ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

[PLH 311]-Τεχνητή Νοημοσύνη

1ο Σετ Θεωρητικών Ασκήσεων

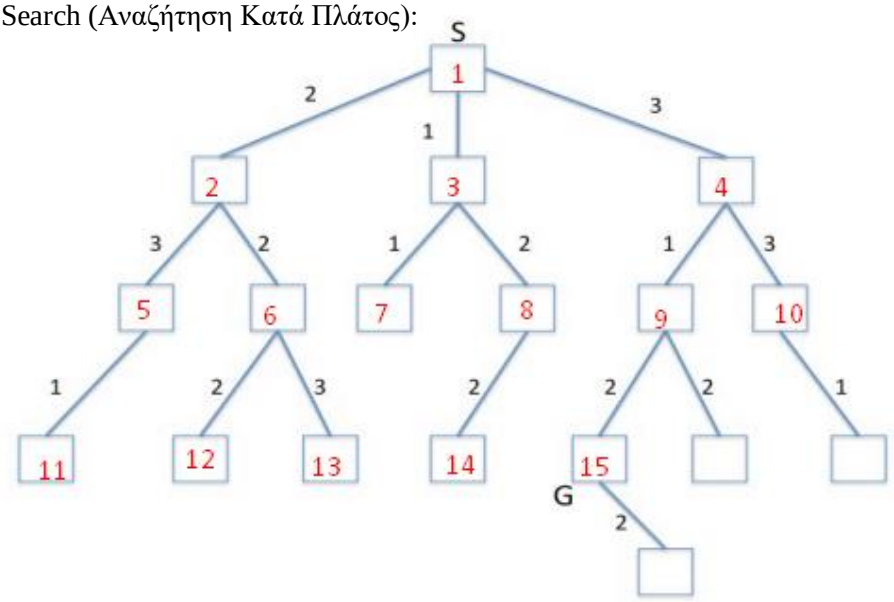
Ομάδα Χρηστών 193:
Ιωάννης Περίδης
A.M. 2018030069

5 Μαΐου 2022

Άσκηση (1):

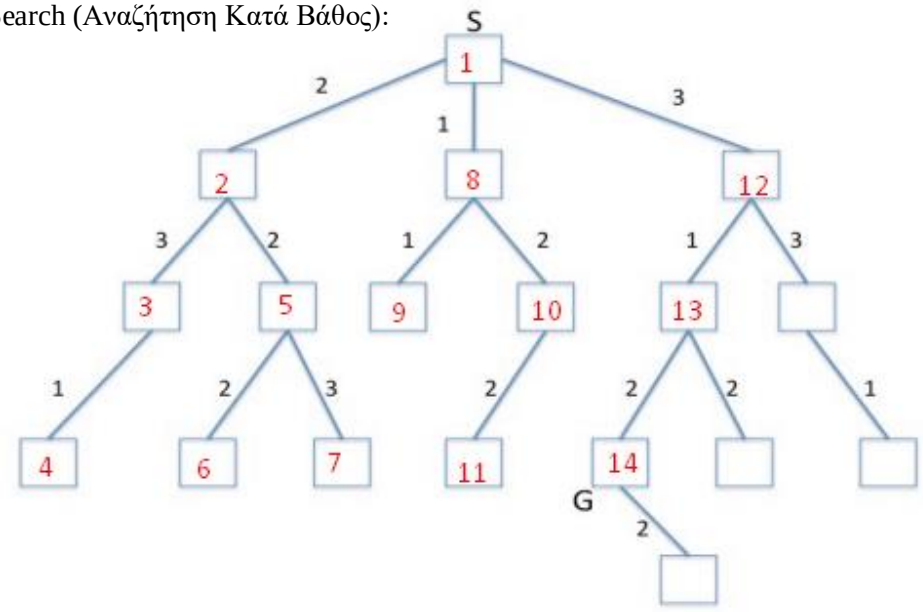
A)

Breadth-First Search (Αναζήτηση Κατά Πλάτος):



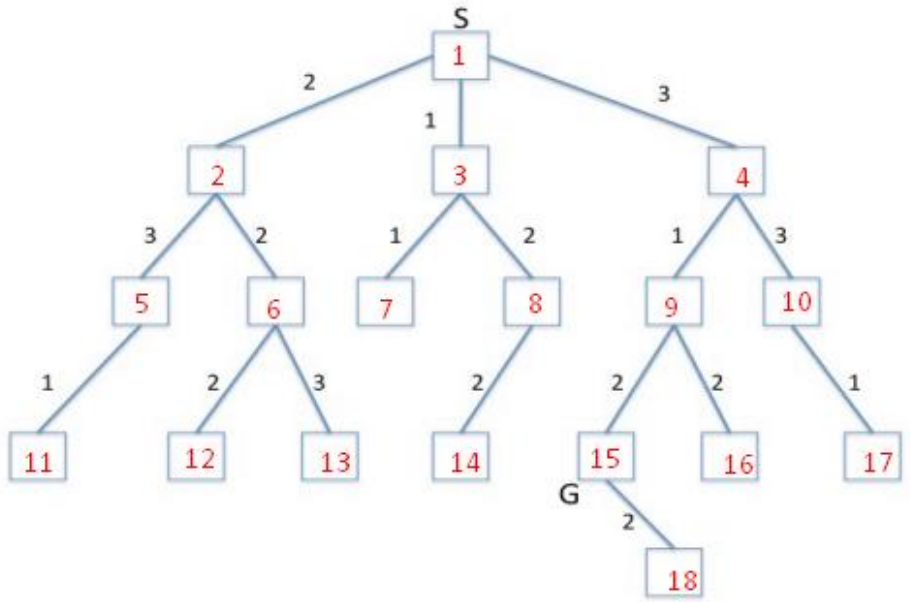
B)

Depth-First Search (Αναζήτηση Κατά Βάθος):



C)

Iterative Deepening Search (Αναζήτηση με Επαναληπτική Εκβάθυνση):



Παρακάτω φαίνονται οι διαδρομές (δηλαδή οι κόμβοι) που ακολουθήθηκαν , μαζί με το όριο βάθους:

Limit=0: (1)

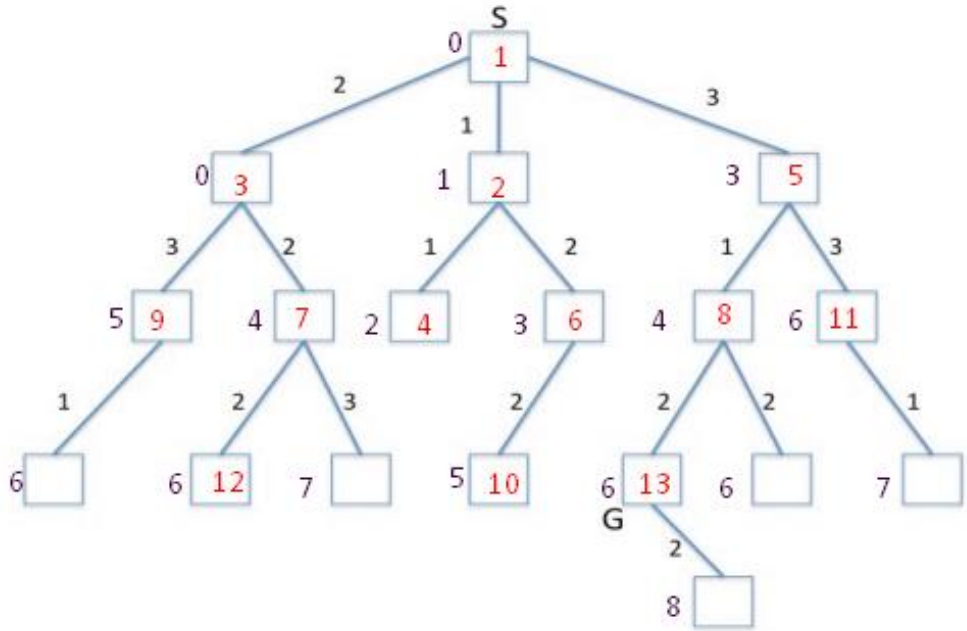
Limit=1: (1) →(2) →(3) →(4)

Limit=2: (1) →(2) →(5) →(6) →(3) →(7) →(8) →(4) →(9) →(10)

Limit=3: (1) →(2) →(5) →(11) →(6) →(12) →(13) →(3) →(7) →(8) →(14) →(4) →(9) →(15)

D)

Uniform-Cost Search (Αναζήτηση Ομοιόμορφου Κόστους):



Άσκηση (2):

Τι είναι το βραβείο Loebner:

Το βραβείο Loebner ήταν ένας ετήσιος διαγωνισμός τεχνητής νοημοσύνης που απονέμει βραβεία στα προγράμματα υπολογιστών που θεωρούν οι κριτές τα πιο ανθρώπινα. Η μορφή του διαγωνισμού βασίζεται στη δοκιμή Turing. Ένας ανθρώπινος κριτής διεξάγει συνομιλίες, μέσω ενός πληκτρολογίου, με ένα «ανθρωπομορφικό» πρόγραμμα υπολογιστή και με έναν άνθρωπο. Με βάση τις απαντήσεις, ο κριτής αποφασίζει ποιος είναι ποιος, και στον διαγωνισμό βραβείων Loebner, οι κριτές κατατάσσουν τις συμπεριφορές των προγραμμάτων αυτών τις πιο ανθρώπινες σε λιγότερο ανθρώπινες. Το πρόγραμμα υπολογιστή με την υψηλότερη μέση κατάταξη κερδίζει τον διαγωνισμό και απονέμεται ένα μετάλλιο και ένα χρηματικό έπαθλο.

Δύο από τους νικητές:

Το *πρώτο* chatrobot που μου έκανε εντύπωση είναι ο Do-Much-More. Το συγκεκριμένο chatrobot, αναπτύχθηκε από τον David Levy και νίκησε το βραβείο Loebner το έτος 2009. Το Do-Much-More έχει σχεδιαστεί έτσι ώστε να φαίνεται πιο φυσικό και πιο ενημερωμένο από άλλα chatbot, χαρακτηριστικά που γίνονται δυνατά από τη γνώση της μορφολογίας της αγγλικής γλώσσας, ορισμένες γενικές γνώσεις και κάποιες γνώσεις που σχετίζονται με τη χρήση λέξεων. Το πιο σημαντικό χαρακτηριστικό του Do-Much-More και αυτό που βρίσκει χρήση στην καθημερινή ζωή και για αυτό μου έκανε εντύπωση είναι η ψυχαγωγική του αξία. Διατηρώντας το ενδιαφέρον και την ψυχαγωγία του χρήστη καθ' όλη τη διάρκεια μιας συνομιλίας, ένας ιστότοπος φιλοξενίας θα μεγιστοποιεί τον χρόνο που αφιερώνει ο χρήστης στην επίσκεψη του στον ιστότοπό του. Η αξία ψυχαγωγίας που παρέχεται από τον ιστότοπο θα ενθαρρύνει επίσης πολλούς επισκέπτες να επιστρέφουν συχνά στον ιστότοπο. Ομοίως, το Do-Much-More μπορεί να ενσωματωθεί σε ορισμένα καταναλωτικά ηλεκτρονικά προϊόντα που θα γίνουν διασκεδαστικοί και ενδιαφέροντες συνομιλητές.

Το *δεύτερο* chatrobot που μου έκανε εντύπωση και πιθανότατα να είναι έως τώρα ένα από τα πιο εξελιγμένα προγράμματα συνομιλίας που υπάρχουν είναι το Kuki AI ή αλλιώς ευρέως γνωστό σαν Mitsuku. Το Mitsuku, αναπτύχθηκε από τον Steve Worswick και έχει νικήσει πολλαπλές χρονιές το βραβείο Loebner, ξεκινώντας από το 2013 και ύστερα συνεχίζοντας με τις χρονιές 2016 έως και 2019 σερί νικών (Guiness wors record!). Φυσικά το Mitsuku κάθε χρονιά ερχόταν ακόμα πιο εξελιγμένο και με καινούργια χαρακτηριστικά. Το Mitsuku είναι ένα απίστευτα ανθρωποειδές chatbot που μιμείται την προσωπικότητα μιας 18χρονης γυναίκας στη Βόρεια Αγγλία. Οποιοσδήποτε έχει σύνδεση στο διαδίκτυο μπορεί ελεύθερα να συνομιλήσει με τη Mitsuku και να τη ρωτήσει οτιδήποτε και θα απαντήσει. Κάνει καλή δουλειά μιμούμενος τις ανθρώπινες απαντήσεις, αναγνωρίζοντας σύγχρονες καθομιλούμενες, συντομογραφίες υπολογιστών όπως το LOL και τρέχοντα γεγονότα .Ακόμη, η νοημοσύνη της περιλαμβάνει την ικανότητα να συλλογίζεται με συγκεκριμένα αντικείμενα. Τέλος, μπορεί ακόμα και να παίζει παιχνίδια και να κάνει μαγικά κόλπα κατόπιν αιτήματος του χρήστη! Είναι συναρπαστικό πως ακόμα και σήμερα το Mitsuku ανταλλάσσει εκατομμύρια μηνύματα με χρήστες από όλον τον κόσμο, πολλοί εκ των οποίων είναι πλέον τακτικοί. Παρόλα αυτά, ενώ το Mitsuku βελτιώνεται χρόνο με το χρόνο, εξακολουθούν να υπάρχουν αναγνωρίσιμες διαφορές μεταξύ του chatbot και ενός ανθρώπου στην άλλη άκρη του κειμένου. Η Mitsuku δυσκολεύεται να αναγνωρίσει κοινά ορθογραφικά λάθη ή γράμματα και οι συνομιλίες έχουν ένα πεπερασμένο τελικό σημείο ακόμα κι αν το άτομο που αλληλεπιδρά μαζί της προσπαθήσει να συνεχίσει τη συνομιλία

Άσκηση (3):

Δίνεται συνάρτηση $\boldsymbol{f(n) = a * \big(g(n) + \varepsilon(n)\big) = a * g(n) + a * \varepsilon(n)}$,

$\boldsymbol{με\ a(n) \in (0,1]\ και\ \varepsilon(n)}$ παραδεκτή ευρετική συνάρτηση.

Παρατηρώ πως ο αλγόριθμος που δίνεται από την παραπάνω συνάρτηση, έχει ίδια λειτουργία με τον γνωστό αλγόριθμο A*. Η συνάρτηση του A* είναι $\boldsymbol{f(n) = g(n) + h(n)}$,

όπου $\boldsymbol{g(n)}$ = πραγματικό κόστος από αρχή έως κόμβο $\boldsymbol{n}$ και

$\boldsymbol{h(n)}$ = εκτιμώμενο κόστος φθηνότερης διαδρομής απο κόμβο $\boldsymbol{n}$ σε στόχο.

Έχω την αντιστοιχία $\boldsymbol{f(n) = g'(n) + \varepsilon'(n) = a * g(n) + a * \varepsilon(n)}$, (προφανώς $\boldsymbol{\varepsilon(n) = h(n)}$) .

Και αφού η ε είναι παραδεκτή και το α θετικό , τότε $\boldsymbol{\varepsilon(n) \leq g(n) \rightarrow a * \varepsilon(n) \leq a * g(n) \rightarrow \varepsilon'(n) \leq g'(n)}$.

Άρα και η ε' είναι επίσης παραδεκτή . Από θεωρία γνωρίζω πως αν ο A* έχει h(n) παραδεκτή τότε είναι βέλτιστος. Συμπερασματικά και ο αλγόριθμος της συνάρτησης που δίνεται θα είναι βέλτιστος.

Άσκηση (4):

Αρχικά, θα αποδειχθεί πως οι j1(n) και j2(n) είναι παραδεκτές. Γνωρίζω πως j1(n)=min(hi) και j2(n)=max(hi), επομένως είναι φανερό πως όλες οι τιμές που θα παίρνουν θα βρίσκονται εντός των παραδεκτών τιμών ,αφού j1(n) και j2(n) ≤ hmax. Επομένως , θα είναι και αυτές παραδεκτές.

Όσον αφορά την j3(n) γνωρίζω πως $\boldsymbol{j3(n) = \sum w_i h_i(n),\ με\ \sum w_i = w1 + w2 + w3 = 1\ και\ w_i \geq 0}$ Συνεπώς καταλαβαίνω πως $\boldsymbol{w_i \leq 1}$.

Έστω διαλέγω h1=hmax, τότε ισχύει για την j3(n):

$\boldsymbol{j3(n) = w1 * h1 + w2 * h2 + w3 * h3 \leq w1 * h1 + w2 * h2 + w1 * h1 = h1 * (w1 + w2 + w3) =}$

$\boldsymbol{= h1 * 1 = hmax}$ Άρα $\boldsymbol{j3(n) \leq hmax}$, δηλαδή και η j3(n) θα παίρνει μόνο τιμές που είναι παραδεκτές , άρα θα είναι και εκείνη παραδεκτή.

Στο σημείο αυτό, θα πρέπει να βρεθεί ποια από τις παραπάνω ευρετικές είναι η κυρίαρχη, δηλαδή ποια από αυτές έχει την μεγαλύτερη αποδοτικότητα , άρα και το καλύτερο εκτιμώμενο πραγματικό κόστος. Επόμενος επιλέγω εκείνη την ευρετική k που ικανοποιεί την σχέση : $\boldsymbol{jk(n) \geq j_i(n)\ για\ κάθε\ i}$.

Άμεσα απορρίπτω την j2(n) αφού είναι ίση με το ελάχιστο hmin. Ύστερα μεταξύ των άλλων 2 γνωρίζω πως

$\boldsymbol{j3(n) \leq hmax = j1(n)}$, επομένως η κυρίαρχη είναι η j1(n).

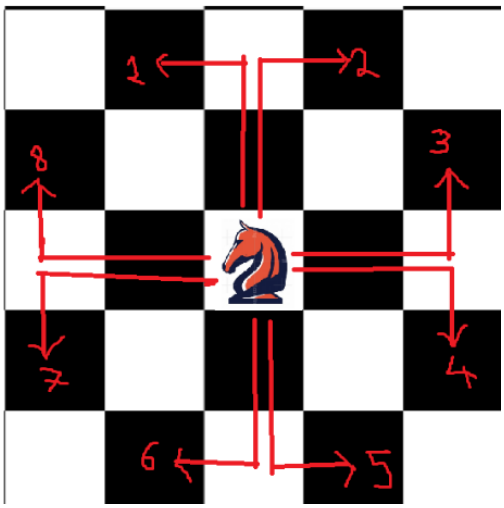
Άσκηση (5):

Η τοποθέτηση κ ίππων σε διαφορετικές θέσεις σε μια σκακιέρα nxn με τέτοιο τρόπο ώστε να μην υφίστανται απειλές, ανάγεται στα προβλήματα ικανοποίησης περιορισμών (Constraint Satisfaction Problems). Για την λύση , θα χρησιμοποιηθεί αλγόριθμος τοπικής αναζήτησης (Local Search), αφού διατυπωθούν και έχουν γίνει κατανοητοί όλοι οι περιορισμοί.

Συγκεκριμένα, για τους περιορισμούς, δεν γίνεται 2 ίπποι να βρίσκονται στο ίδιο τετράγωνο , δηλαδή

$\boldsymbol{Si \neq Sj, για\ κάθε\ i < j}$. Ακόμη, είναι φανερό πως οι ίπποι πρέπει να ακολουθούν τις κινήσεις που είναι επιτρεπτές βάση των κανονισμών που έχει το σκάκι («να κινούνται σε Γ»). Έστω σύνολο Α με όλες τις δυνατές κινήσεις αυτές θα είναι 8 (πρώτα 2 βήματα πάνω κάτω, δεξιά ή αριστερά και μετά ένα βήμα πάνω ,κάτω, δεξιά ή αριστερά , αναλόγως την κάθε περίπτωση) και θα είναι οι εξής παρακάτω:

$\boldsymbol{A = \{[2,1],[2,-1],[-2,1],[-2,-1],[1,2],[1,-2],[-1,2],[-1,-2]\}}$ *(πηγή υπάρχει στο τέλος)*

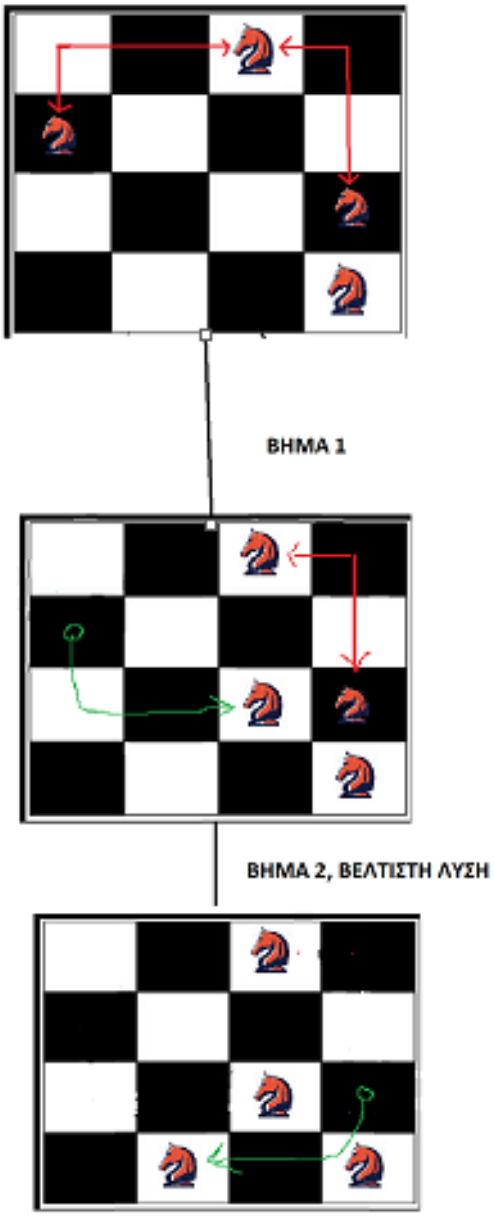


Τέλος, όλοι οι ίπποι θα πρέπει να είναι τοποθετημένοι έτσι ώστε να μην απειλείται κανείς από κανέναν, δηλαδή $\boldsymbol{για\ κάθε\ i < j, [xi, yi] \notin \{[xj + a, yj + b] | a, b \in A\}}$

Συνεχίζοντας , ορίζουμε την ευρετική συνάρτηση του κόστους να είναι ίση με τον αριθμό των ζευγών των ίππων που αλληλοαπειλούνται. Αναλυτικότερα, σε κάθε βήμα του αλγορίθμου, θα γίνεται η κίνηση με την οποία θα απαλειφθούν τα περισσότερα ζεύγη απειλούμενων ίππων. Αν υπάρχει ισοπαλία σε κάποιες κινήσεις, τότε επιλέγεται μια από αυτές στην τύχη. Το γεγονός αυτό , μπορεί ορισμένες φορές να μας παγιδεύσει σε κάποιο τοπικό ελάχιστο , με αποτέλεσμα να μην μπορέσουμε να βρούμε την βέλτιστη λύση, η οποία θα προερχόταν σε κάποιο επόμενο βήμα, αν επιλέγαμε (τυχαία) μια από τις άλλες ισοπαλίες. Αυτό συμβαίνει διότι δεν έχουμε backtracing και άρα δεν μπορούμε να γυρνάμε πίσω σε καταστάσεις που θα μας έδιναν καλύτερο αποτέλεσμα. Ο αλγόριθμος αυτός επαναλαμβάνεται μέχρι όλες οι γειτονικές καταστάσεις να έχουν τιμές ίσες ή χειρότερες με την τωρινή κατάσταση, δηλαδή να μην μπορεί να βελτιωθεί με καμία δυνατή κίνηση η συνάρτηση κόστους.

Παρακάτω δίνεται ένα παράδειγμα που μας δίνει μια βέλτιστη λύση:

Έστω σκακιέρα 4χ4 με 4 ίππους τοποθετημένους σε τυχαίες θέσεις. Παρατηρείται πως έχουμε 2 ζεύγη απειλούμενων ίππων, άρα η ευρετική του κόστους μας θα είναι ίση με h=2. Στην περίπτωση μας, υπάρχουν μόνο ισοπαλίες σε κινήσεις που μπορούμε να κάνουμε , δηλαδή μόνο κινήσεις που μειώνουν το κόστος από 2 σε 1, επομένως ο αλγόριθμος διαλέγει στην τύχη μια από αυτές και πλέον .Στο επόμενο βήμα θα έχω h=1 και αφού πραγματοποιηθεί και αυτό θα έχω βέλτιστη λύση , χωρίς κανένα απειλούμενο , άρα με h=0.

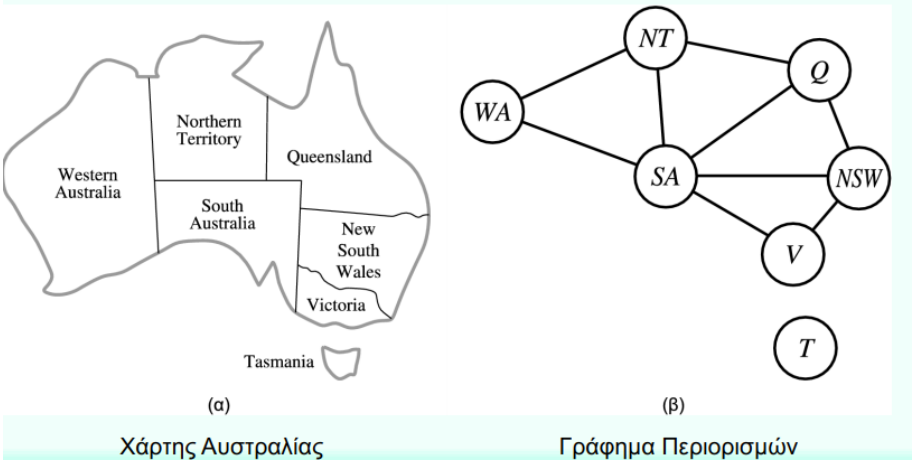


Είναι φανερό πως αφού τα άλογα αρχίζουν από τυχαίες τιμές και δεν γίνεται και backtracing, ο αλγόριθμος μας δεν θα βρίσκει πάντα λύση στο πρόβλημα , μπορεί να κολλάει σε τοπικά ελάχιστα. Σίγουρα θα το βελτιώνει όμως σε κάποια καλύτερη κατάσταση αν αυτή υπάρχει. Συμπερασματικά δεν μπορούμε να τον χαρακτηρίσουμε πλήρης.

Άσκηση (6):

Το ερώτημα με πόσους διαφορετικούς τρόπους μπορώ να χρωματίσω την Αυστραλία με μόνο 3 χρώματα και ορισμένους περιορισμούς ,ανάγεται στα προβλήματα ικανοποίησης περιορισμών (Constraint Satisfaction Problems). Για την λύση θα χρησιμοποιηθεί ένας αλγόριθμος backtracing, αφού έχουν γίνει κατανοητοί οι περιορισμοί. Συγκεκριμένα, η άσκηση ζητούσε να χρωματιστεί η Αυστραλία με τέτοιο τρόπο ώστε καμία περιοχή να μην είναι του ίδιο χρώματος με αυτά από όλους τους γείτονες τις (δηλαδή τις περιοχές που συνορεύει) και να γίνει αναπαράσταση ,ε απαρίθμηση επιτρεπτών συνδυασμών.

Παρακάτω φαίνεται η φωτογραφία του γράφου περιορισμού που χρησιμοποιήθηκε στην άσκηση, η φωτογραφία είναι από τις διαλέξεις του μαθήματος.

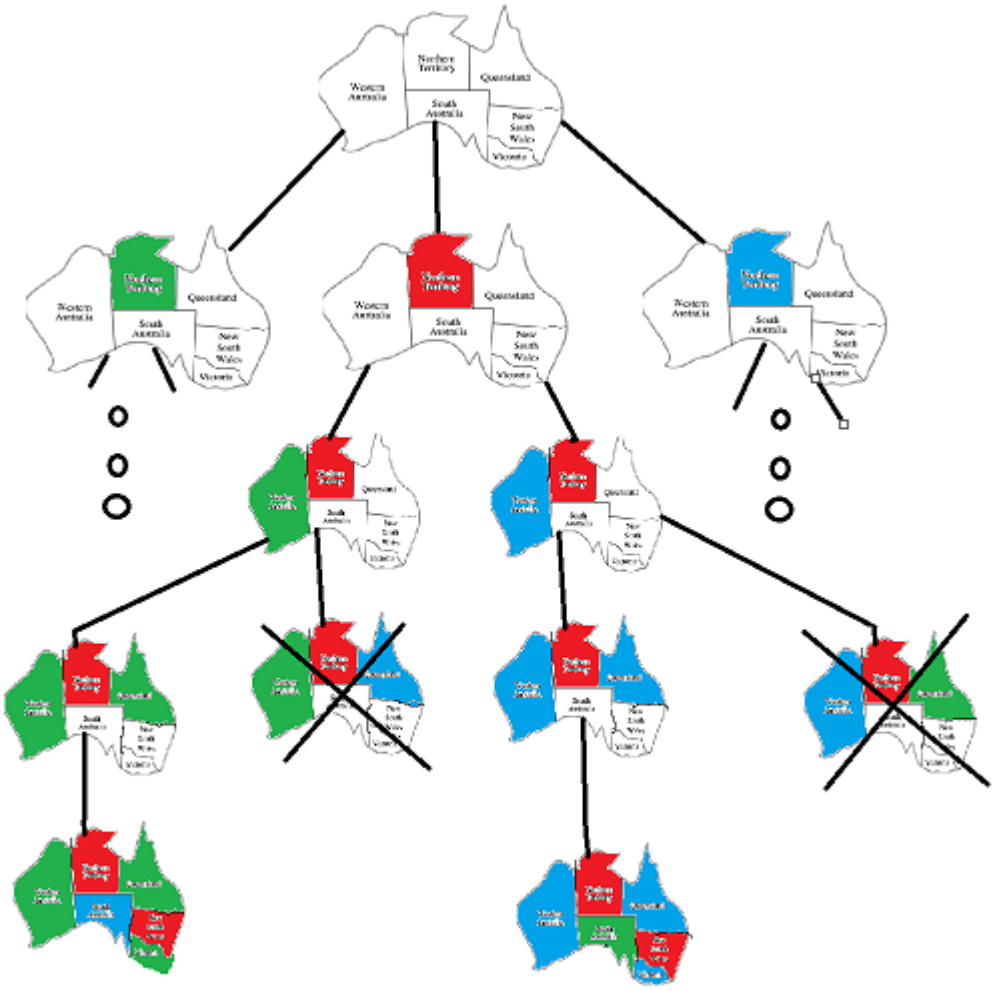


Χάρτης Αυστραλίας Γράφημα Περιορισμών

Οι μεταβλητές μας θα είναι οι περιοχές της Αυστραλίας : WA,NT,Q,NSW,V,SA,T ενώ τα πεδία τιμών μας θα είναι τα χρώματα {κόκκινο, πράσινο, μπλε} για όλες τις μεταβλητές.

Στο σημείο αυτό, πρέπει να σημειωθεί πως δεν μας ενδιαφέρει να δώσουμε όλες τις δυνατές λύσεις, δηλαδή δεν χρειάζεται να αναθέσουμε τιμές στις μεταβλητές μας, απλά πρέπει να βρούμε το πλήθος των δυνατών λύσεων.

Θα δημιουργήσουμε ένα δέντρο αποφάσεων . Για αρχή θα αφήσουμε έξω την περιοχή T, καθώς και αφού είναι αποκομμένη από τις άλλες, δεν μας επηρεάζει καθόλου στους χρωματικούς περιορισμούς μας (θα την συμπεριλάβουμε στο τέλος). Έστω αρχίζουμε τον χρωματισμό μας από την περιοχή NT. Έχουμε 3 επιλογές χρωματισμών. Στο επόμενο επίπεδο , για να διατηρήσουμε τους περιορισμούς έχουμε μόνο 2 επιλογές χρωματισμών. Στο επόμενο επίπεδο έχουμε πάλι 2 επιλογές χρωματισμών, αλλά η 1 απορρίπτεται , γιατί θα είναι λανθασμένη στο ακριβώς επόμενο επίπεδο μετά από αυτό. Άρα τελικά μένει μια επιλογή χρωματισμού η οποία είναι και μια από τις τελικές λύσεις σε κάθε ένα από τα φύλα του δέντρου. Αντίστοιχη διαδικασία θα γίνει για όλα τα κλαδιά του δέντρου μέχρι να φτάσουμε στις τελικές καταστάσεις φύλα.



Παρατηρούμε πως για κάθε μια από τις 3 περιπτώσεις θα έχω 2 διαφορετικές λύσεις. Άρα θα έχω σύνολο 2*3=6 λύσεις που δεν παραβιάζουν τους περιορισμούς. Τώρα όμως θα πρέπει να προσθέσουμε στην λύση και την περιοχή Τ που είχαμε αφήσει απ' έξω. Συνεπώς θα έχουμε για κάθε μια λύση 3 διαφορετικούς συνδυασμούς χρωμάτων που μπορεί να πάρει η Τ. Επομένως το τελικό πλήθος διαφορετικών χρωματισμών της Αυστραλίας θα είναι 6*3=18 συνδυασμοί.

Άσκηση (7):

Για να γίνει επεξήγηση του τρόπου που θα επηρεάσει η σταθερή και απειροστά μικρή θερμοκρασία , στον αλγόριθμο προσομοιωμένης ανόπτησης (Simulated Annealing), πρέπει πρώτα να γίνει κατανοητός ο τρόπος λειτουργίας του.

Ο αλγόριθμος, αρχικά ξεκινάει από μια κατάσταση , στην οποία επιλέγει μια τυχαία κίνηση για να πάει στην επόμενη κατάσταση. Αν αυτή η κίνηση μας μεταφέρει σε μια καλύτερη κατάσταση, δηλαδή η αντικειμενική συνάρτηση να έχει καλύτερη τιμή , τότε κρατάω πάντα αυτή την μετάβαση. Αντίθετα, αν δεν δοθεί κάποια καλύτερη κατάσταση και μεταφερθούμε σε μια χειρότερη, τότε δεν την κρατάμε και δοκιμάζουμε νέα τυχαία κίνηση από την αρχή. Αυτή η διαδικασία ορισμένες φορές μπορεί να κολλήσει τον αλγόριθμο γύρω από ένα τοπικό μέγιστο. Για να αποφευχθεί αυτό επιλέγουμε να μεταφερθούμε σε μια χειρότερη κατάσταση (χειρότερη τιμή) με πιθανότητα $P = E^{(\frac{\Delta E}{T})}, με\ \Delta E < 0\ \ και\ \ όπου\ T = θερμοκρασία.$

Επομένως, γίνεται αντιληπτό πως έτσι ο δεν θα υπάρχουν προβλήματα τοπικών μεγίστων και κορυφογραμμών που προκαλούνται από τον αλγόριθμο Hill climbing.Για να επιτευχθεί σωστά αυτή η διαδικασία , επιλέγουμε στην αρχή που τρέχει ο αλγόριθμος μια μεγαλύτερη πιθανότητα επιλογής χειρότερης λύσης, δηλαδή χρησιμοποιούμε υψηλές θερμοκρασίες. Συνεχίζοντας , όσο ο αλγόριθμος τρέχει για

περισσότερη ώρα , θέλουμε να περιορίσουμε τις επιλογές χειρότερης κατάστασης ,επομένως βάζουμε μια μικρότερη πιθανότητα, δηλαδή χρησιμοποιούμε πλέον μικρότερες θερμοκρασίες. Αποτέλεσμα αυτού , είναι να οδηγηθούμε βήμα προς βήμα σε ένα ολικό μέγιστο.

Στην περίπτωση που βάλουμε μια σταθερή και απειροστά μικρή θερμοκρασία , η πιθανότητα επιλογής χειρότερης κατάστασης θα είναι τείνει στο 0. Αυτό θα έχει σαν αποτέλεσμα να μην δεχόμαστε καμία χειρότερη λύση και προφανώς θα αντιμετωπίσουμε το πρόβλημα του να κολλήσουμε σε τοπικά ακρότατα και ποτέ να μην καταφέρουμε να φτάσουμε σε ένα ολικό ακρότατο. Άρα ο Simulated Annealing θα ταυτιστεί με τον απλό Hill Climbing αλγόριθμο.

Άσκηση (8):

Για να δοθεί απάντηση στο ερώτημα αυτό, πρέπει πρώτα να παρατηρήσουμε τι μπορεί να αλλάξει στον κάθε αλγόριθμο και τι όχι.

Αρχικά, για τον A*, έχουμε την συνάρτηση με την οποία επιλέγεται η σειρά προτεραιότητας επέκτασης κόμβων :$\boldsymbol{f(n) = g(n) + h(n)}$,

ό$\boldsymbol{που\ g(n) = πραγματικό\ κόστος\ από\ αρχή\ έως\ κόμβο\ n\ \ και}$

$\boldsymbol{h(n) = εκτιμώμενο\ κόστος\ φθηνότερης\ διαδρομής\ απο\ κόμβο\ n\ σε\ στόχο}$.

Είναι φανερό πως το κόστος g(n), δεν εξαρτάται από κάποια δική μας επιλογή , επομένως δεν μπορούμε να το πειράξουμε. Άρα, θα πρέπει να κάνουμε αλλαγές στο κόστος h(n).

Όσον αφορά τον DFS, θα προσπαθήσουμε να εκμεταλλευτούμε το γεγονός ότι δεν είναι ένας πλήρης αλγόριθμος .Δηλαδή μπορούμε να χρησιμοποιήσουμε μια μη παραδεκτή ευρετική συνάρτηση η οποία θα μας βολέψει στην ταυτοποίηση των δύο συναρτήσεων αξιολόγησης.

Για να πετύχουμε την DFS μέσω του A*, θα πρέπει η κατά βάθος αναζήτηση να μην σταματήσει ποτέ, δηλαδή να μην εξερευνηθεί κόμβος σε μεγαλύτερο βάθος από αυτό που βρίσκεται σε κάθε επίπεδο (μέχρι να ολοκληρώσει την εξερεύνηση όλου του επιπέδου).Για να επιτευχθεί το παραπάνω, θα πρέπει η ευρετική του A* να μην επιλέξει ποτέ διακλάδωση από μεταγενέστερα στάδια, το οποίο είναι εφικτό, αν η απόφαση παίρνεται μονάχα από το h(n). Δηλαδή θέλω να ισχύει πως το κόστος h(n) είναι τόσο μεγαλύτερο από το κόστος g(n) που πλέον το g(n) μπορεί να θεωρηθεί αμελητέο.

Καταλήγουμε λοιπόν στον αλγόριθμο Greedy-Best FS, αφού η νέα ευρετική θα είναι:

$\boldsymbol{f(n) = g(n) + h(n), με\ h(n) \gg g(n) \Rightarrow f(n) = h(n)}$.

Γνωρίζουμε ακόμα πως η χειρότερη περίπτωση του GBFS, οδηγεί σε DFS. Επομένως, διαλέγοντας μια ευρετική που δίνει την χειρότερη περίπτωση καταφέραμε να μετατρέψουμε τον A* σε DFS.

Μια λεπτομέρεια στην διαδικασία αυτή είναι πως σε περίπτωση ισοπαλίας κάποιων κόμβων, θα υπάρχει πρόβλημα, καθώς και θα σπάει την ισοπαλία η τιμή του κόστους g(n), γεγονός που μπορεί να μας χαλάσει τον αλγόριθμο. Άρα θα πρέπει να φροντίσουμε οι τιμές των κόμβων να είναι όλες διαφορετικές αναμεταξύ τους, για να είμαστε σίγουροι πως η διαδικασία αυτή θα δουλεύει πάντα.

Άσκηση (9):

Α)

Στο ερώτημα αυτό , μας ζητήθηκε η αντιμετώπιση του προβλήματος κατασκευής σταυρολέξων σε ένα ορθογώνιο πλέγμα , σαν πρόβλημα ικανοποίησης περιορισμών (Constraint Satisfaction Problems).

Αρχικά, ορίστηκαν οι μεταβλητές οι οποίες στην περίπτωσή μας θα είναι οι λέξεις, δηλαδή σειριακά γράμματα του αλφάβητου.

Περιορισμοί είναι η λέξη να χωράει στα κελιά που είναι επιτρεπτά για το σταυρόλεξο, δηλαδή αυτά που δεν είναι γραμμοσκιασμένα και επίσης αν υπάρχει ένωση (διασταύρωση) 2 λέξεων θα πρέπει το κοινό τους κελί να περιέχει το ίδιο γράμμα.

Ο αλγόριθμος που θα χρησιμοποιηθεί στο ερώτημα αυτό θα είναι ένας backtracking search.Συγκεκριμένα, θα δοκιμάζει αναθέσεις λέξεων που υπάρχουν στο σετ και που δεν έχουν συμπληρωθεί ακόμα ,σε κελία που είναι διαθέσιμα κάθε φορά. Θα συνεχίζει τις αναθέσεις ώσπου να μην είναι δυνατό να ικανοποιηθεί κάποιος από τους περιορισμούς. Στην περίπτωση που συμβεί αυτό , γίνεται απόρριψη της κατάστασης, (δηλαδή του κλαδιού στο δέντρο με τις περιπτώσεις) και δεν συνεχίζονται οι επεκτάσεις του συγκεκριμένου κλαδιού. Ο αλγόριθμος θα δοκιμάσει προφανώς όλες τις δυνατές διακλαδώσεις μέχρι να βρει κάποια λύση , αν αυτή υπάρχει.

Επομένως, είναι φανερό πως είναι πλήρης, εφόσον άμα υπάρχει κάποια λύση θα βρεθεί σε κάποιο από τα φύλλα του δέντρου με τις περιπτώσεις.

Β)

Στο ερώτημα αυτό , μας ζητήθηκε η αντιμετώπιση του προβλήματος κατασκευής σταυρολέξων σε ένα ορθογώνιο πλέγμα , σαν πρόβλημα αναζήτησης.

Αρχικά, θα επιλέξουμε για την λύση τον αλγόριθμο τοπικής αναζήτησης (Local search). Ορίζουμε σαν μεταβλητές ολόκληρες τις λέξεις και όχι τα γράμματα. Η ευρετική μας συνάρτηση για το κόστος , θα γίνει ο αριθμός των λέξεων που δίνονται στο σταυρόλεξο για να συμπληρωθούν.

Η λειτουργία του αλγόριθμου έχει ως εξής, αρχικά τοποθετούνται λέξεις από το σετ των λέξεων προς συμπλήρωση , στο σταυρόλεξο με τυχαίο τρόπο (στα επιτρεπτά κελία φυσικά). Ύστερα, σε περίπτωση που σε μια διασταύρωση έχω λέξεις με γράμματα που δεν είναι ίδια, τότε μια από τις δύο αυτές λέξεις θα πρέπει να αντικατασταθεί με μια από το σετ λέξεων που δεν έχει συμπληρωθεί ακόμα. Αυτό διότι είναι φανερό πως αν το κρατάγαμε όπως έχει τότε θα αλλοιωνόταν μια από τις δύο λέξεις. Η αναζήτηση θα συνεχίζει έτσι μέχρι να έχουν συμπληρωθεί όλες οι λέξεις από το σετ στο σταυρόλεξο, δηλαδή θα έχουμε μια ολοκληρωμένη λύση, ή μέχρι να μην μπορεί να συμπληρωθεί καμία άλλη λέξη στα κελιά, δηλαδή να μην βρεθεί η καλύτερη κατάσταση, απλά να βρεθεί μια μερική λύση του προβλήματος.

Είναι γεγονός πως ο αλγόριθμος αυτός δεν είναι πλήρης καθώς και δεν βρίσκει πάντα την ολοκληρωμένη λύση και προφανώς εξαρτάται σε μεγάλο βαθμό πως θα γίνει η τοποθέτηση των αρχικών λέξεων του σετ η οποία γίνεται με τυχαίο τρόπο.

C)]

Για το συγκεκριμένο πρόβλημα, πιστεύω πως η λύση του ερωτήματος α είναι καλύτερη, καθώς και βρίσκει πάντα την λύση (είναι πλήρης) , ενώ η λύση του ερωτήματος β έχει πιθανότητα να βρεθεί σε μια μερική λύση η οποία λόγο κάποιας λανθασμένης αρχικής τοποθέτησης, να μην γίνεται να φτάσει ποτέ στην βέλτιστή λύση.

Βιβλιογραφία/Πηγές:

Για την άσκηση 2:

http://www.worldsbestchatbot.com/The_Loebner_Prize

https://en.wikipedia.org/wiki/Loebner_Prize

http://www.worldsbestchatbot.com/Do_Much_More

https://en.wikipedia.org/wiki/Kuki_AI

https://www.aimltechbrief.com/index.php/ai/item/6739-the-winner-of-the-2019-loebner-prize

Για την άσκηση 5:

https://ai.stackexchange.com/questions/16845/how-can-i-formulate-the-k-knights-problem-as-a-constraint-satisfaction-problem?fbclid=IwAR3_aLTE_epfqiZuzK7xzAPqSq-M92bTTz_YBU8yQJWHPV59677jcVld4wI