email at: sdi2000055@di.uoa.gr

Περιεχομενα

1. Ασκηση 1

1.

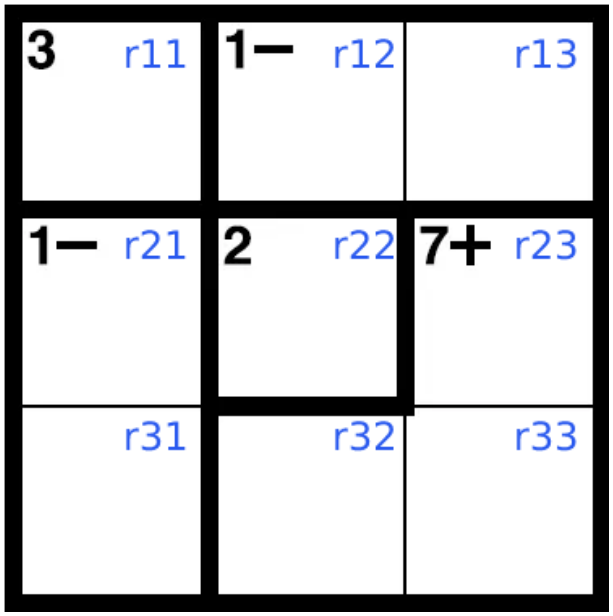
Η ΕΞΗΓΗΣΗ ΤΟΥ ΚΩΔΙΚΑ ΒΡΙΣΚΕΤΑΙ ΣΤΗΝ ΕΝΟΤΗΤΑ 8 (σελίδα 14 του PDF)

Μοντελοποίηση

Για την αναλυτική αναπαράσταση του προβλήματος ως ένα πρόβλημα ικανοποίησης περιορισμών θα πρέπει να ορίσουμε σαφώς:

Μεταβλητές, πεδίο και περιορισμούς

Θα ορίσω μια μοντελοποίηση για το πρόβλημα Kenken σε μια απλούστερη μορφή του, για ένα 3×3 grid και έπειτα θα καταγράψω και τους γενικούς κανόνες για ένα $n \times n$ grid.



Για το συγκεκριμένο πρόβλημα έχουμε 9 μεταβλητές $r11$ εώς $r33$ (γενικώς $n \cdot n$ μεταβλητές), και το πεδίο είναι το $\{1, 2, 3\}$ (γενικώς έχουμε πεδίο $V = \{1, 2, 3, \ldots, n\}$). Οι περιορισμοί είναι απόρροια των κανόνων του ίδιου του παιχνιδιού. Έτσι έχουμε ότι:
Ορίζω ότι ένα κελί έχει μια μορφή r_i όπου $i = kl, \quad k, l \in V$. Για παράδειγμα για το κελί r_{23}, έχουμε $i = 23, k = 2, l = 3$.

2) Κάθε νούμερο θα πρέπει να εμφανίζεται μια φορά σε κάθε στήλη. Άρα έχουμε ότι: $r_{11} \neq r_{21} \neq r_{31}, r_{12} \neq r_{22} \neq r_{32}, r_{13} \neq r_{23} \neq r_{33}$.
Γενικώς $r_{i,j} \neq r_{k,j}, \quad i \neq k, \quad i, k \in V$ για $j = 1, \ldots, n$
1) Κάθε νούμερο θα πρέπει να εμφανίζεται μια φορά σε κάθε γραμμή. Άρα έχουμε ότι: $r_{11} \neq r_{12} \neq r_{13}, r_{21} \neq r_{22} \neq r_{23}, r_{31} \neq r_{32} \neq r_{33}$.
Γενικώς $r_{i,j} \neq r_{i,k}, \quad j \neq k, \quad j, k \in V$ για $i = 1, \ldots, n$
3) Σε κάθε γκρουπ κελιών, τα ψηφία των κελιών να πετυχαίνουν το σωστό αποτέλεσμα με χρήση του αντίστοιχου τελεστή. Έστω ότι ο τελεστής ($\square$) είναι μια από τις βασικές μαθηματικές πράξεις: $+, -, \times, \div$ και ένα αποτέλεσμα R. Τότε για κάθε κελί στο γκρούπ θα πρέπει να ισχύει: $\underbrace{r_i \square r_j \square \ldots r_k}_{\text{στοιχεία του γκρουπ}} = R$. Για παράδειγμα στο παραπάνω παράδειγμα της φωτογραφίας θα πρέπει να ισχύει ότι τα στοιχεία r_{12} και r_{13} με αφαίρεση να μας οδηγούν στο 1, τα r_{23}, r_{32}, r_{33} με πρόσθεση να οδηγούν στο 7 κλπ...

2-3.
Δεν επέλεξα μόνο 2 τρόπους αναζήτησης. Αντίθετα έλεγξα τους αλγορίθμους αναζήτησης BT, FC, MAC μόνους τους καθώς και με σε συνδυασμό με την MRV. (Όπως και στις διαφάνειες του μαθήματος)

- N=4 lasted Time and made Assignments assignments
- BT lasted 0.0009906291961669922 and made 7 assignments
- FC lasted 0.0007956027984619141 and made 7 assignments
- MAC lasted 0.002397775650024414 and made 7 assignments
- BT+MRV lasted 0.0012578964233398438 and made 7 assignments
- FC+MRV lasted 0.0025229454040527344 and made 8 assignments
- MAC+MRV lasted 0.006712436676025391 and made 7 assignments

- N=5 lasted Time and made Assignments assignments
- BT lasted 0.05651998519897461 and made 296 assignments
- FC lasted 0.020213842391967773 and made 40 assignments
- MAC lasted 0.1282033920288086 and made 88 assignments
- BT+MRV lasted 0.031006336212158203 and made 111 assignments
- FC+MRV lasted 0.02799677848815918 and made 45 assignments
- MAC+MRV lasted 0.08553218841552734 and made 35 assignments

- N=6 lasted Time and made Assignments assignments
- BT lasted 0.5179715156555176 and made 2013 assignments
- FC lasted 0.24219989776611328 and made 991 assignments
- MAC lasted 0.1708986759185791 and made 215 assignments
- BT+MRV lasted 0.04543590545654297 and made 70 assignments
- FC+MRV lasted 0.016040802001953125 and made 51 assignments
- MAC+MRV lasted 0.10108184814453125 and made 41 assignments

- N = 7 (1) lasted Time and made Assignments assignments
- BT lasted 13.194404125213623 and made 48781 assignments
- FC lasted 2.3827178478240967 and made 3584 assignments
- MAC lasted 5.005670070648193 and made 5239 assignments
- BT+MRV lasted 0.40851378440856934 and made 456 assignments
- FC+MRV lasted 0.07795882225036621 and made 47 assignments
- MAC+MRV lasted 0.25486063957214355 and made 31 assignments

- N = 7 (2) lasted Time and made Assignments assignments
- BT lasted 13.194404125213623 and made 48781 assignments
- FC lasted 2.3827178478240967 and made 3584 assignments
- MAC lasted 5.005670070648193 and made 5239 assignments
- BT+MRV lasted 0.40851378440856934 and made 456 assignments
- FC+MRV lasted 0.07795882225036621 and made 47 assignments
- MAC+MRV lasted 0.25486063957214355 and made 31 assignments

- N = 8 (1) lasted Time and made Assignments assignments
- BT lasted DNF and made DNF assignments
- FC lasted 6.067245006561279 and made 30680 assignments
- MAC lasted 23.092783451080322 and made 58628 assignments
- BT + MRV lasted 0.4821617603302002 and made 1839 assignments
- FC + MRV lasted 0.0561373233795166 and made 76 assignments
- MAC + MRV lasted 0.32049036026000977 and made 61 assignments

- N = 8 (2) lasted Time and made Assignments assignments
- BT lasted DNF and made DNF assignments
- FC lasted 0.7823257446289062 and made 3220 assignments
- MAC lasted 68.02875280380249 and made 62576 assignments
- BT + MRV lasted 31.989097118377686 and made 52185 assignments
- FC + MRV lasted 0.1085805892944336 and made 169 assignments
- MAC + MRV lasted 0.2760286331176758 and made 67 assignments

- N = 9 (1) lasted Time and made Assignments assignments

- BT lasted DNF and made DNF assignments
- FC lasted 4.721073150634766 and made 17171 assignments
- MAC lasted 142.34558129310608 and made 230691 assignments
- BT + MRV lasted 172.83838653564453 and made 451862 assignments
- FC + MRV lasted 0.10938096046447754 and made 134 assignments
- MAC + MRV lasted 0.4790527820587158 and made 101 assignments

- N = 9 (2) lasted Time and made Assignments assignments
- BT lasted DNF and made DNF assignments
- FC lasted 5.07830286026001 and made 17072 assignments
- MAC lasted 10.567398071289062 and made 11078 assignments
- BT + MRV lasted 352.9237883090973 and made 486947 assignments
- FC + MRV lasted 0.10108423233032227 and made 109 assignments
- MAC + MRV lasted 1.0737078189849854 and made 151 assignments

Όπως περιμέναμε, ο καλύτερος αλγόριθμος με βάσει την μικρότερη απόδοση assignments είναι ο MAC ο οποίος consistently είχε το μικρότερο αριθμό ανεξάρτητα από το μέγεθος του προβλήματος. Σε συνεργασία με τον MRV τα αποτελέσματα είναι εντυπωσιακά, καθώς στο πιο μεγάλο grid γίνονται μόνο 151 assignments (!). Έπεται ο αλγόριθμος FC ο οποίος και αυτός με την χρήση MRV ελαχιστοποιεί κατά πολύ τα assignments. Τελευταίος έρχεται ο αλγόριθμος BT ο οποίος χρειάζεται την χρήση του MRV για να πέσει σε υποφερτά επίπεδα χρόνου. Η διαφορά ωστόσο παραμένει χαοτική αφού οι αλγόριθμοι FC+MRV, MAC+MRV τελειώνουν σε πολύ 1 δευτερόλεπτο. Αξίζει να σημειωθεί ότι η καθυστέρηση του MAC προέρχεται από την χρήση αλγορίθμων arc consistency οι οποίοι προσδίδουν overhead στον χρόνο (αλλά όχι στα assignments, όπως και αναφέραμε παραπάνω)
4. Ο αλγόριθμος min conficts δρα πιο περίεργα. Για το grid μεγέθους 4 άλλοτε τερματίζει και με ευκολία (κάνοντας assignments της τάξης των 10,20) άλλοτε δεν τερματίζει λόγω εξάντλησης των 100.000 βημάτων. Για μεγέθη 5 και πάνω, δεν μπόρεσα να τρέξω τον αλγόριθμο σε σημείο που να τελειώνει με εύρεση λύσης και όχι λόγω εξάντλησης βημάτων.
Ο κώδικας του περιλαμβάνει ένα random.choice το οποίο πιστεύω ότι ευθύνεται για την συμπεριφορά στα $n = 4$. Σαφώς σε αυτή την αδυναμία συμβάλλει το ότι το KenKen είναι ένα πρόβλημα αρκετά δύσκολο στην επίλυση. Μην ξεχνάμε ότι και παραπάνω, αναγκαστήκαμε να χρησιμοποιήσουμε εντατικά heuristics για να πέσει η πολυπλοκότητα.

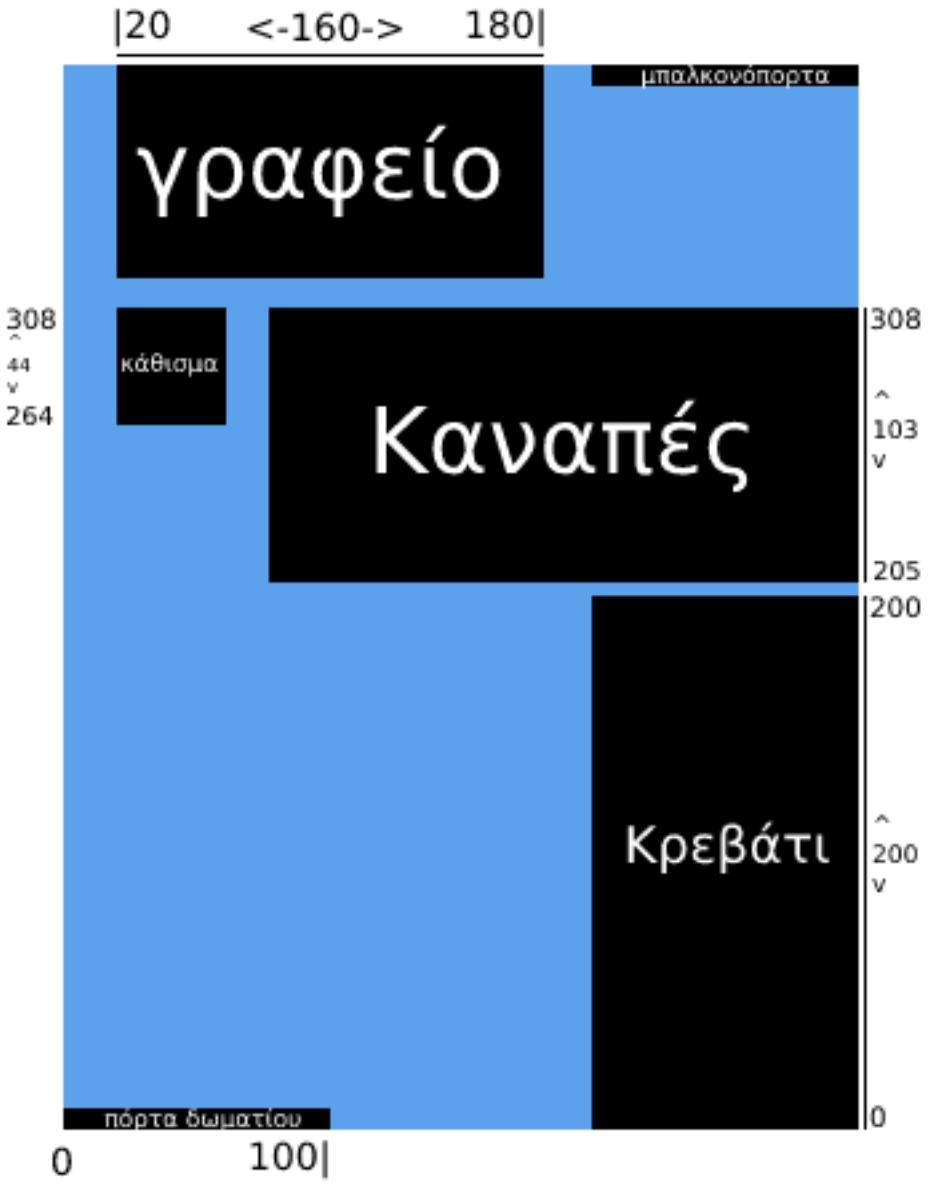
2. Ασκηση 2

Αρχικά αντιμετώπισα το πρόβλημα σαν να μην επιτρέπεται η περιστροφή των επίπλων, αφού κάτι τέτοιο νομίζω θα δυσκόλευε την λύση κατα πολύ αν αφήναμε την περιστροφή κατά οποιαδήποτε γωνία ϕ (δηλαδή όχι μόνο 90,180,270 μοιρών κλπ), καθώς και τον ορισμό του πεδίου τιμής. Αν επιτρέπαμε την περιστροφή κατά μόνο 90(ή

180,270) μοίρες τότε το πρόβλημα γίνεται πιο απλό αλλά χωρίς κάποια σπουδαία πρόσθεση στην επίλυση. Η λογική παραμένει ολόιδια.

Αρχικά θα μοντελοποίησουμε το πρόβλημα σας ένα πρόβλημα ικανοποίησης περιορισμών. Έτσι θα χρειαστούμε: **μεταβλητές, πεδίο τιμών και περιορισμούς**. Από εδώ και κάτω θα τοποθετήσουμε ένα ορθοκανονικό σύστημα αξόνων στην κάτω αριστερή γωνία του δωματίου, με μέγιστο πλάτος τα $300cm$ και μέγιστο μήκος $400cm$.
Μεταβλητές
Οι μεταβλητές του προβλήματός μας είναι τα έπιπλα που καλούμαστε να τοποθετήσουμε, επομένως έχουμε το κρεβάτι, το γραφείο, την καρέκλα γραφείου και τον καναπέ. Για την κάθε μεταβλητή θέλουμε να κρατάμε όλες τις πληροφορίες που μας δίνονται άρα θ α έχουμε ένα συνόλο μεταβλητών όπως το παρακάτω: (κρεβάτι, 100, 200, 80), (γραφείο, 160, 80, 90), (καρέκλα γραφείου, 41, 44, 57), (καναπές, 221, 103, 84).
Πεδίο τιμών
Ζητούμενο μας είναι το αναθέσουμε μια τιμής για κάθε μεταβλητή η οποία θα ορίζει την θέση της στο δωμάτιο. Επομένως κάθε μεταβλητή θα δέχεται μια θέση (πλάτος, μήκος). Ωστόσο θα πΐρπειε να είμαστε σε θέση να τα χωρέσουμε και στο δωμάτιο, άρα για την κάθε μεταβλητή έχουμε και ένα όριο τιμών.
Τα όρια είναι τα εξής:
· Για το κρεβάτι έχουμε $dom($ κρεβάτι $) = \pi_{\text{κρεβάτι}}, \mu_{\text{κρεβάτι}}, \pi_{\text{κρεβάτι}} \in [0, 200], \mu_{\text{κρεβάτι}} \in [0, 200]$
· Για το γραφείο έχουμε $dom($ γραφείο $) = \pi_{\text{γραφείο}}, \mu_{\text{γραφείο}}, \pi_{\text{γραφείο}} \in [0, 140], \mu_{\text{γραφείο}} \in [0, 320]$
· Για την καρέκλα έχουμε $dom($ καρέκλα $) = \pi_{\text{καρέκλα}}, \mu_{\text{καρέκλα}}, \pi_{\text{καρέκλα}} \in [0, 259], \mu_{\text{καρέκλα}} \in [0, 356]$
· Για τον καναπέ έχουμε $dom($ καναπέ $) = \pi_{\text{καναπέ}}, \mu_{\text{καναπέ}}, \pi_{\text{καναπέ}} \in [0, 79], \mu_{\text{καναπέ}} \in [0, 297]$
Περιορισμοί
· Τα έπιπλα δεν πρέπει να εφάπτονται ή να πατάνε το ένα πάνω στο άλλο.
· Το γραφείο θα πρέπει να είναι δίπλα σε κάποια είσοδο φωτός στο δωμάτιο.
Παρατήρηση: Αν και δεν αναφέρεται ρητά ως κάποιος περιορισμός, υποθέτω ότι επίσης ως 3ο περιορισμό το ότι:
· Δεν θα πρέπει να τοποθετήσουμε τα έπιπλα έτσι ώστε να μπλοκάρεται το άνοιγμα της πόρτας (όπως φαίνεται εμφανώς και από το σχήμα)
1. Έστω δύο έπιπλα: έπιπλο$_1$ και έπιπλο$_2$. Αρχικά θα ξεκινήσω με το τι συμβαίνει αν έχουμε overlap στο μήκος. Αυτό συμβαίνει αν το έπιπλο$_1$ (ή το έπιπλο$_2$) τελειώνει μετά από το μήκος που έχει ξεκινήσει το έπιπλο$_2$ (ή το έπιπλο$_1$). Ομοίως το ίδιο συμβαίνει αν το έπιπλο$_2$ (ή το έπιπλο$_1$) τελειώσει μετά το τελείωμα του έπιπλο$_1$ (ή το έπιπλο$_2$).
Άρα οι κανόνες από τα παραπάνω είναι ότι: (χωρίς να έχει σημασία το για ποιο έπιπλο μιλάμε):
· έπιπλο$_1$.μ+μήκοςΕπίπλου $<$ έπιπλο$_2$.μ

· $\text{έπιπλο}_1.\mu > \text{έπιπλο}_2.\mu + \text{μήκοςΕπίπλου}$
Ομοίως, με την ίδια λογική για το πλάτος, έχουμε τους περιορισμούς:
· $\text{έπιπλο}_1.\pi + \text{πλάτοςΕπίπλου} < \text{έπιπλο}_2.\pi$
· $\text{έπιπλο}_1.\pi > \text{έπιπλο}_2.\pi + \text{πλάτοςΕπίπλου}$
2.
Ορίζουμε την μέγιστη απόσταση κατά την οποία κάνουμε δεκτή την παραδοχή ότι το γραφείο ειναι κοντά στην μπαλκονόπορτα, με την παραδοχή ότι αυτή δρα ως πηγή φωτός. Έστω ότι αυτή η απόσταση είναι 20 εκατοστά. Η μπαλκονόπορτα βρίσκεται τοποθετημένη στο άκρα (πλάτος,μήκος) = (200,400) (200 αφού έχει πλάτος 100 εκατοστών) και (πλάτος,μήκος) = (300,400). Θα πρέπει να υπολογίσουμε τις αποστάσεις μεταξύ των άκρων της μπαλκονόπορτας και των δύο πάνω άκρων του γραφείου. Έστω ότι αυτη η απόσταση είναι ευκλείδια (θα μπορούσαμε να έχουμε οποιαδήποτε τέτοια αλλά είναι ικανοποιητική για τα δεδομένα μας). Επομένως έχουμε υπολογισμό των:
min{
$\text{distance}_{euclidean}(\ \text{γραφείο}_{\text{αριστερή γωνία}},\ \text{μπαλκονόπορτα}_{\text{αριστερή γωνία}}\ ),$
$\text{distance}_{euclidean}(\ \text{γραφείο}_{\text{αριστερή γωνία}},\ \text{μπαλκονόπορτα}_{\text{δεξιά γωνία}}\ ),$
$\text{distance}_{euclidean}(\ \text{γραφείο}_{\text{δεξιά γωνία}},\ \text{μπαλκονόπορτα}_{\text{αριστερή γωνία}}\ ),$
$\text{distance}_{euclidean}(\ \text{γραφείο}_{\text{δεξιά γωνία}},\ \text{μπαλκονόπορτα}_{\text{δεξιά γωνία}}\ )$
}
Θεωρώντας ότι και η πόρτα είναι πηγή φωτός, το ίδιο ισχύει και για τις αποστάσεις από την πόρτα.
Αν η μέγιστη αυτή απόφαση είναι μικρότερη της μέγιστης αποδεκτής απόστασης που έχουμε ορίσει τότε είμαστε αρκούντως κοντά.
3. Αφού η πόρτα έχει πλάτος $100cm$, καταλαμβάνει έναν χώρο 100×100. Άρα έχουμε περιορισμούς ότι πρέπει $\pi > 100$ ή $\mu > 100$.
Μια λύση για το πρόβλημα παρουσιάζω παρακάτω (Χωρίς περιστροφή):



3. Ασκηση 3

3.1)
Για την μοντελοποιήση σε CSP θα χρειαστεί να ορίσουμε: **Μεταβλητές, Πεδίο Τιμών και Περιορισμούς**
Μεταβλητές

Είναι οι 5 ενέργειες που χρείζουν χρονοπρογραμματισμό, δηλαδή: $A_i, \quad i \in [1, 5]$

Πεδίο Τιμών

Οι μεταβλητές μπορούν να εκκινήσουνσ τις 9:00, 10:00 ή 11:00, και αφού έχουν σταθερό χρόνο ολοκλήρωσης τα 60 λεπτά, μας αρκεί να κρατάμε τους χρόνους εκκίνησης. Άρα το πεδίο τιμών της κάθε μεταβλητής είναι: $\Pi = \{9:00, 10:00, 11:00\}$

Περιορισμοί

Με βάσει τους κανόνες που μας δίνονται:

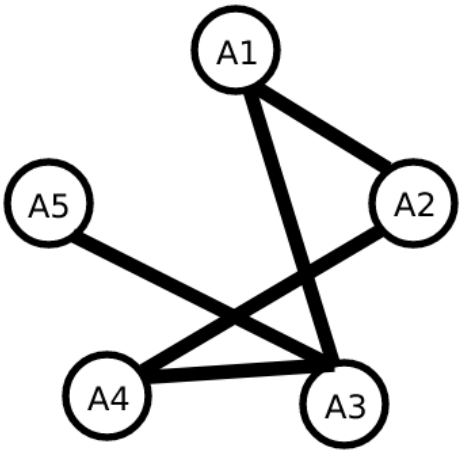
· Η A1 πρέπει να αρχίσει μετά την A3: $\mathrm{A1}_{\text{Εκκίνηση}} > \mathrm{A3}_{\text{Εκκίνηση}}$

· Η A3 πρέπει να αρχίσει πριν την A4 και μετά την A5: $\mathrm{A5}_{\text{Εκκίνηση}} < \mathrm{A3}_{\text{Εκκίνηση}} < \mathrm{A4}_{\text{Εκκίνηση}} \equiv \mathrm{A5}_{\text{Εκκίνηση}} < \mathrm{A3}_{\text{Εκκίνηση}} \wedge \mathrm{A3}_{\text{Εκκίνηση}} < \mathrm{A4}_{\text{Εκκίνηση}}$

· Η A2 δεν μπορεί να εκτελείται την ίδια ώρα με την A1 ή την A4: $\mathrm{A2}_{\text{Εκκίνηση}} \neq \mathrm{A4}_{\text{Εκκίνηση}} \wedge \mathrm{A2}_{\text{Εκκίνηση}} \neq \mathrm{A1}_{\text{Εκκίνηση}}$

· Η A4 δεν μπορεί να αρχίσει στις 10:00: $\mathrm{A4}_{\text{Εκκίνηση}} \neq 10:00$

2)



3)
Για να ξεκινήσουμε τον αλγόριθμο AC-3 αρχικά δημιουργούμε τους περιορισμούς τους οποίους αναγράφουμε τώρα και από τις δύο μεριές ώστε να είναι διαφορετικοί για τον αλγόριθμο.
Επομένως έχουμε τους παρακάτω περιορισμούς για τον αλγόριθμο, και για να μην αναγράφω παντού την ¨εκκίνηση¨, θα υποθέσουμε ότι όλοι οι παρακάτω περιορισμοί αναφέρονται στους χρόνους εκκίνησης χωρί να το αναγράφουμε:
· $A1 > A3, A3 < A1$
· $A5 < A3, A3 > A5$
· $A3 < A4, A4 > A3$
· $A2 \neq$ A1,A1 $\neq$ A2
· $A2 \neq$ A4, A4 $\neq$ A2
·$A4 \neq 10:00, 10:00 \neq A4$

1) Ξεκινάμε με $A_1 = 9$.
Ελέγχω από τα αρχς πρώτο το $< A_3, A_1 >$, από το οποίο προκύπτει το $D_{A_3} = \emptyset$ άρα και ο αλγόριθμος τερματίζει αφού δεν υπάρχει λύση.

2) Ξεκινάμε με $A_1 = 10$.
Ελέγχω την ακμή $< A_2, A_1 >$ από το οποίο προκύπτει $D_{A_2} = \{9, 11\}$. Δεν προσθέτω την $< A_4, A_2 >$ γιατί υπάρχει ήδη στην ουρά.

Έπειτα ελέγχω την ακμή $< A_3, A_1 >$ από την οποία προκύπτει $D_{A_3} = \{9\}$. Δεν προσθέτω τις ακμές $< A_5, A_3 >, < A_4, A_3 >$ αφού είναι ήδη στην ουρά.
Έπειτα ελέγχω την ακμή $< A_3, A_5 >$ από την οποία προκύπτει ότι $D_{A_3} = \emptyset$ άρα ο αλγόριθμος τερματίζει αφού δεν υπάρχει λύση.

3) Ξεκινάμε με $A_1 = 11$.

Ελέγχω την ακμή $< A_2, A_1 >$ από το οποίο προκύπτει $D_{A_2} = \{9, 10\}$. Δεν προσθέτω την $< A_4, A_2 >$ γιατί υπάρχει ήδη στην ουρά.
Έπειτα ελέγχω την ακμή $< A_3, A_1 >$ από την οποία προκύπτει $D_{A_3} = \{9, 10\}$. Δεν προσθέτω τις ακμές $< A_5, A_3 >, < A_4, A_3 >$ αφού είναι ήδη στην ουρά.
Έπειτα ελέγχω την ακμή $< A_4, A_3 >$ από την οποία προκύπτει $D_{A_4} = \{11\}$. Δεν προσθέτω τις ακμές $< A_2, A_4 >$ αφού είναι ήδη στην ουρά.
Έπειτα ελέγχω την ακμή $< A_4, A_2 >$ η οποία είναι συνεπής. Ομοίως για τις $< A_2, A_4 >$ και $< A_3, A_3 >$
Έπειτα ελέγχω την ακμή $< A_3, A_5 >$ από την οποία προκύπτει $D_{A_3} = \{10\}$. Δεν προσθέτω την ακμές $< A_1, A_3 >$ αφού είναι ήδη στην ουρά, αλλά προσθέτω ξανά την $< A_4, A_3 >$.
Έπειτα ελέγχω την ακμή $< A_5, A_3 >$ από την οποία προκύπτει ότι $D_{A_5} = \{9\}$.
Τώρα ελέγχω όλες τις άλλες ακμές και είναι όλες συνεπείς άρα δεν χάνω χρόνο στην αναγραφή τους.
Άρα μέχρι στιγμής έχω: $D_{A_2} = \{9, 10\}, D_{A_3} = \{10\}, D_{A_4} = \{11\}, D_{A_5} = \{9\}$
Τώρα προχωράμε στην ανάθεση της τιμής: $A_2 = 9$, για την οποία συνεχίζουμε να έχουμε συνέπεια.
Άρα φτάσαμε στην λύση:
$A_1 = 11, A_2 = 9, A_3 = 10, A_4 = 11, A_5 = 9$

4. Ασκηση 4

α)

A	B	C	D	$\neg(A \wedge \neg B \wedge C \to D) \iff (\neg A \to (B \to (C \to D)))$
F	F	F	F	F
F	F	F	T	F
F	F	T	F	F
F	F	T	T	F
F	T	F	F	F
F	T	F	T	F
F	T	T	F	T
F	T	T	T	F
T	F	F	F	F
T	F	F	T	F
T	F	T	F	T
T	F	T	T	F
T	T	F	F	F
T	T	F	T	F
T	T	T	F	F
T	T	T	T	F

β)

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

γ)

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

δ)

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

Άρα έχουμε:
1. Από τον ορισμό της εγκυρότητας (Μια πρόταση ϕ είναι έγκυρη αν για κάθε ερμηνεία I, ισχύει $I(\phi) = true$), βλέπουμε ότι δεν υπάρχει κάποια πρόταση η οποία να έγκυρη, αφού για καμία δεν ισχύει ότι για κάθε ερμηνεία το αποτέλεσμα είναι true.
2. Από τον ορισμό της ικανοποιησιμότητας, βλέπουμε ότι όλες οι παραπάνω προτάσεις είναι ικανοποίησιμες αφού υπάρχει μια ερμηνεία ώστε να βγαίνουν αληθείς. Συγκεκριμένα, στην πρώτη πρόταση η ερμηνεία (F, T, T, F) στην δεύτερη η (F, F), στην τρίτη η (F, F, F) και στην τέταρτη η (F, T, F)
3. Καμία πρόταση δεν είναι μη ικανοποιήσιμη, αφού για κάθε μια υπάρχει τουλάχιστον μία ερμηνεία ώστε να είναι true.
4. Όλες οι προτάσεις έχουν τουλάχιστον ένα μοντέλο, αφού όλες είναι και ικανοποιήσιμες.
5. Καμία πρόταση δεν είναι ταυτολογία αφού καμία πρόταση δεν είναι έγκυρη.

5. Ασκηση 5

Αν μοντελοποιήσουμε την πρόταση με προτασιακή λογική τότε έχουμε ότι αν $P =$ "ο Σήφης είναι καλός μάγειρας", $Q =$ "Εγώ είμαι αστροναύτης", η πρόταση μπορεί είναι ισοδύναμη με: $P \Rightarrow Q$.
Επειδή το $Q = false$, αυτό σημαίνει ότι, έχουμε δύο ενδεχόμενα ανάλογα με το αν ισχύει το P. Αν το $P = true$, τότε έχουμε ότι από κάτι αληθές συνεπάγεται κάτι ψευδές, που δεν ισχύει άρα όλη η πρότασή μας είναι ψευδής. Άρα απομένει το να ισχύει ότι $P = false$. Άρα από μια ψευδή υπόθεση (ότι ο Σήφης είναι καλός μάγειρας), μπορεί να βγει ένα ψευδές αποτέλεσμα (ότι εγώ είμαι αστροναύτης).

6. Ασκηση 6

Αρχικά μετατρέπουμε από προτασιακή λογική σε προτάσεις τις δωθείσες προτάσεις. Έχουμε ότι:
· *μικρό*, θα παριστάνει την πρόταση: "Το σχήμα α είναι μικρό"
· *μεσαίο*, θα παριστάνει την πρόταση: "Το σχήμα α είναι μεσαίο"
· *μεγάλο*, θα παριστάνει την πρόταση: "Το σχήμα α είναι μεγάλο"
· *κύβος*, θα παριστάνει την πρόταση: "Το σχήμα α είναι κύβος"
· *δωδεκάεδρο*, θα παριστάνει την πρόταση: "Το σχήμα α είναι δωδεκάεδρο"
· *τετράεδρο*, θα παριστάνει την πρόταση: "Το σχήμα α είναι τετράεδρο"

Οι δοσμένες προτάσεις παριστάνονται σε προτασιακή λογική ως:
μικρό $\oplus$ μεσαίο $\oplus$ μεγάλο
κύβος $\oplus$ δωδεκάεδρο $\oplus$ τετράεδρο
μεσαίο $\iff$ δωδεκάεδρο
μεγάλο $\iff$ τετράεδρο
Εμείς θέλουμε να δείξουμε ότι: μικρό $\iff$ κύβος δηλαδή ότι μικρό $\Rightarrow$ κύβος και κύβος $\Rightarrow$ μικρό
Τώρα πρέπει να αποδείξουμε ότι:
(μικρό $\oplus$ μεσαίο $\oplus$ μεγάλο)$\wedge$(κύβος $\oplus$ δωδεκάεδρο $\oplus$ τετράεδρο)$\wedge$(μεσαίο $\iff$ δωδεκάεδρο) $\wedge$ (μεγάλο $\iff$ τετράεδρο) $\models$ μικρό $\Rightarrow$ κύβος

Τώρα θα χρησιμοποιήσω τον κανόνα της αναλύσης για να καταλήξω σε αντίφαση στην πρόταση $KB \wedge \neg a$

· Στόχος = μικρό $\iff$ κύβος
· Άρνηση στόχου: $\neg$(μικρό $\iff$ κύβος)
CNF
·μικρό $\oplus$ μεσαίο $\oplus$ μεγάλο = ($\neg$μικρό $\vee\neg$ μεσαίο $\vee$ μεγάλο)$\wedge$($\neg$μικρό $\vee$ μεσαίο $\vee$ $\neg$ μεγάλο) $\wedge$ (μικρό $\vee\neg$ μεσαίο $\vee\neg$ μεγάλο) $\wedge$ (μικρό $\vee$ μεσαίο $\vee$ μεγάλο)

·κύβος $\oplus$ δωδεκάεδρο $\oplus$ τετράεδρο = ($\neg\kappa \vee \neg\delta \vee \tau$) $\wedge$ ($\neg\kappa \vee \delta \vee \neg\tau$) $\wedge$ ($\kappa \vee \neg\delta \vee \neg\tau$) $\wedge$ ($\kappa \vee \delta \vee \tau$)

·μεσαίο $\iff$ δωδεκάεδρο = ($\neg$μεσαίο $\vee$δωδεκ) $\wedge$ (μεσαίο $\vee\neg$δωδεκ)

·μεγάλο $\iff$ τετρ = ($\neg$μεγάλο $\vee$ τετρ) $\wedge$ (μεγάλο $\vee\neg$τετρ)

· Άρνηση στόχου: $\neg$(μικρό $\iff$ κύβος) = ($\neg$μικρό $\vee\neg$κύβος) $\wedge$ (μικρό $\vee$κύβος)
Extracting the clauses from above

1.($\neg$μικρό $\vee\neg$ μεσαίο $\vee$ μεγάλο)
2.($\neg$μικρό $\vee$ μεσαίο $\vee\neg$ μεγάλο)
3.(μικρό $\vee\neg$ μεσαίο $\vee\neg$ μεγάλο)
4.(μικρό $\vee$ μεσαίο $\vee$ μεγάλο)
5.($\neg\kappa \vee \neg\delta \vee \tau$)
6.($\neg\kappa \vee \delta \vee \neg\tau$)
7.($\kappa \vee \neg\delta \vee \neg\tau$)
8.($\kappa \vee \delta \vee \tau$)
9.($\neg$μεσαίο $\vee$δωδεκ)
10.(μεσαίο $\vee\neg$δωδεκ)
11.($\neg$μεγάλο $\vee$ τετρ)
12.(μεγάλο $\vee\neg$τετρ)
13.($\neg$μικρό $\vee\neg$κύβος)
14.(μικρό $\vee$κύβος)
Συμπεράσματα:
15. Από Resolve(1,10) ($\neg$μικρό $\vee\neg\ \delta\ \vee$ μεγάλο)
16. Από Resolve(15,11) ($\neg$μικρό $\vee\neg\ \delta\ \vee\ \tau$)

17. Από Resolve(16,7) (¬μικρό $\vee \neg$ δ $\vee$ κύβος)
18. Από Resolve(2,12) (¬μικρό $\vee$ μεσαίο $\vee \neg$ τ)
19. Από Resolve(18,9) (¬μικρό $\vee$ δ $\vee \neg$ τ)
20. Από Resolve(19,8) (¬μικρό $\vee$ δ $\vee$ κύβος)

Ακολουθώντας τις ίδιες τακτικές με παραπάνω μπορούμε να εξάγουμε ακόμα τα συμπεράσματα:

21. (μικρό $\vee \neg$ δ $\vee \neg$ κύβος)
22. (μικρό $\vee$ δ $\vee \neg$ κύβος)
23. (μικρό $\vee \neg$ κύβος) από Resolve(21,22)
24. (¬μικρό $\vee$ κύβος) από Resolve(17,20)
25. (¬μικρό) από Resolve(13,24)
26. (μικρό) από Resolve(14,23)
27. $\varnothing$ ή Αντίφαση από το Resolve(25,26).

7. Ασκηση 7

Γνωρίζουμε ότι δύο προτάσεις ϕ, ψ είναι ισοδύναμες αν ισχύει ότι $(\phi \vDash \psi) \wedge (\psi \vDash \phi)$
Επίσης γνωρίζουμε ότι αν δύο προτάσεις ϕ, ψ είναι ισοδύναμες ανν $\phi \iff \psi$ είναι έγκυρη. Λόγω του ανν ισχύει ότι: $\phi \equiv \psi \Rightarrow \phi \iff \psi$ είναι έγκυρη.
Έχουμε τώρα τις προτάσεις $\phi = \neg(A \iff B), \psi = \neg((A \wedge B) \vee (\neg A \wedge \neg B))$
Αρχικά θα δείξουμε ότι $\phi \vDash \psi$, το οποίο για να το αποδείξουμε αρκεί να αποδείξουμε ότι η πρόταση $\phi \wedge \neg\psi = \varnothing$.
Έχουμε λοιπόν ότι
$\cdot KB = \phi = \neg(A \iff B)$
$\cdot$ Στόχος $= a = \psi = \neg((A \wedge B) \vee (\neg A \wedge \neg B))$
$\cdot$ Άρνηση στόχου $= \neg a = ((A \wedge B) \vee (\neg A \wedge \neg B))$

CNF

$\cdot KB = \phi_{CNF} = (\neg A \vee \neg B) \wedge (A \vee B)$
$\cdot$ Άρνηση στόχου $= \neg a = \neg\psi_{CNF} = ((\neg A \vee B) \wedge (\ A \vee \neg B))$
Άρα έχουμε ότι $Q = KB \wedge$ Άρνηση στόχου $= (\neg A \vee \neg B) \wedge (A \vee B) \wedge (\neg A \vee B) \wedge (\ A \vee \neg B)$

Εξαγωγή clauses από το Q:

1. $(\neg A \vee \neg B)$
2. $(A \vee B)$
3. $(\neg A \vee B)$
4. $(\ A \vee \neg B)$

Συμπεράσματα

5. $\neg A$ - Resolve(1,3)
6. A - Resolve(2,4)
7. $\varnothing$ ή Αντίφαση - Resolve(5,6)

Άρα αποδείξαμε ότι $\phi \vDash \psi$. Επαναλαμβάνουμε τώρα για να αποδείξουμε ότι $\psi \vDash \phi$.

Έχουμε λοιπόν ότι
$\cdot KB = \psi = \neg((A \wedge B) \vee (\neg A \wedge \neg B))$
$\cdot$ Στόχος $= a = \phi = \neg(A \iff B)$
$\cdot$ Άρνηση στόχου $= \neg a = (A \iff B)$

CNF

$\cdot KB = \psi_{CNF} = (\neg A \vee \neg B) \wedge (A \vee B)$
$\cdot$ Άρνηση στόχου $= \neg a = \neg\phi_{CNF} = ((\neg A \vee B) \wedge (\ A \vee \neg B))$
Άρα έχουμε ότι $Q = KB \wedge$ Άρνηση στόχου $= (\neg A \vee \neg B) \wedge (A \vee B) \wedge (\neg A \vee B) \wedge (\ A \vee \neg B)$
Εξαγωγή clauses από το Q:

1. $(\neg A \vee \neg B)$
2. $(A \vee B)$
3. $(\neg A \vee B)$
4. $(\ A \vee \neg B)$

Συμπεράσματα

5. $\neg A$ - Resolve(1,3)
6. A - Resolve(2,4)
7. $\varnothing$ ή Αντίφαση - Resolve(5,6)

Άρα αποδείξαμε ότι $\psi \vDash \phi$.

Απο τα παραπάνω ισχύει ότι $\phi \equiv \psi$. Άρα η $\phi \iff \psi$ είναι έγκυρη.

8. README KenKen

Main
Στο αρχείο kenken.py υπάρχει ενσωματωμένη και η main.
Για να τρέξουμε το αρχείο χρειαζόμαστε 1 μόνο όρισμα, αυτό του αρχείου εισόδου το οποίο ωστόσο θα πρέπει να είναι formatted με τον σωστό τρόπο.
Ως είσοδος δίνεται ένα αρχείο κειμένου με συγκεκριμένη μορφή, την οποία έχω αναγράψει και σε σχόλια. Η πρώτη γραμμή συνιστά το μέγεθος του παιχνιδιού που θα παίξουμε, ενώ από κάτω έχουμε γραμμές στις οποίες πρώτα αναγράφεται ο στόχος του γκρούπ κελιών, έπειτα έχουμε τον ειδικό χαρακτήρα #, έπειτα έχουμε τους αριθμούς των κελιών που ανήκουν στο συγκεκριμένο γκρούπ κελιών seperated με την χρήση παύλας (¨-¨), έπειτα πάλι τον ειδικό χαρακτήρα # και τέλος την πράξη που πρέπει να γίνει για το γκρούπ. Η ιδέα αυτή ήρθε από το piazza.
Αφού διαβάσω το αρχείο κρατάω την πρώτη γραμμή ως το μέγεθος και τις άλλες γραμμές ως τα γκρούπ κελιών και τα δίνω ως είσοδο στο αντικείμενο του KenKen.
Ποιοί αλγόριθμοι ακολουθούνται·

Οι αλγόριθμοι προς επίλυση τοποθετούνται ως ακέραιοι στον πίνακα choose. Οι ακέραιοι αναφέρονται στους παρακάτω αλγορίθμους:
$\cdot 1 \mapsto$ BT
$\cdot 2 \mapsto$ FC
$\cdot 3 \mapsto$ MAC
$\cdot 4 \mapsto$ BT + MRV
$\cdot 5 \mapsto$ FC + MRV
$\cdot 6 \mapsto$ MAC + MRV
$\cdot 7 \mapsto$ MIN-CONFLICTS

Για παράδειγμα ο πίνακας choose = [7] θα εκτελέσει μόνο τον min conflicts, ο choose = [2,3,5,6] θα εκτελέσει τους FC,MAC,FC+MRV,MAC+MRV,ενώ ο choose = [1,2,3,4,5,6,7] θα τους εκτελέσει όλους.

Επίσης οι χρόνοι βγαίνουν καταγράφονται σε αρχείο με όνομα: results.txt το οποίο δημιουργείται εκείνη την στιγμή.

KenKen class
Για το KenKen ακολούθησα σε μεγάλο βαθμό την μοντελοποίηση που περιέγραψα στο (1) αλλά άλλαξα μερικά πράγματα τα οποία μου φάνηκαν πιο όμορφα στην σχεδίαση. Έτσι, αντί να μοντελοποιήσω ως μεταβλητή το κάθε κελί από μόνο του, αποφάσισα ως μεταβλητή να μοντελοποιήσω το κάθε **γκρούπ κελιών**. Επίσης όρισα την κλάση να κάνει inherit από την κλάση csp.CSP αφού και το KenKen είναι ενα πρόβλημα ικανοποίησης περιορισμών.

KenKen:: init (size,lines)

Τα size, lines δίνονται από τον χρήστη όπως αναφέρεται παραπάνω.

Επειδή ο αριθμός του κάθε κελιού είναι αύξοντας, χρησιμοποιώ την διαίρεση και το υπόλοιπο της διαίρεσης με το μέγεθος του grid για να λάβω τις συντεταγμένες σε μορφή (x, y).

Για παράδειγμα, ας κοιτάξουμε το αρχείο .Έχουμε ένα grid με μέγεθος 4.

Το πρώτο γκρούπ (έστω με νούμερο 0) αποτελείται μονάχα από 1 κελί με αριθμό 0. Αυτό αντιστοιχείται στο $(x, y) = (0, 0)$. Έπειτα σώζω τις δύο πληροφορίες αφού θα μου είναι χρήσιμες μετά. Συγκεκριμένα, σώζω ότι το κελί $(0, 0)$ είναι μέλος του γκρούπ 0 (self.cage[(x,y)] = i), και ότι το γκρούπ 0 έχει μέσα το $(0, 0)$ (blocks[i].append((x,y))).

Ομοίως, το δεύτερο γκρούπ (με νούμερο 1) αποτελείται από τα νούμερα κελιών: 1,2,6. Αυτά αντιστοιχούνται στα $(x, y) = (0, 1), (x, y) = (0, 2), (x, y) = (1, 2)$. Έπειτα σώζω τις δύο πληροφορίες αφού θα μου είναι χρήσιμες μετά. Συγκεκριμένα, σώζω ότι τα κελιά $(0, 1), (0, 2), (1, 2)$ είναι μέλος του γκρούπ 1 (self.cage[(x,y)] = i), και ότι το γκρούπ 1 έχει μέσα το $(0, 1), (0, 2), (1, 2)$ (blocks[i].append((x,y))). Ομοίως και για τα υπόλοιπα.

Επίσης για το κάθε γκρούπ σώζω και πληροφορία σχετικά με την πράξη ΚΑΙ το στόχο. Η πληροφορία αυτή σώζεται στο cageGoal[i] = (int(val), op). Η λογική είναι ίδια με παραπάνω με το πρώτο γκρούπ να σώζει το (3,=), το δεύτερο να σώζει το (24,*) κ.ο.κ.

Τώρα ήρθε η ώρα για να δώσουμε το domain, το οποίο είναι ίσως το πιο δύσκολο κομμάτι της δεδομένης υλοποίησης. Αφού δεν έχουμε το κάθε κελί να είναι η κάθε μεταβλητή, το domain δεν είναι πλέον απλά μια λίστα από το 1 μέχρι το n. Το κάθε γκρούπ αποτελείται από πολλά κελιά. Έτσι, αρχικά λαμβάνω το πόσα κελιά έχει το κάθε γκρούπ κελιών (numOfBlocks[i] = len(blocks[i])). Έπειτα θα πρέπει να λάβω το εσωτερικό/καρτεσιανό γινόμενο για να δω όλες τις πιθανές επιλογές για τις μεταβλητές/κελιά εντός των γκρούπ, δηλαδή όλους τους δυνατούς συνδυασμούς τιμών. Με ένα απλό google search βρήκα ότι η συνάρτηση η οποία μπορεί να το κάνει αυτό είναι η iterools. Στο documentation υπάρχει ακριβώς αυτό που ζητάμε εμείς, με την χρήση του repeat, το οποίο θα μου δώσει το επιθυμητό καρτεσιανό γινόμενο του domain των διαφορετιών κελιών όσες φορές όσα και τα κελιά ΕΝΤΟΣ του γκρούπ. (perms= [perm for perm in itertools.product(x, repeat=len(blocks[i]))]).

Για παράδειγμα. Το κάθε κελί μπορεί να λάβει τις τιμές από 1 μέχρι n (x = [n for n in range(1,n+1)]). Έστω ότι έχουμε ένα γκρουπ με 2 κελιά, και έστω $n = 4$. Τότε όλες οι πιθανές επιλογές τιμών για το γκρούπ που έχουμε είναι το εσωτερικό γινόμενο του:
$[1,2,3,4] \times [1,2,3,4] = [(1,1),(1,2),(1,3),(1,4),$
$(2,1),(2,2),(2,3),(2,4),$
$(3,1),(3,2),(3,3),(3,4),$
$(4,1),(4,2),(4,3),(4,4)]$
Τώρα από όλες αυτές τις δυνατές διαφορετικές τιμές, αρκεί να δούμε ποιές μπορούν όντως να συνιστούν ένα δυνατό assignment με βάσει και τα operations,goal values. Για να το κάνω αυτό , καλώ την συνάρτηση satAssignments, η οποία λαμβάνει το ποιος είναι ο στόχος και ποιο το operation του γκρούπ/μπλόκ, και δοκιμάζει με βάσει αυτά αν υπάρχει κάποιο permutation των assigned values ώστε να τηρούνται οι περιορισμοί. Τέλος επιστρέφει μια λίστα με όλα τα ζεύγη για όλες τις μεταβλητές που να τηρούν τους περιορισμούς. Προφανώς πολλά ζεύγη αριθμήσεων μπορεί να ικανοποιούν τους περιορισμούς του μπλόκ. Αυτή η λίστα είναι και το domain του κάθε γκρούπ κελιών.

Τέλος καλούμαστε να δώσουμε και τους neighbours του κάθε γκρούπ κελιών. Για να το κάνω αυτό περνάω από το κάθε γκρούπ και για κάθε κελί του γκρούπ αυτού: κάνω ένα sweep όσα και τα κελιά και ελέγχω αν υπάρχει στην ίδια γραμμή ή στήλη ένα άλλο κελί που ΔΕΝ ανήκει στο ίδιο γκρούπ με αυτό που εμείς ελέγχουμε τώρα (self.cage[(i,y)] != cageIndex, self.cage[(x,i)] != cageIndex) και το γκρούπ αυτού δεν είναι ήδη στους γείτονες (self.cage[(i,y)] not in self.neighbours[cageIndex],self.cage[(x,i)] not in self.neighbours[cageIndex]). Τέλος αρχικοποίησα το dictionary με κενές λίστες στην αρχή για να μην υπάρξουν προβλήματα με το not in.

KenKen::kenken constaints

Με όλη την παραπάνω δουλειά, η συνάρτηση για τα contraints είναι σχετικά απλή. Α και Β είναι τα μπλοκ/γκρούπ κελιών και α,β είναι δύο αντίστοιχες λύσεις/assignments για τα κελιά εντός των 2 μπλοκ αντίστοιχα.
Χρησιμοποιώ ένα κενό dictionary: assignment το οποίο αποθηκεύει τις τιμές που έχει λάβει το κάθε κελί. Αυτό γίνεται εύκολα με 2 for loops, λαμβάνοντας τις συνεταγμένες του κελιού στο (x, y) και έπειτα πηγαίνοντας στην αντίστοιχη θέση του assignment (a[i]) και αποθηκεύοντας στο dictionary.
Τέλος χρησιμοποιώ και πάλι την συνάρτηση itertools.product για να πάρω όλους τους συνδυασμούς σημείων μέσα στα μπλοκς/γκρουπ κελιών. Παιρνώντας τώρα από αυτά, κοιτάμε να ικανοποιείται ότι: 2 σημεία που είναι στην ίδια γραμμή ή στήλη, ΔΕΝ πρέπει να έχουν ίδια τιμή. Αν ισχύει αυτό καταστρατηγείται το contraint και επομένως πρέπει να βγάλουμε False. Αν περάσουμε από τον έλεγχο αυτό σημαίνει ότι οι ανατεθειμένες τιμές στα κελιά καλύπτουν τους περιορισμούς.

KenKen::assertSolution

Η συνάρτηση αυτή χρησιμοποιείται για να σιγουρευτούμε η λύση μας είναι σωστή. Η λύση πρέπει να είναι σε αρχείο με όνομα ίδιο με της εισόδου με την προσθήκη του -Solution. Μέσα πρέπει να είναι η λύση με human readable format, με τα στοιχεία την κάθε σειράς να είναι space seperated και η αλλαγή σειράς με ένα απλό new line.