Τεχνητή Νοημοσύνη
(Project 3)

Ονοματεπώνυμο: Δημήτριος Σιταράς
Αριθμός μητρώου: 1115201800178
Εξάμηνο: 5ο

Πρόβλημα 1

		11	2-f24	2-f25	3-f10	3-f11	8-f10	8-f11	14-f27	14-f28	6-w2	7-w1-f4
Forward Checking	**Time**	0m5.584s	0m0.596s	0m38.617s	0m10.712s	1m53.910s	-	2m26.581s	-	1m26.083s	0m0.507s	0m4.725s
	Visited Nodes	6316	937	135607	68841	563197	-	506070	-	103387	683	46395
	Checks	1203335	100754	22428615	5904239	95850540	-	61216543	-	5805381	72182	1721153
MAC	**Time**	0m6.833s	0m0.561s	1m7.499s	0m1.542s	1m29.770s	0m34.816s	1m9.180s	0m8.688s	0m22.541s	0m0.579s	0m0.639s
	Visited Nodes	2956	200	28322	776	22089	17487	30663	14185	22043	42	479
	Checks	6401738	158995	66044241	1167548	107331420	33106552	73692412	4103679	13098591	397482	342291
FC-CBJ	**Time**	0m10.080s	0m0.452s	0m1.995s	0m9.677s	2m1.688s	3m26.172s	1m15.880s	3m4.582s	0m11.676s	0m0.542s	0m13.416s
	Visited Nodes	6316	265	3525	25262	378954	141041	135444	30728	5563	642	10745
	Checks	1203335	21577	555494	3320694	59137092	14780817	27170626	1661822	337953	72220	427492
Min Conflicts	**Time**	10m0.926s	3m8.090s	-	6m13.906s	-	-	-	-	-	2m19.465s	-
	Visited Nodes	100680	100200	-	100400	-	-	-	-	-	100200	-

Εκτέλεση του προγράμματος

Ο φάκελος με τα στιγμιότυπα (“rlfap”) πρέπει να βρίσκεται στον ίδιο φάκελο με το rlfa.py.
Εντολή εκτελεσης: ***python3 rlfa.py “instance” “algorithm”***
βάζοντας οπου “instance” το αντίστοιχο όνομα του στιγμιότυπου και όπου algorithm το αντιστοιχο όνομα του αλγορίθμου που θέλουμε.
Παρακάτω δίνω μερικά παραδείγματα εκτελέσεων:
python3 rlfa.py 11 fc ,
python3 rlfa.py 6-w2 mac ,
python3 rlfa.py 2-f10 fc-cbj,
python3 rlfa.py 14-f28 min-con

Παρατηρήσεις:

- Παρατηρώ συγκριτικά πως για το πλήθος κόμβων που επισκέπτεται ο κάθε αλγόριθμος ισχύει ότι:

  FC-CBJ ≤ FC και MAC≤ FC

- Επίσης, για το πλήθος ελέγχων συνέπειας παρατηρώ πως ισχύει ότι:

  FC-CBJ ≤ FC

- Σε οποια instances/στιγμιότυπα έχω βάλει “παύλα” ο αντίστοιχος αλγόριθμος έτρεχε παραπάνω απο 15 λεπτά οπότε διέκοψα την εκτέλεση του προγράμματος (χωρίς βεβαια να πάρω κάποιο αποτέλεσμα/λύση στο αντίστοιχο πρόβλημα).

- Τα κριτήρια για την αξιολόγηση των αλγορίθμων που χρησιμοποιήθηκαν είναι ο χρόνος εκτέλεσης τους, ο πλήθος κόμβος που επισκέπτονται και το πλήθος ελέγχων συνέπειας που πραγματοποιούν για κάθε στιγμιότυπο που δόθηκε (η χρονική/χωρική πολυπλοκότητα χείριστης περίπτωσης είναι εκθετική και είναι ίδια σε όλους του αλγορίθμους).

- Οι αλγόριθμοι FC, MAC και FC-CBJ σε συνδυασμό με την ευρετική dom/wdeg δεν περιέχουν κάποιον παράγοντα τύχης, επομένως τα visited nodes και τα checks ενός οποιουδήποτε instance δεν διαφέρουν απο εκτέλεση σε εκτέλεση.

Οι αλγόριθμοι FC και MAC ήταν έχουν ήδη υλοποιηθεί (προφανώς και ο αλγόριθμος backtracking που τις χρησιμοποιει) στον έτοιμο κώδικα που δίνεται από τον σύνδεσμο στην εκφώνηση. Συνεπώς, το μόνο που άλλαξα όσον αναφορά αυτούς τους αλγορίθμους είναι να προσθέσω στην συνάρτηση revise() (που χρησιμοποιει ο AC3 που καλείται απο τον mac) και στην forward_checking() “μια γραμμή κώδικα”, ώστε όταν οδηγείται σε αποτυχία να αυξάνεται το βάρος του αντίστοιχου περιορισμού κατά 1 (χρησιμοποιώ αυτα τα βάρη περιορισμών ώστε να υπολογίσω στη συνέχεια το σταθμισμένο βαθμό της κάθε μεταβλητής που χρειάζομαι στην ευρετική dom/wdeg που υλοποίησα).

domdivwdeg(...)

Υλοποίησα την ευρετική dom/wdeg που προτείνεται να χρησιμοποιηθεί με βάση την ιστοσελίδα που δίνεται επίσης στην εκφώνηση. Συγκεκριμένα, η ευρετική αυτή επιστρέφει κάθε φορά την μεταβλητή που εχει την μικρότερη αναλογιά (αποτελέσμα) dom/wdeg. Επομένως, για κάθε μεταβλητή που δεν της έχει ανατεθεί ακόμα τιμή υπολογίζω το σταθμισμένο βαθμό της προσθέτοντας τα βάρη των περιορισμών που συμμετέχει με το οποίο διαιρώ το πλήθος των διαθέσιμων τιμων που μπορούν να ανατεθούν στην αντίστοιχη μεταβλητή. Οπότε, τελικά επιστρεφώ την μεταβλητή που έχει τό μικροτερο πυλίκο διαίρεσης.

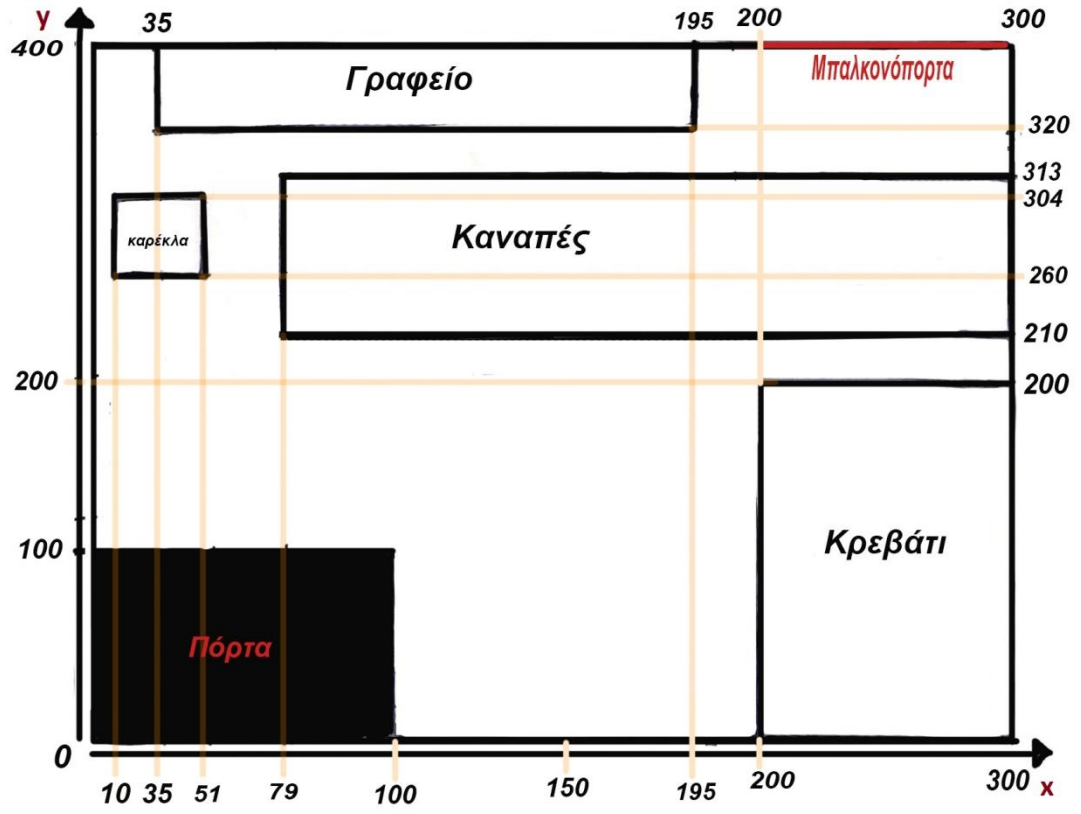
FC-CBJ

Ο αλγόριθμος του Forward Checking είναι ετοιμος στον κώδικα που δίνεται, οπότε υλοποίησα μόνο τον αλγόριθμο του Conflict-Directed Backjumping (cbj). Ουσιαστικα, βασίστηκα στον αλγόριθμο του backtrack προσθέτοντας οποια στοιχεία χρειάζονται από τις διαφάνειες. Συγκεκριμένα, έχω ένα σύνολο συγκρούσεων για κάθε μεταβλητή το οποίο ανανεωνεται για κάθε νέα ανάθεση που γίνεται. Οπότε, όταν δεν υπάρχουν αλλες τιμές προς δοκιμή για μια μεταβλητή έστω Xi (dead-end, τοτε την κραταω στην μεταβλητη last_variable) ο αλγόριθμος υπαναχωρεί προς την μεταβλητή εστω Xh του συνόλου συγκρούσεων της Xi που βρίσκεται βαθύτερα στο δεντρο ανάζητησης, ενώ παραλληλα οι μεταβλητές στο συνολο συγκρούσεων της Xi (εκτός απο την Xh) προστίθενται στο σύνολο συγκρούσεων της Xh (με την κλήση της συνάρτησης merging).

Min Conlficts

Ο αλγόριθμος τοπικης αναζήτησης MinConflicts δεν είναι τοσο αποδοτικος συγκριτικα με τους παραπανω αλγορίθμους κάθως οι λύσεις των περισσότερων στιγμιοτύπων δεν ειναι πυκνά κατανεμημένες στο χώρο καταστάσεων του κάθε στιγμιοτύπου. Επομένως, βρίσκει λύση σε σύντομο σχετικά χρόνο για τα στιγμιότυπα που ειναι “απλά και εύκολα”.

Πρόβλημα 2



Ορίζω ένα σύστημα συντεταγμένων, με αρχή των αξόνων την κάτω αριστερή γωνία του δωματίου.
Έτσι, το πρόβλημα ικανοποίησης περιορισμών (CSP) ορίζεται από:

~Το σύνολο μεταβλητων, δηλαδή τις: Sofa, Bed, Chair, Office
Κάθε μεταβλητή αναπαριστάνται απο 2 σημεία, το πρώτο σημείο εκφράζει την κάτω αριστερή γωνία του επιπλου και το δεύτερο σημέιο την πάνω δεξιά γωνία αντίστοιχα. Το δευτερο σημείο, δηλαδή η πάνω δεξιά γωνία εξαρτάται απο την κάτω αριστερή (για το δευτερο σημείο απλα προσθέτω τις διαστάσεις του αντίστοιχου επίπλου στις συντεταγμένες χ1,y1 που έχουν επιλεχθεί για το πρώτο σημείο), για παραδειγμα Sofa: (x1,y1) και (x2=x1+221,y2=y1+103).
Συνεπώς, μόνο οι συντεταγμένες του πρώτου σημειου (που αναπαριστά την κατω αριστερη γωνία) αρκει να παίρνουν τιμές απο ενα μή κενό πεδίο (domain) δυνατών τιμών, πιο συγκεκριμένα:

Sofa: D1 = {(0<=x1<=300-221, 0<=y1<=400-103)}
Bed: D2 = {(0<=x1<=300-100, 0<=y1<=400-200)}
Chair: D3 = {(0<=x1<=300-41, 0<=y1<=400-44)}
Office: D4 = {(0<=x1<=300-160, 0<=y1<=400-80)}

~Ένα σύνολο απο περιορισμούς, δηλαδή:
(κάτω αριστερή γωνία (x1,y1) και πάνω δεξιά (x2,y2))

Sofa: C1={ (Sofa.x1>Bed.x2 OR Sofa.x2<Bed.x1 OR Sofa.y1>Bed.y2 OR Sofa.y2<Bed.y1) AND (Sofa.x1>Chair.x2 OR Sofa.x2<Chair.x1 OR Sofa.y1>Chair.y2 OR Sofa.y2<Chair.y1) AND (Sofa.x1>Office.x2 OR Sofa.x2<Office.x1 OR Sofa.y1>Office.y2 OR Sofa.y2<Office.y1) AND (Sofa.x1>100 OR Sofa.y1>100) }

Bed: C2={ (Bed.x1>Sofa.x2 OR Bed.x2<Sofa.x1 OR Bed.y1>Sofa.y2 OR Bed.y2<Sofa.y1) AND (Bed.x1>Chair.x2 OR Bed.x2<Chair.x1 OR Bed.y1>Chair.y2 OR Bed.y2<Chair.y1) AND (Bed.x1>Office.x2 OR Bed.x2<Office.x1 OR Bed.y1>Office.y2 OR Bed.y2<Office.y1) AND (Bed.x1>100 OR Bed.y1>100) }

Chair: C3={ (Chair.x1>Bed.x2 OR Chair.x2<Bed.x1 OR Chair.y1>Bed.y2 OR Chair.y2<Bed.y1) AND (Chair.x1>Sofa.x2 OR Chair.x2<Sofa.x1 OR Chair.y1>Sofa.y2 OR Chair.y2<Sofa.y1) AND (Chair.x1>Office.x2 OR Chair.x2<Office.x1 OR Chair.y1>Office.y2 OR Chair.y2<Office.y1) AND (Chair.x1>100 OR Chair.y1>100) }

Office: C4={ ((Office.x1>Bed.x2 OR Office.x2<Bed.x1 OR Office.y1>Bed.y2 OR Office.y2<Bed.y1) AND (Office.x1>Chair.x2 OR Office.x2<Chair.x1 OR Office.y1>Chair.y2 OR Office.y2<Chair.y1) AND (Office.x1>Sofa.x2 OR Office.x2<Sofa.x1 OR Office.y1>Sofa.y2 OR Office.y2<Sofa.y1) AND (Office.x1>100 OR Office.y1>100)) AND
((Office.x2-200)^2 + (Office.y2-400)^2)^1/2 <= 10 }

Γενικότερα, για 2 (ορθογώνια) επιπλα για να ελέγξω αν μεταξύ τους εφάπτονται ή πατάνε το ενα πανω στο αλλο χρειάζομαι για κάθε έπιπλο μόνο 2 σημεία, ειτε της κατω αριστερης με της πανω δεξίας γωνίας είτε της πάνω αριστερής με της κάτω δεξιάς γωνίας.
Επίσης, το γραφείο για να ελέγξω αν ειναι δίπλα στην μπαλκονόπορτα (δηλαδή στην εισοδο φωτός του δωματίου) παίρνω την ευκλείδια απόσταση με σημεία την πάνω δεξια γωνία του γραφείου και το αριστερό σημείο της μπαλκονόπορτας (200,400). Για να ειναι το γραφείο διπλα στην μπαλκονόπορτα η μεταξύ τους απόσταση θα πρεπει να ειναι μικρότερη ή ίση απο 10 εκατοστά.

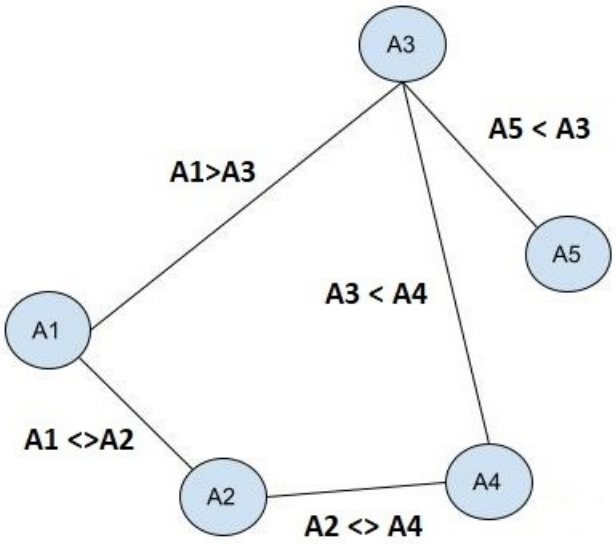
Πρόβλημα 3

1)
Το πρόβλημα ικανοποίησης περιορισμών (CSP) ορίζεται από:

~Ένα σύνολο μεταβλητων, δηλαδή τις: A1, A2, A3, A4, A5.
Κάθε μεταβλητή έχει ενα μή κενό πεδίο (domain) δυνατών τιμών, συγκεκριμένα έχω ως τιμές στο domain των μεταβλητών μόνο τις ώρες έναρξης (δίοτι κάθε ενέργεια για να ολοκληρωθεί χρειάζεται 60 λεπτά, δηλαδή 1 ώρα) :
A1: D1={9,10,11}
A2: D2={9,10,11}
A3: D3={9,10,11}
A4: D4={9,11}
A5: D5={9,10,11}

~Ένα σύνολο απο περιορισμούς, δηλαδή:
A1: C1={A1 > A3}
A2: C2={A2 <> A1 AND A2 <> A4}
A3: C3={A5 < A3 AND A3 < A4}
A4: C4={ }
A5: C5={ }

2)
Ο γράφος των περιορισμών του προβλήματος ειναι:



3)

Σειρά ανάθεσης τιμών: A1, A2, A3, A4, A5
Σειρά επιλογής τιμών: 9, 10 ,11

Αρχικά όλες οι ακμές ειναι συνεπείς

- **A1=9** -> D2={10,11} και D3={ } -> δοκιμάζω την επομενη τιμή της A1, D1={10,11}.

- **A1=10** -> D2={9,11} και D3={9} -> εξετάζω τις ακμές: (A2,A4), (A3,A4) και (A3,A5).

  (A2,A4) -> OK
  (A3,A4) -> OK
  (A3,A5) -> μη συνεπής -> D3={ } -> δοκιμάζω την επομενη τιμή της A1, D1={11}.

- **A1=11** -> D2={9,10} και D3={9,10} -> εξετάζω τις ακμές: (A2,A4), (A3,A4) και (A3,A5)

  (A2,A4) -> OK
  (A3,A4) -> OK
  (A3,A5) -> OK

- **A2=9** -> D4={11}

  (A4,A3) -> OK

- **A3=9** -> D4={11} και D5={ } -> δοκιμάζω την επομενη τιμή της A3, D3={10}.

- **A3=10** -> D4={11}, D5={9}

Άρα, η λύση ειναι: **A1=11, A2=9, A3=10, A4=11, A5=9**