search with Maintaining Arc Consistency		
	Time	nassigns	Number of constraint calls	Time	nassigns	Number of constraint calls
3x3	0.0007173	9	95	0.0018496	9	176
4x4	0.0015551	41	566	0.007867899999999999	52	1771
5x5	0.0444042	1160	16632	0.1125046	509	20475
6x6	0.029188	769	16645	0.2104298	510	53127
7x7	0.0843727	1360	39398	0.34538	650	83886
8x8	42.244427599999995	830457	21145437	51.9803552	108445	11988468

MinConflicts			
	Time	nassigns	Number of constraint calls
3x3	9.4181504	100018	3546
4x4	16.9258936	100068	12000192
5x5	54.2983422	100534	24000500
6x6	64.12298969999999	100036	42001080
7x7	112.56800750000001	100049	
8x8	167.47309539999998	100064	100803584

	Backtrack Search with Forward Checking and MRV			Backtrack Search with Maintaining Arc Consistency and MRV		
	Time	nassigns	Number of constraint calls	Time	nassigns	Number of constraint calls
3x3	0.0016570000000000196	10	108	0.004293999999999992	25	400
4x4	0.0020408999999999844	41	660	0.019730200000000003	142	5005
5x5	0.008258600000000005	192	2917	0.0820421	302	16073
6x6	0.0062570000000000126	76	2376	0.05397259999999998	126	13318
7x7	16.8136674	1360	8095961	0.07564699999999999	144	22756
8x8	14.9975511	6705807	191602	174.1870489	354547	34925993

Μετρήσεις:

Οι μετρήσεις που έκανα για τους αλγόριθμους έγιναν με βάση τον χρόνο εκτέλεσης τους , τον αριθμό των αναθέσεων και τον αριθμό τον φορών που καλέστηκε η συνάρτηση constraints για κάθε περίπτωση. Φαίνεται ξεκάθαρα από τις μετρήσεις, ότι ο MAC συνολικά κάνει λιγότερες αναθέσεις όπως είναι και φυσικό καθώς όπως είπα προνοεί τις παραβιάσεις καλύτερα απο τον FC .Επίσης φαίνεται ότι συνολικά ο χρόνος που κάνει ο MAC είναι περισσότερος σε σχέση με τον FC για το λόγο παραπάνω που προανέφερα.Τέλος γίνονται περισσότερες κλήσεις της συνάρτησης constraint στον MAC καθώς για κάθε μεταβλητή ελέγχεται συνεχώς αν είναι συνεπής με τις άλλες οπότε ουσιαστικά ελέγχεται συνεχώς η συνέπεια όλων των

μεταβλητών μέχρι να βρεθεί η λύση για το πρόβλημα.Για τον FC συνολικά θα γίνουν περισσότερες κλήσεις της συνάρτησης παραβίασεις περιορισμών κάθως θα ελέγχθούν περισσότερες μεταβλητές λόγω του γεγονότος ότι δεν θαμ πορέσει να προνοίσει τόσο καλά ο αλγόριθμος σε σχέση με τον MAC , επίσης ο fc θα κάνει περισσότερες αναθέσεις ακριβώς επειδή θα κάνει λάθη μέχρι να βρεί την σωστή τιμή, αλλά ο χρόνος που θα κάνει σε σχέση με τον mac θα είναι μικρότερος καθώς ένας λόγος είναι ότι η προδιεργασία για την σωστή ανάθεση που θα κάνει αυτός ο αλγόριθμος δεν θα είναι τόσο χρονοβόρα σε σχέση με τον mac , ουσιαστικά θα κάνει τις αναθέσεις του με βάση κάθε φορά των πληροφοριών που του δίνει η προηγούμενη ανάθεση και θα το συνεχίσερι μέχρι να βρεί το σωστό , δεν θα κάνει δηλαδή μια χρονοβόρα διεργασία να ελέγξει τις συνέπειες των μεταβλητών όπως ο MAC (προφανώς ο FC έχει μεγαλύτερη πιθανότητα να κάνει λάθος αλλά σίγουρα θα είναι πιο γρήγορος ανεξαρτήτως αποτελέσματος σε σχέση με τον MAC).

Παρατηρούμε απο τις μετρήσεις ότι ο min conflicts κάνει και περισσότερο χρόνο και περισσότερες αναθέσεις αλλά και περισσότερες κλήσεις της συνάρτησης περιορισμών σε σχέση με τις άλλες συναρτήσεις. Αρχικά η συνάρτηση για τον έλεγχο παραβίασης περιορισμών θα κλήθεί πάρα πολλές φορές καθώς ο αλγόριθμος ουσιαστικά αυτό κάνει ελέγχει κάθε φορά τις παραβιάσεις που θα προκληθούν θα τις μετρήσει και ύστερα θα επιδιορθώσει τα λάθη του , οπότε θα συγκρίνει συνεχώς όλες τις τιμές μεταξύ τους για κάθε μεταβλητή. Προφανώς όσον αφορά τις αναθέσεις θα κάνει πάρα πολλές αναθέσεις μέχρι να βρεί την σωστή δηλαδή θα δοκιμάζει λάθος τιμές συνεχώς μέχρι κάποια στιγμή να βρεθεί στην σωστή.Οπότε είναι φυσικό αυτή η διαδικασία να παίρνει πολύ χρόνο , καθώς θα πρέπει να γίνει πάρα πολλοί έλεγχοι και πάρα πολλές αναθέσεις .

Με την χρήση του MRV για τους αλγορίθμους παρατηρούμε ότι σύμφωνα με τα αποτελέσματα η χρήση της ευρετικής αυτής σε ορισμένες περιπτώσεις βελτιώνει τον αλγόριθμο σε άλλες περιπτώσεις τους χειροτερεύει.Δηλαδή :

MRV + FC σε σύγκριση με σκέτο FC:
Παρατηρούμε στο πρώτο ,δεύτερο παράδειγμα τα αποτελέσματα είναι χειρότερα στον MRV με FC απο ότι στον σκέτο και οι τρείς μετρηκές είναι έχουν αποτελέσματα χειρότερα.Το τρίτο παράδειγμα είναι κατά πολύ καλύτερο ως προς τις τρείς μετρηκές , έχει μειωμένο αρκετά και χρόνο και ανθέσεις και καλέσματα της constraint function. Το τέταρτο παράδειγμα είναι κατά πολύ χειρότερος ο αλγόριθμος όπως τα πρώτα δύο παραδείγματα δηλαδή οι μετρηκές είναι κατά πολύ μεγαλύτερες .Τέλος στα δύο τελευταία είναι πάλι πολύ βελτιωμένος και ως προς τις τρείς μετρηκές .Να σημειωθεί μάλιστα ότι ο MRV χρησιμοποιείται συνήθως με τον FC καθώς οι απαιτούμενες δομές για την υλοποίηση του MRV χρησιμοποιούνται απο τον FC. Προφανώς ο MRV θεωρητικά είναι χρήσιμος καθώς σαν ευρετική θα δώσει μία έξτρα πληροφορία για τον αλγόριθμο ώστε να κινηθεί προς την σωστή κατεύθυνση για να πετύχει την κατάσταση στόχου, αλλά στην πράξη βλεπουμε ότι δεν ισχύει πάντα αυτό.

MRV + MAC σε σύγκριση με σκέτο MAC:
Ομοίως παρατηρούμε και στον MAC ότι σε κάποιες περιπτώσεις είναι καλή η ευρετική και σε άλλες όχι σε σχέση με τον αλγόριθμο σκέτο .Δηλαδή στο πρώτο και στο δεύτερο παράδειγμα είναι κακή η χρήση της ευρετικής δηλαδή δεν βελτιώνει τον αλγόριθμο .Στην τρίτη περίπτωση η χρήση της ευρετικής βελτιώνει πολύ τον αλγόριθμο , ειδικότερα στην τέταρτη και πέμπτη .Στην τελευταία περίπτωση 'καταστρέφει' σε μεγάλο βαθμο τον αλγόριθμο

Γενικότερα: Παρατηρούμε ότι ο MRV σε κάποιες περιπτώσεις βοηθάει τους αλγορίθμους τείνει να βοηθήσει λίγο περισσότερο προς τα πιο δύσκολα παραδείγματα αλλα και εκεί όχι πάντα , όπως ένα παράδειγμα αποτελεί η τελευταία περίπτωση του mac με mrv όπου το αποτέλεσμα είναι χειρότερο απο ότι χωρίς τον mrv.

Πρόβλημα 2

Το πρόβλημα ικανοποίησης περιορισμών μοντελοποιείται ως εξής:

Μεταβλητές:Κρεβάτι,Γραφείο,Καρέκλα γραφείου,Καναπές.
(οι μεταβλητές δέχονται ως τιμές τα εξής Πλάτος , Ύψος , Μήκος , (Βάθος) .Το βάθος σε κάποιες

μεταβλητές.

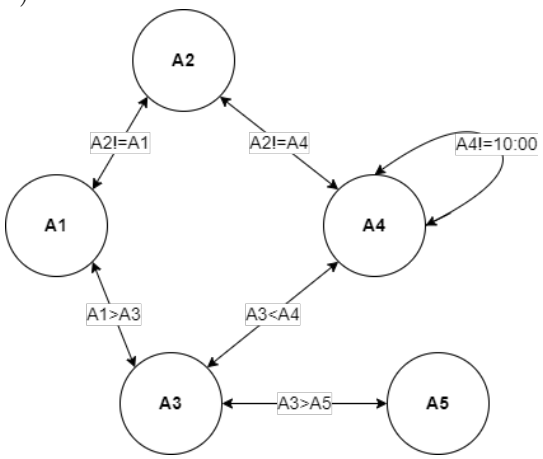
Πεδίο ορισμού:Πλάτος:[1-300cm],Μήκος:[1-400cm],Ύψος:[1-230cm], Βάθος:[1-60cm] (Ουσιαστικά για τιμές παίρνουμε ως μικρότερη σε όλα την τιμή 1 και ως μεγαλύτερη τη μεγαλύτερη για κάθε παράμετρο με βάση τις μετρήσεις του δωματίου, δηλαδή για το πλάτος παίρνουμε ως μεγαλύτερη το πλάτος του δωματίου , την ίδια λογική εφαρμόζουμε στο μήκος και το ύψος εκτός απο το βάθος . Να σημειωθεί ότι προφανώς δεν θα φτάσει κάποια απο τις μεταβλητές πλάτος,μήκος,ύψος την μέγιστη τιμή της απλώς θέτουμε έν όριο. Για το βάθος δώθηκε προσεγγίστικά μια τιμή ως η μεγαλύτερη για το διάστημα τιμών της.

Περιορισμοί:(Οι περιορισμού που μας δώθηκαν δηλαδή:)
1)Τα έπιπλα δεν πρέπει να εφάπτονται ή να πατάνε το ένα πάνω στο άλλο.
2)Το γραφείο θα πρέπει να είναι δίπλα σε κάποια είσοδο φωτός στο δωμάτιο.

Πρόβλημα 3

1)
Μεταβλητές: A1,A2,A3,A4,A5
Πεδίο ορισμού: 9:00,10:00,11:00
Περιορισμοί:A1>A3,A3<A4,A3>A5,(A2≠A1 or A2≠A4),A4≠10 : 00

2)



3)
Έστω ότι έχουμε αρχικά τις παρακάτω τιμές για τα χαρακτηριστικά του προβλήματος:
A1=[9 : 00, 10 : 00, 11 : 00]
A2=[9 : 00, 10 : 00, 11 : 00]
A3=[9 : 00, 10 : 00, 11 : 00]
A4=[9 : 00, 10 : 00, 11 : 00]
A5=[9 : 00, 10 : 00, 11 : 00]
Συνολικά προκύπτουν οι εξής περιορισμοί απο τους ήδη υπάρχοντες περιορισμούς του προβλήματος :
A1>A3,A3<A1,A3<A4,A4>A3,A3>A5,A5<A3,A4≠10 : 00,(A2≠A1 and A1≠A2 or A2≠A4 and A4≠A2).

Με την σειρά ικανοποιούμε τους παραπάνω περιορισμούς και κάθε φορά μεταβάλλουμε τις τιμές που μπορούν να πάρουν οι μεταβλητές .

1. Αρχικά το A1 θα πρέπει να έχει τιμές μεγαλύτερες απο το A3 οπότε διαγράφουμε την τιμή 9:00 απο τις τιμές του A1.
2. Το A3 θα πρέπει να έχει τιμές μικρότερες απο το A1 οπότε διαγράφουμε την τιμή 11:00
3. Το A3 θα πρέπει να έχει μικρότερες τιμές απο το A4 όπου και έχει οπότε παραμένει ίδιο ως προς τις τιμές .
4. Το A4 θα πρέπει να έχει τιμές μεγαλύτερες απο το A3 οπότε διαγράφουμε απο το A4 την τιμή 9:00.

5. Το Α3 θα πρέπει να έχει τιμές μεγαλύτερες απο το Α5 άρα θα διαγραφεί η τιμή 9:00 απο το Α3.

6. Το Α5 θα πρέπει να έχει τιμές μικρότερες απο το Α3 οπότε θα διαγραφούν οι τιμές 10:00 και 11:00 απο το Α5.

7. Το Α4 δεν θα πρέπει να έχει την τιμή 10:00.

8. Σε περίπτωση που το Α2 δεν θα πρέπει να εκτελεστεί την ίδια ώρα με το Α1 τότε αν το Α1 ξεκινάει στις 10:00 τότε το Α2 μπορεί να πάρει τις τιμές 9:00 ή 11:00 . Αν το Α1 ξεκινάει στις 11:00 τότε το Α2 μπορεί να ξεκινήσει στις 9:00 ή στης 10:00.

9. Αν το Α2 δεν θα πρέπει να εκτελεστεί όταν εκτελείται το Α4 τότε εφόσον το Α4 ξεκινάει στης 11:00 τότε το Α2 θα πρέπει να ξεκινήσει στης 9:00 ή στης 10:00.

Προκύπτουν τα εξής :
Α1=[10:00,11:00]
Αν Α1=[10:00] τότε Α2=[9:00,11:00], αν Α1=[11:00] τότε Α2=[9:00,10:00] και στις δύο περιπτώσεις αυτές Α4=[11:00] . Αλλιώς αν το Α2 εξαρτάται απο το Α4 τότε Α4=[11:00] και Α2=[9:00,10:00]
Α3=[10:00]
Α4=[11:00]
Α5=[9:00]

Πρόβλημα 4

1. Έγκυρες ονομάζονται οι προτάσεις (φ) για τις οποίες για κάθε ερμηνεία I ισχύει I(φ)=true.
   Έγκυρη πρόταση δεν υπάρχει.

2. Ικανοποιήσιμες ονομάζονται οι προτάσεις (φ) για τις οποίες υπάρχει μια ερμηνεία I τέτοια ώστε I(φ)=true.
   Ικανοποιήσημες : η πρώτη , η δεύτερη , τρίτη και η τέταρτη

3. Μη ικανοποιήσιμες ονομάζονται οι προτάσεις (φ) για τις οποίες δεν υπάρχει ερμηνεία I τέτοια ώστε I(φ)=true.
   Μη ικανοποίησιμες: καμία

4. Αν I μία ερμηνεία τέτοια ώστε I(φ)=true , τότε αυτό σημαίνει ότι I είναι ένα μοντέλο της φ.
   Τουλάχιστον ένα μοντέλο έχουν η πρώτη, η δεύτερη , τρίτη και η τέταρτη

5. Οι έγκυρες προτάσεις προτασιακής λογικής λέγονται ταυτολογίες.
   Ταυτολογίες δεν υπάρχουν.

Πρόβλημα 5

Έστω η πρόταση Ο Σήφης είναι καλός μάγειρας , την οποία έστω την ονομάζουμε πρόταση Α. Ακόμα έστω η πρόταση εγώ είμαι αστροναύτης την οποία ονομάζουμε πρόταση Β.
Από την εκφώνηση κιόλας γνωρίζουμε με σιγουριά ότι αυτός που λέει την πρόταση "Αν ο Σήφης είναι καλός μάγειρας τότε εγώ είμαι αστροναύτης." ΔΕΝ είναι αστροναύτης. Άρα γνωρίζουμε ότι σίγουρα η πρόταση Β είναι λάθος.Απο την άλλη η πρόταση Α δεν γνωρίζουμε αν είναι λάθος ή σωστή με σιγουριά , δηλαδή ο Σήφης μπορεί να είναι καλός μάγειρας , μπορεί και όχι , άρα δεν μπορούμε να πούμε με σιγουριά ότι η πρόταση Α είναι λάθος ή σωστή. Η πρόταση "Αν ο Σήφης είναι καλός μάγειρας τότε εγώ είμαι αστροναύτης. αποτελεί μια συνεπαγωγή των προτάσεων Α και Β ($A -> B$) .Σε περίπτωση που η πρόταση Α είναι αληθής τότε η συνεπαγωγή είναι ψευδής διότι στην συνεπαγωγή μας η πρόταση Α θα είναι αληθής ενώ το συμπέρασμα μας θα είναι ψευδές άρα συνολικά η πρόταση ψευδής οπότε ο Σήφης δεν είναι καλός μάγειρας. Σε περίπτωση που η υπόθεση μας είναι ψευδής και εφόσον ξέρουμε ότι το συμπέρασμα είναι ψευδές τότε η πρόταση συνολικά θα είναι αληθής , ότι δηλαδή ο Σήφης δεν είναι καλός μάγειρας.

Πρόβλημα 6

(Το ερώτημα αυτό λύθηκε σύμφωνα μετα slides που δίνονται στο φροντιστήριο με όνομα Propositional Logic - An Entailment Proof) για xor χρησιμοποιώ το Χ.

Σημείωση:Παρακάτω για xor χρησιμοποιώ το Χ. Έχουμε την εξής μετατροπή των παρακάτω προτάσεων:

Το σχήμα α είναι κύβος ή δωδεκάεδρο ή τετράεδρο:
Κύβος Χ Δωδεκάεδρο Χ Τετράεδρο.

Το σχήμα α είναι μικρό ή μεσαίο ή μεγάλο:
Μικρό Χ Μεσσαίο Χ Μεγάλο.

Το σχήμα α είναι μεσαίο αν και μόνο αν είναι δωδεκάεδρο.
Μεσσαίο <=> Δωδεκάεδρο.

Το σχήμα α είναι τετράεδρο αν και μόνο αν είναι μεγάλο.
Τετράεδρο <=> Μεγάλο

Το σχήμα α είναι κύβος αν και μόνο αν είναι μικρό.
Κύβος <=> Μικρό.

Θα πρέπει να δειχθεί ότι :
Κύβος Χ Δωδεκάεδρο Χ Τετράεδρο ∧ Μικρό Χ Μεσσαίο Χ Μεγάλο ∧ Μεσσαίο<=> Δωδεκάεδρο ∧ Τετράεδρο <=> Μεγάλο | = (ή αλλιώς έπεται λογικά) Κύβος <=> Μικρό

Έστω ότι παίρνουμε μια ερμηνεία η οποία ικανοποιεί την πρόταση Κύβος Χ Δωδεκάεδρο Χ Τετράεδρο ∧ Μικρό Χ Μεσσαίο Χ Μεγάλο ∧ Μεσσαίο<=> Δωδεκάεδρο ∧ Τετράεδρο <=> Μεγάλο . Αυτό σημαίνει ότι υπάρχει Ι τέτοιο ώστε να ισχύει Ι(η πρόταση παραπάνω)=true.
Άρα θεωρώντας ότι υπάρχει ένα Ι για το οποίο η πρόταση θα είναι True προκύπτει ότι κάθε τμήμα τις πρότασης απο τα τέσερρα (1)Κύβος Χ Δωδεκάεδρο Χ Τετράεδρο , 2) Μικρό Χ Μεσσαίο Χ Μεγάλο , 3) Μεσσαίο<=> Δωδεκάεδρο , 4) Τετράεδρο <=> Μεγάλο) θα πρέπει να είναι true λόγω των όρων ∧ για να βγεί true η πρόταση συνολικά και να ισχύει το Ι(πρόταση)=true.Για να δείξουμε αυτό που θέλουμε θα ασχοληθούμε με το τρίτο και τέταρτο κομμάτι της πρότασης βάζοντας τιμές σαυτά τα κομμάτια (τιμές True,False) συγκεκριμένα θα ξεκινάμε βάζοντας τις τιμές αυτές στο τρίτο κομμάτι της πρότασης , αλλά κάλιστα θα μπορούσαμε να δίνουμε τις τιμές αυτές και στο τέταρτο κομμάτι της πρότασης (θα φανεί στην συνέχεια η υλοποίηση) .Οπότε έχουμε τις εξής περιπτώσεις :

Έστω ότι βάζουμε την τιμή True στο Μεσσαίο(θα μπορούσαμε να κάνουμε το ίδιο και στο Μεγάλο)
Εφόσον έχει την τιμή True τότε αναγκαστικά θα πρέπει και το Δωδεκάεδρο να έχει την τιμή True . Οπότε παρατηρούμε ότι στα τμήματα ένα και δύο True παίρνει μόνο μία τιμή για να βγεί εν τέλει true η πρόταση και στο ένα και δύο είναι οι Δωδεκάεδρο και Μεσσαίο τότε στα τμήματα ένα και οι δύο οι υπόλοιπες μεταβλητές θα πρέπει να είναι False για να βγούν True τα τμήματα .Ομοίως και το τελευταίο τμήμα θα έχουμε False< − >False άρα True και συνολικά True η πρόταση.Άρα στην περίπτωση αυτή ο Κύβος και η μεταβλητή Μικρό έχουν ίδιες τιμές.

Έστω ότι βάζουμε την τιμή False στο Μεσσαίο(θα μπορούσαμε να κάνουμε το ίδιο και στο Μεγάλο)
Αυτό σημαίνει ότι και η μεταβλητή Δωδεκάεδρος θα πρέπει να είναι False . Οπότε στα τμήματα ένα και δύο οι μεταβλητές Δωδεκάεδρο και Μεσσαίο θα είναι False .Οπότε στα τμήματα ένα και δύο αναγκαστικά θα πρέπει να έχουμε μία True μεταβλητή οπότε προκύπτουν οι εξής περιπτώσεις:

Η μεταβλητή Τετράεδρο True
Σε περίπτωση που η μεταβλητή Τετράεδρο είναι True τότε αναγκαστικά και η μεταβλητή Μεγάλο θα είναι True. Οπότε στα τμήματα ένα και δύο True θα είναι μόνο οι μεταβλητές Τετράεδρο και Μεγάλο άρα οι μεταβλητές Κύβος και Μικρό θα πρέπει να είναι False προκειμένου να βγεί συνολικά True η πρόταση .Άρα

ομοίως στην περίπτωση αυτή θα έχουμε ότι οι μεταβλητές Κύβος και Μικρό θα πρέπει έχουν την ίδια τιμή.

Η μεταβλητή Τετράεδρο False
Σε περίπτωση που η μεταβλητή Τετράεδρο είναι False τότε αναγκαστικά η μεταβλητή Μεγάλο θα είναι False.Οπότε στα τμήματα ένα και δύο προκειμένου να είναι True θα πρέπει οι μεταβλητές Κύβος και Μικρό να είναι True για να βγεί τελικά ότι η πρόταση είναι True. Άρα και στην περίπτωση αυτή έχουμε ότι πρέπει να έχουν την ίδια τιμή οι μεταβλητές Κύβος και Μικρό για να βγεί True συνολικά η πρόταση.

Παρατηρούμε ότι σε κάθε περίπτωση οι μεταβλητές Κύβος και Μικρό θα έχουν την ίδια τιμή . Οπότε ισχύει η πρόταση ότι Κύβος<=>Μικρό.

Πρόβλημα 7

Ζητείται να δείξουμε ότι η πρόταση αυτή είναι έγκυρη, δηλαδή ότι για κάθε ερμηνεία I η πρόταση είναι πάντα αληθής. Στο κεφάλαιο 7.5 του βιβλίου στην σελίδα 258 , αναφέρεται το εξής , το οποίο και θα χρησιμοποιήσουμε : 'Η πρότσαση α είναι έγκυρη αν και μόνο αν η ¬α είναι μη ικανοποιήσιμη (όπου μη ικανοποιήσιμη θεωρείται μια πρόταση για την οποία δεν υπάρχει I ώστε να είναι αληθής). Άρα με χρήση της ανάλυσης θα αποδείξουμε ότι η άρνηση της πρότασης μας είναι μη ικανοποιήσιμη , οπότε η πρόταση μας θα είναι και έγκυρη.

Επομένως ακολουθεί η μέθοδος ανάλυσης δηλαδή τα βήματα:

1. Απαλοιφή του <=>
2. Απαλοιφή τπυ =>
3. Το φέρουμε σε μορφή CNF
4. Απλοποιήσεις μέχρι να φτάσουμε σε κενή πρόταση

$\neg(A \iff B) \iff \neg((A \wedge B) \vee (\neg A \wedge \neg B))$
$\neg((A \Rightarrow B) \wedge (B \Rightarrow A)) \iff (\neg A \vee \neg B) \wedge (A \vee B)$
$\neg((\neg A \vee B) \wedge (\neg B \vee A)) \iff (\neg A \vee \neg B) \wedge (A \vee B)$
$(A \wedge \neg B) \vee (B \wedge \neg A) \iff (\neg A \vee \neg B) \wedge (A \vee B)$
$(A \wedge \neg B) \vee (B \wedge \neg A) \Rightarrow (\neg A \vee \neg B) \wedge (A \vee B) \wedge (\neg A \vee \neg B) \wedge (A \vee B) \Rightarrow (A \wedge \neg B) \vee (B \wedge \neg A)$
$((\neg A \vee B) \wedge (\neg B \vee A)) \vee ((\neg A \vee \neg B) \wedge (A \vee B)) \wedge ((A \wedge B) \vee (\neg A \wedge \neg B)) \vee ((A \wedge \neg B) \vee (B \wedge \neg A))$
$[(A \wedge \neg B) \vee (B \wedge \neg A)] \wedge [(A \wedge B) \vee (\neg A \wedge \neg B) \vee (\neg A \vee \neg B)] \wedge [(A \vee B)] \wedge [(\neg A \vee B)] \wedge [\neg B \vee A]$

Παρατηρούμε ότι παραπάνω προκύπτει κενό διάστημα πριν καν φτάσουμε στην CNF μορφή.

Προκειμένου να δείξουμε ότι η πρόταση είναι που μας δίνεται αρκεί να δείξουμε πως τόσο η πρόταση$\neg((A \wedge B) \vee (\neg A \wedge \neg B))$ έπεται λογικά της πρότασης $\neg((A \wedge B) \vee (\neg A \wedge \neg B))$, όσο η πρόταση $\neg((A \wedge B) \vee (\neg A \wedge \neg B))$ έπεται λογικά της $\neg(A \iff B)$.Αυτό γιατί το κάνουμε καθώς μαυτόν τον τρόπο ουσιαστικά θα 'αποδείξουμε' την ισοδυναμία και με κάθε τρόπο απο αριστερά προς τα δεξιά και απο δεξιά προς τα αριστερά.Επομένως έχουμε για την πρώτη περίπτωση ότι :
Το knowledge base μας θα είναι η πρόταση $\neg(A \iff B)$ και η προταση που λέμε ότι έπεται λογικά της knowledge base θα είναι η πρόταση : $\neg((A \wedge B) \vee (\neg A \wedge \neg B))$.Ισχύει για την μέθοδο ανάλυσης ότι θα εκτελέσουμε την ανάλυση σε πρόταση της μορφής KB$\wedge\neg a$, οπότε στην περίπτωση μας θα προκύψει η πρόταση $\neg(A \iff B) \iff ((A \wedge B) \vee (\neg A \wedge \neg B))$ την οποία και θα φέρουμε σε μορφή CNF . Οπότε θα φέρουμε την πρόταση σε μορφή CNF και θα προκύψει η πρόταση : $(\neg A \vee B) \wedge (\neg B \vee A) \wedge (\neg A \vee \neg B) \wedge (A \vee B)$. Οπότε εκτελόντας της απαλοιφής μεταξύ των διαζεύξεων προκύπτει κενό .

Στην συνέχεια ως knowledge base έχουμε την πρόταση $\neg((A \wedge B) \vee (\neg A \wedge \neg B))$ απο την οποία έπεται τώρα η πρόταση $\neg(A \iff B)$. Άρα ομοίως με παραπάνω θα φέρουμε την πρόταση $\neg((A \wedge B) \vee (\neg A \wedge \neg B))(A \iff B)$ σε μορφή CNF . Όπου και η μορφή αυτή θα είναι η εξής $((\neg A \vee \neg B) \wedge (A \vee B) \wedge \neg A \vee B) \wedge (\neg B \vee A))$.Στην συνέχεια μέσω της απαλοιφής των διαζεύξεων προκύπτει κενό .

Εν τέλει αποδείχθηκε πως η άρνηση της πρότασης δεν είναι ικανοποίησιμη άρα η πρόταση μας θα ε'ναι έγκυρη.

Περιγραφή της υλοποίησης που χρησιμοποιήθηκε για την αναπαράσταση του προβλήματος κενκεν :

Αρχείο kenken.txt

Αποφάσισα να δημιουργήσω δικά μου παραδείγματα να τα βάλω σε ένα αρχείο (kenken.txt) και να παίρνω απο εκεί παραδείγματα ώστε να κάνω τις δοκιμές. Κάθε γραμμή του αρχείου αποτελεί και ένα παράδειγμα . Όσο κατεβαίνουμε στις γραμμές τόσο πιο μεγάλη γίνεται η διάσταση του πίνακα Κενκεν, δηλαδή στην πρώτη γραμμή έχουμε ένα πίνακα κεν κεν 3x3 στην γραμμή δύο ένα πίνακα 4x4 και ούτε ο καθεξής.Να σημειωθεί ακόμα ότι το τέλος κάθε γραμμλης που περιέχει ένα παράδειγμα σηματοδοτείται με το σύμβολο δολάριο (θα χρειαστεί στον κώδικα μας όταν θα διαβάζουμε το παράδειγμα).

Ο πρώτος αριθμός σε κάθε γραμμή ορίζει το μέγεθος του πίνακα για παράδειγμα στην πρώτη γραμμή έχουμε τον αριθμό 3 , αυτό σημαίνει ότι το παράδειγμα θα είναι ένας πίνακας μεγέθους 3x3 , στην δεύτερη γραμμή 4 αυτό σημαίνει ότι το παράδειγμα αυτό θα είναι ένας πίνακας μεγέθους 4x4 .

Οι μεταβλητές του προβλήματος (variables) αναπαρίστανται ως πραγματικοί αριθμοί με δύο μέροι σύμφωνα με την γραμμή και την στήλη στην οποία βρίσκονται .Συγκεκριμένα το πρώτο μέρος δηλώνει την γραμμή και το δεύτερο μέρος την στήλη πχ η μεταβλητή 1.1 αναφέρεται στον πίνακα κενκεν για το κουτί στην πρώτη γραμμή και πρώτη στήλη.

Κάθε κλίκα ενός παραδείγματος αναπαρίσταται με δύο αγκύλες .Ουσιαστικά μέσα σε μία αγκύλη έχουμε τις μεταβλητές που την απαρτίζουν , οι οποίες χωρίζονται με κόμμα για να ξεχωρίζουν . Αμέσως μετά την δεύτερη αγκύλη έχουμε το σύμβολο του ίσου και να το ακολουθεί ένας αριθμός ο οποίος αριθμός θα είναι ο στόχος που θα πρέπει να φτάσει με κάποια πράξη η κλίκα . Μετά απο αυτόν τον αριθμό στόχο ακολουθεί ένα σύμβολο πράξης το οποίο μας υποδηλώνει και την πράξη που θα κάνουμε προκειμένου να φτάσουμε σ'αυτόν τον αριθμό στόχο στην συγκεκριμένη κλίκα. Κάθε κλίκα χωρίζεται με ένα κόμμα.

Αρχείο csp.py

Για το πρόβλημα Κενκεν έχω φτιάξει μια κλάση με η οποία θα κληρονομεί απο την κλάση CSP.

Πριν την δημιουργία της κλάσης έχω φτιάξει μια συνάρτηση reading που θα διαβάζει την γραμμή που θα της λέμε με έναν αριθμό απο το αρχείο kenken.txt , όπου όπως είπαμε κάθε γραμμή αποτελεί και ένα παράδειγμα.
Ένα csp πρόβλημα θα διαθέτει μεταβλητές,πεδία , περιορισμούς **Μεταβλητές:**
Οι μεταβλητές του προβλήματος θα είναι κάθε κελί του πίνακα όπου θα έχει τον αριθμό για παράδειγμα 1.1,1.2 κ.τ.λ όπως περιγράφω και ποιο πάνω στο αρχείο κενκεν.τεχτ.Κάθε τέτοια μεταβλητή ουσιαστικά θα παίρνει μια τιμή απο το πεδιό της χωρίς να παραβιάζει τους περιορισμούς. Ουσιαστικα για τις μεταβλητές αυτό που έκανα είναι να τις βάλω σε μια λίστα . Για τις μεταβλητές ξέρω πόσες θα είναι και ποιες για το παράδειγμα που εκτελείται εκείνη την στιγμή από το μέγεθος του πίνακα δηλαδή αν είναι 3χ3 τότε ξέρω ότι οι μεταβλητές στο δεύτερο μέλος (.)κάθε μίας μετά το κόμμα δηλαδή θα φτάνει μέχρι το 3 αλλά και πριν το κόμμα (.)θα φτάνει μέχρι το 3 . Οπότε κρατάω σε μια λίστα τις μεταβλητές ώστε ώστε στην συνέχεια να καταφέρω να φτιάξω τα dictionaries για τους γείτονες τους ,κλίκες κ.α.
Πεδία :
Τα πεδία θα είναι οι τιμές που θα μπορούν να πέρουν οι μεταβλητές και ανάλογα με το μέγεθος του πίνακα θα έχουμε και το πεδίο των μεταβλητών. Τα domains είναι τα πιο απλά το μόνο που κάνω είναι να δίνω τιμές μέχρι το μέγεθος του πίνακα δηλαδή για παράδειγμα αν είναι 3χ3 κάθε μεταβλητή μπορεί να πάρει τις τιμές 1,2,3 κ.τ.λ.
Γείτονες :
Για τους γείτονες έφτιαξα ένα dictionary neighbors στο οποίο έβαλα για κάθε μεταβλητή τους γείτονες που διαθέτει στην ίδια στήλη και σειρά και επίσης έφτιαξα ένα dictionary clique.neighbors στο οποίο

βάζω τους γείτονες που είναι γείτονες λόγω κλίκας .

goallist :

Αυτό το dictionary το έφτιαξα για να ξέρουμε για κάθε μταβλητή τον τρόπο που θα φτάσουμε στον αριθμό στόχου της κλικας στην οποία βρίσκεται η μεταβλητή , που θα είναι πρόσθεση , αφαίρεση , πολλαπλασιασμός , διαίρεση ή και θαυμαστικό δηλαδή τίποτα θα πρέπει να είναι μια συγκεκριμένη τιμή.

goallist2 :

Αυτό το dictionary το έφτιαξα για να ξέρουμε για κάθε μεταβλητή ποιος είναι ο αριθμός στόχου της κλίκας στην οποία βρίσκεται.

Περιορισμοί:

Για τους περιορισμούς σκέφτηκα ότι θα υπάρχουν δυο ειδών περιορισμοί : οι περιορισμοί όσον αφορά την ίδια σειρά και την ίδια στήλη αλλά και τους περιορισμούς που προκύπτουν μέσα σε μια κλίκα. Στο συγκεκριμένο πρόβλημα έχουμε δεν μπορούμε να έχουμε στην ίδια στήλη τον ίδιο αριθμό και στην ίδια σειρά ομοίως δεν μπορούμε να έχουμε τον ίδιο αριθμό.Όσον αφορά τον περιορισμό της κλίκας δεν μπορούμε να ξεπερνάμε τον αριθμό που έχει ως στόχο η κλίκα.

Εκτέλεση προγράμματος Για την εκτέλεση του προγράμματος , βάζετε κάθε φορά στην γραμμή 1125 σαν παράμετρο έναν αριθμό για το πιο παράδειγμα να τρέξετε 1-¿(3χ3),2-¿(4χ4)..... Στην γραμμή 1128 βάζετε όποια συνάρτηση θέλετε να χρησιμοποιηθεί για να τρέξει μεταξύ των fc και mac , όπου και τυπώνεται το αποτέλεσμα , οι αναθέσεις , ο χρόνος και ο αριθμός καλέσματος της συνάρτησης constraints.