Πολυτεχνειο Κρητης
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

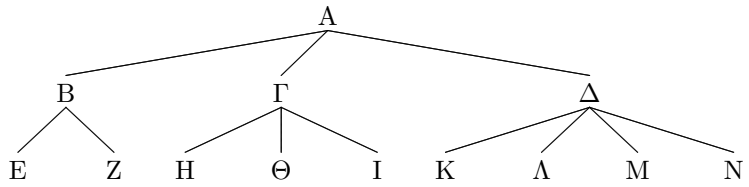
ΠΛΗ 311 – Τεχνητή Νοημοσύνη – 2024

Διδάσκων: Μ. Γ. Λαγουδάκης **1η Σειρά Ασκήσεων** Παράδοση: **20.12.2024, 23:59**

Οδηγίες

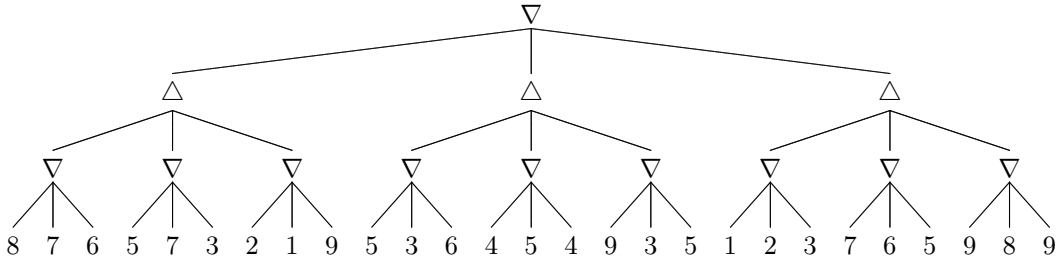
⋆ Οι ασκήσεις είναι ατομικές και η συνεργασία δεν επιτρέπεται (η αντιγραφή συνεπάγεται μηδενισμό).
⋆ Η παράδοση γίνεται **μόνο** ηλεκτρονικά σε μορφή **pdf** μέσω της ιστοσελίδας www.eclass.tuc.gr.

1 [15%] Θεωρήστε το παρακάτω δένδρο αναζήτησης, όπου η αρχική κατάσταση αντιστοιχεί στον κόμβο Α και οι μόνες καταστάσεις-στόχοι αντιστοιχούν στους κόμβους Ζ και Λ. Βέλτιστη είναι όποια διαδρομή οδηγεί σε κόμβο-στόχο με το ελάχιστο συνολικό κόστος διαδρομής. Υποθέστε ότι η συνάρτηση κόστους για μετάβαση σε κάποιον κόμβο από τον γονέα του (εκτός της ρίζας, φυσικά) είναι $c = 3$ για τους κόμβους Γ, Κ, Λ, Μ, Ν και $c = 4$ για τους υπόλοιπους κόμβους. Επίσης, θεωρήστε μια ευρετική συνάρτηση h που ορίζεται ως εξής: $h = 2$ για τον Α, $h = 3$ για τον Β, $h = 1$ για τον Γ, $h = 2$ για τον Δ, και $h = 0$ για τους υπόλοιπους κόμβους. Κατά την επέκταση ενός κόμβου οι απόγονοι δημιουργούνται από αριστερά προς τα δεξιά και σε περιπτώσεις ισοτιμίας προηγείται ο αρχαιότερος κόμβος.



(α) Στο παραπάνω δένδρο, δώστε την ακολουθία επέκτασης κόμβων από την αρχή μέχρι το τέλος της αναζήτησης για κάθε μία από τις παρακάτω στρατηγικές αναζήτησης: πρώτα σε πλάτος (breadth-first), πρώτα σε βάθος (depth-first), ομοιόμορφου κόστους (uniform-cost), επαναληπτική εκβάθυνση (iterative deepening), επαναληπτική επιμήκυνση (iterative lengthening), άπληστη πρώτα στο καλύτερο (greedy best-first), A*, A* με επαναληπτική εκβάθυνση (iterative deepening A*), RBFS, SMA* (μνήμη=3).
(β) Ποιες στρατηγικές βρίσκουν τη βέλτιστη λύση; Γιατί οι υπόλοιπες αποτυγχάνουν;

2 [15%] Θεωρήστε το παρακάτω δένδρο ενός παιχνιδιού και έστω ότι ο MIN παίκτης χρησιμοποιεί αναζήτηση minimax με α-β pruning και η σειρά επέκτασης είναι από αριστερά προς τα δεξιά.



(α) Δώστε την τιμή του παιχνιδιού στη ρίζα και σημειώστε την κίνηση που θα επιλέξει ο MIN.
(β) Σημειώστε όλα τα κλαδιά του δένδρου που θα κλαδευτούν κατά την αναζήτηση.
(γ) Υπάρχει καλύτερη σειρά επέκτασης στη ρίζα (MIN), ώστε να κλαδευτούν περισσότεροι κόμβοι;
(δ) Γενικά, υπάρχει περίπτωση ποτέ ο α-β pruning να κλαδέψει παιδιά της ρίζας; Αιτιολογήστε.
(ε) Αν ο MAX αποκαλύψει ότι επιλέγει πάντα την πρώτη κίνηση, ποια είναι η βέλτιστη κίνηση του MIN;
(στ) Στην περίπτωση (ε), ο MIN πρέπει να ψάξει όλο το δένδρο; Μπορεί να χρησιμοποιήσει α-β pruning;

3 [20%] Θεωρήστε το πρόβλημα της τοποθέτησης 6 βασιλισσών σε σκακιέρα 6×6 και έστω ότι προσπαθούμε να το λύσουμε ως πρόβλημα ικανοποίησης περιορισμών (CSP). Θα αξιοποιήσουμε το γεγονός ότι κάθε βασίλισσα πρέπει να τοποθετηθεί σε διαφορετική στήλη και θα αναφερόμαστε στα τετράγωνα της σκακιέρας με τις συντεταγμένες τους ξεκινώντας από το κάτω-αριστερά τετράγωνο ως $(1, 1)$.

(α′) Διατυπώστε πλήρως το πρόβλημα (μεταβλητές, πεδία τιμών, περιορισμοί) ορίζοντας μία μεταβλητή για κάθε στήλη της σκακιέρας, όπου η τιμή της δηλώνει που τοποθετείται η αντίστοιχη βασίλισσα.

(β′) Έστω ότι εφαρμόζουμε αναζήτηση με υπαναχώρηση στον χώρο των μερικών αναθέσεων (αυξητική διατύπωση). Έστω ότι ξεκινάμε από μία κατάσταση όπου μόνο η μεταβλητή για την πρώτη στήλη έχει πάρει την τιμή 3, δηλαδή η πρώτη βασίλισσα έχει τοποθετηθεί στο τετράγωνο $(1, 3)$. Δείξτε πώς θα προχωρήσει η αναζήτηση (από ποιες καταστάσεις θα περάσει), σημειώνοντας πάνω στις αντίστοιχες σκακιέρες τις αναθέσεις τιμών σε μεταβλητές και τις αποκλεισμένες τιμές από τα πεδία τους, έως ότου φτάσει σε λύση με πιθανές υπαναχωρήσεις. Εξετάστε ξεχωριστά τις παρακάτω παραλλαγές του αλγορίθμου αναζήτησης ανάλογα με τη χρήση γενικών ευριστικών για CSPs:

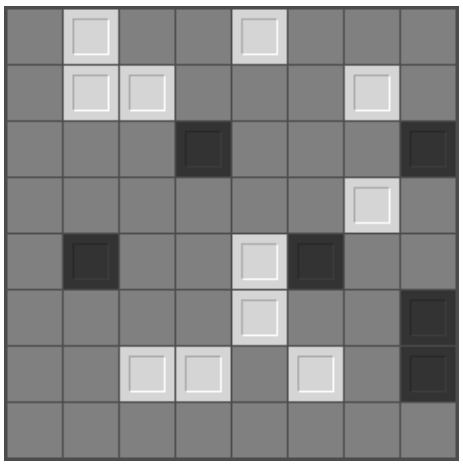
i . πρώιμος έλεγχος, επιλογή μεταβλητής με λιγότερες τιμές, επιλογή τιμής σε αύξουσα σειρά

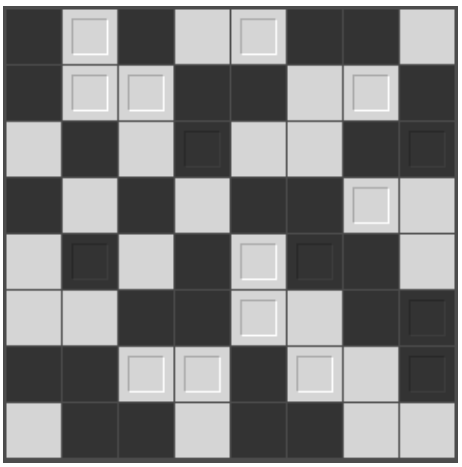
ii . πρώιμος έλεγχος, επιλογή μεταβλητής με λιγότερες τιμές, επιλογή λιγότερο δεσμευτικής τιμής

iii . συνέπεια τόξου, επιλογή μεταβλητής με λιγότερες τιμές, επιλογή λιγότερο δεσμευτικής τιμής

Υπόδειξη: σε πιθανές ισοπαλίες, θεωρήστε ότι προηγείται ο μικρότερος αριθμός (στήλης ή τιμής).

4 [25%] Χρησιμοποιώντας τη γλώσσα προγραμματισμού της αρεσκείας σας, κωδικοποιήστε το παιχνίδι/σπαζοκεφαλιά *Unruly* ως πρόβλημα αναζήτησης και επιλύστε το χρησιμοποιώντας κάποια μέθοδο συστηματικής αναζήτησης. Πρόκειται για ένα παιχνίδι λογικής το οποίο ξεκινά με έναν πίνακα $n \times m$ (τα n, m είναι μεγαλύτερα ή ίσα του 6 και πάντα άρτια) όπου κάποιες θέσεις είναι χρωματισμένες εξ αρχής μαύρες ή άσπρες και οι υπόλοιπες είναι αχρωμάτιστες (στις παρακάτω εικόνες αυτές οι θέσεις φαίνονται γκρίζες). Στόχος μας είναι να χρωματίσουμε τις αχρωμάτιστες θέσεις είτε με μαύρο είτε με άσπρο, χωρίς να χρησιμοποιούμε το ίδιο χρώμα σε τρία διαδοχικά τετράγωνα (οριζόντια ή κάθετα), έτσι ώστε, όταν όλες οι θέσεις έχουν χρωματιστεί, μέσα σε κάθε στήλη και σε κάθε γραμμή του πίνακα να συναντάμε τον ίδιο αριθμό μαύρων και άσπρων τετραγώνων. Παρακάτω, φαίνεται ένα παράδειγμα, με έναν 8×8 πίνακα στην αρχή του παιχνιδιού στα αριστερά και μία πιθανή λύση του στα δεξιά. Μπορείτε να εξασκηθείτε στο παιχνίδι μέσα από την έκδοση που περιλαμβάνεται στο πακέτο `sgt-puzzles` των Debian-based διανομών του Linux ή στην δικτυακή του έκδοση στην ιστοσελίδα http://www.chiark.greenend.org.uk/~sgtatham/puzzles/js/unruly.html.





Διατυπώστε το παιχνίδι ως πρόβλημα αναζήτησης για να μπορέσετε να βρείτε κάποια λύση για ένα δεδομένο στιγμιότυπό του, εφόσον υπάρχει λύση, ή να καταλήξετε στο συμπέρασμα ότι το συγκεκριμένο στιγμιότυπο δεν λύνεται. Σκεφτείτε τι είδους πρόβλημα είναι και ποια μέθοδος αναζήτησης είναι η καταλληλότερη ώστε να εξασφαλίζεται η πληρότητα και η βελτιστότητα.

Ο κώδικάς σας θα πρέπει να λειτουργεί για οποιοδήποτε μέγεθος πίνακα $n \times m$, ($n \geq 6$, $m \geq 6$, n, m άρτια) και για οποιοδήποτε στιγμιότυπο για δεδομένα n, m (πώς θα μπορούσατε να δημιουργήσετε στιγμιότυπα για τα οποία εγγυημένα υπάρχει λύση;). Κατά την εκτέλεση, το πρόγραμμά σας θα πρέπει

να ζητάει από τον χρήστη να επιλέξει το όνομα του αρχείου εισόδου που περιέχει το στιγμιότυπο που θα επιλυθεί, καθώς και το μέγιστο αριθμό κόμβων που θέλει να επεκταθούν κατά την αναζήτηση. Το αρχείο εισόδου θα είναι απλό αρχείο κειμένου ASCII με μία γραμμή και θα πρέπει να ακολουθεί τη δομή που παρουσιάζεται παρακάτω και περιγράφει την αρχική κατάσταση του παραπάνω παραδείγματος:

```
8x8:bceadEDgCcAgCcabBi
```

Αρχικά δίνεται το μέγεθος του πίνακα και κατόπιν μια συμβολοσειρά με γράμματα του λατινικού αλφαβήτου που δηλώνουν τις θέσεις των χρωματισμένων τετραγώνων (κεφαλαία: μαύρα, πεζά: άσπρα). Οι θέσεις μετρώνται ξεκινώντας από πάνω-αριστερά και προχωρώντας οριζόντια μέχρι το τέλος της πρώτης γραμμής, μετά στην αρχή της δεύτερης γραμμής μέχρι το τέλος της, κ.ο.κ. Κάθε γράμμα δηλώνει πόσα τετράγωνα απέχει το επόμενο χρωματισμένο τετράγωνο από το προηγούμενο χρωματισμένο, π.χ. το `g` (ως έβδομο γράμμα του αλφαβήτου) δηλώνει 7 θέσεις απόσταση. Στο παράδειγμα που δόθηκε παραπάνω, το `b` σημαίνει ότι υπάρχει άσπρο τετράγωνο σε απόσταση 2 από την αρχή $(1, 0)$, δηλαδή στην θέση $(1, 2)$, το `c` ότι υπάρχει άσπρο τετράγωνο σε απόσταση 3 από το προηγούμενο, δηλαδή στην θέση $(1, 5)$, το `e` ότι υπάρχει άσπρο τετράγωνο σε απόσταση 5, δηλαδή στην θέση $(2, 2)$, κ.ο.κ. Η κωδικοποίηση ολοκληρώνεται με ένα τελευταίο πεζό γράμμα (`i` στο παράδειγμα) που μας «οδηγεί» ένα τετράγωνο μετά το τέλος της σκακιέρας, το οποίο μπορείτε να το δείτε σαν να υπάρχει ένα τελευταίο άσπρο τετράγωνο στην θέση $(n, m + 1)$. Η ίδια κωδικοποίηση μπορεί να χρησιμοποιηθεί για την αναπαράσταση λύσεων, όπου θα υπάρχουν ακριβώς $nm + 1$ γράμματα, `a` ή `A`. Η λύση του παραπάνω παραδείγματος κωδικοποιείται ως εξής:

```
8x8:AaAaaAAaAaaAAaaAaAaAaaAAAaAaAAaaaAaAaAAaaaAAaaAAAAaaAaaAaAAaAAaaa
```

Η επιλογή της συγκεκριμένης κωδικοποίησης έγινε διότι είναι αυτή που χρησιμοποιείται στην υλοποίηση του παιχνιδιού και έτσι μπορείτε να λαμβάνετε ή να εισάγετε συγκεκριμένα στιγμιότυπα.

Αφού αναγνωσθεί το στιγμιότυπο, θα προβάλλεται στην οθόνη για επιβεβαίωση. Κατόπιν, το πρόγραμμά σας θα προχωράει στην αναζήτηση λύσης, έως ότου βρεθεί λύση ή εξαντληθεί ο δοσμένος αριθμός κόμβων ή τελειώσει η αναζήτηση χωρίς την εύρεση λύσης. Τα αποτελέσματα θα προβάλλονται στην οθόνη και η λύση (εφόσον βρέθηκε) θα αποθηκεύεται σε κάποιο αρχείο εξόδου που θα ακολουθεί την ίδια δομή με το αρχείο εισόδου. Η εκτέλεση θα τελειώνει εμφανίζοντας τον πραγματικό χρόνο εκτέλεσης σε δευτερόλεπτα και τον ακριβή αριθμό των κόμβων που επεκτάθηκαν.

Για ευκολία στην υλοποίησή σας μπορείτε να χρησιμοποιήσετε ελεύθερα τον κώδικα που δίνεται στις ιστοσελίδες των δύο συγγραμμάτων για το πρόβλημα της αναζήτησης. Ουσιαστικά, θα χρειαστεί να διατυπώσετε σωστά το πρόβλημά σας στην αναπαράσταση που απαιτεί ο κώδικας και τις λειτουργίες εισόδου/εξόδου. Λογικά, δεν θα χρειαστεί να υλοποιήσετε την αναζήτηση καθ΄ εαυτή. Καταγράψτε στο γραπτό σας τα βασικά στοιχεία και τις ιδιαιτερότητες του τελικού σας κώδικα.

[10% Bonus] Σκεφτείτε εάν μπορείτε να αξιοποιήσετε γνώση του προβλήματος για να καθοδηγήσετε την αναζήτηση με κάποιο ευριστικό τρόπο, ώστε ει δυνατόν να καταλήγει πιο γρήγορα στην εύρεση λύσης.

5 [25%] Χρησιμοποιώντας τη γλώσσα προγραμματισμού της αρεσκείας σας, κωδικοποιήστε το πρόβλημα του *Unruly* για επίλυση με προσομοιωμένη ανόπτηση (simulated annealing). Στόχος μας είναι η εύρεση ενός χρωματισμού που παραβιάζει όσο γίνεται λιγότερο τους κανόνες του παιχνιδιού (ει δυνατόν, φυσικά, να μην παραβιάζει κανέναν!). Διατυπώστε το πρόβλημα με πλήρεις καταστάσεις (όλα τα τετράγωνα χρωματισμένα). Για την αξιολόγηση καταστάσεων s, χρησιμοποιήστε την αντικειμενική συνάρτηση f

$$f(s) = \sum_{i=1}^{n} \left|rb_i(s) - rw_i(s)\right| + \sum_{j=1}^{m} \left|cb_j(s) - cw_j(s)\right| + \sum_{i=1}^{n} tr_i(s) + \sum_{j=1}^{m} tr_j(s)$$

όπου τα rb_i, rw_i, cb_j, cw_j μετράνε το πλήθος των μαύρων/άσπρων τετραγώνων στην γραμμή i και στη στήλη j αντίστοιχα, ενώ τα tr_i, tr_j μετράνε το πλήθος των τριάδων συνεχόμενων τετραγώνων με το ίδιο χρώμα στην γραμμή i και στη στήλη j αντίστοιχα ($k > 3$ συνεχόμενα τετράγωνα ίδιου χρώματος δίνουν $k - 2$ τριάδες). Προσέξτε ότι η f μηδενίζεται μόνο σε καταστάσεις που δεν παραβιάζουν κανένα

κανόνα του παιχνιδιού, διαφορετικά δίνει κάποια θετική τιμή. Θα πρέπει επίσης να ορίσετε τις «τοπικές κινήσεις» οι οποίες επιτρέπουν μετάβαση από την τρέχουσα κατάσταση σε κάποια διάδοχη γειτονική της κατάσταση (προσέξτε ώστε οι τοπικές κινήσεις να μην αλλάζουν σε καμία περίπτωση το χρώμα των αρχικά χρωματισμένων τετραγώνων), καθώς και ένα κατάλληλο χρονοδιάγραμμα για τη «θερμοκρασία».

Κατά την εκτέλεση το πρόγραμμά σας θα πρέπει να ζητάει από το χρήστη το όνομα του αρχείου εισόδου και τον μέγιστο αριθμό τοπικών βημάτων. Κατόπιν, θα προχωράει στην αναζήτηση λύσης από την αρχική κατάσταση (αφού το στιγμιότυπο εισόδου επεκταθεί σε πλήρη κατάσταση) έως ότου βρεθεί λύση ή εξαντληθεί ο δοσμένος αριθμός βημάτων. Σε κάθε βήμα θα πρέπει να επιλέγετε την τοπική κίνησή σας σύμφωνα με τους κανόνες της προσομοιωμένης ανόπτησης. Το χρονοδιάγραμμα της θερμοκρασίας θα πρέπει να προσαρμόζεται στον μέγιστο αριθμό βημάτων που έχει δοθεί, δηλαδή να ξεκινάει με υψηλή θερμοκρασία στην αρχή, να μειώνεται με κάποιο γραμμικό, εκθετικό, λογαριθμικό, ή σιγμοειδή τρόπο, και να «μηδενίζεται» τελικά κοντά στο τέλος. Η εκτέλεση θα τελειώνει εμφανίζοντας την καλύτερη «λύση» που βρέθηκε (αυτή με τον μικρότερο αριθμό παραβιάσεων ή αλλιώς αυτή που παίρνει την μικρότερη τιμή από την f), τον αριθμό των παραβιάσεων στην «λύση», τον πραγματικό χρόνο εκτέλεσης σε δευτερόλεπτα, και τον ακριβή αριθμό των βημάτων που χρειάστηκαν. Για ευκολία στην υλοποίησή σας μπορείτε να χρησιμοποιήσετε ελεύθερα τον κώδικα που δίνεται στις ιστοσελίδες των δύο συγγραμμάτων.

Καταγράψτε στο γραπτό σας τα βασικά στοιχεία του κώδικά σας και δώστε έναν πίνακα που παρουσιάζει το ποσοστό επιτυχίας (εύρεση λύσης χωρίς παραβιάσεις) για κάποιο δεδομένο στιγμιότυπο, τον μέσο αριθμό παραβιάσεων στη «λύση», τον μέσο αριθμό βημάτων, και τον μέσο χρόνο εκτέλεσης για αυξανόμενες τιμές του μέγιστου αριθμού βημάτων που περιλαμβάνεται στην είσοδο, έπειτα από πολλές εκτελέσεις του κώδικά σας (για να πάρετε μέσες τιμές). Βεβαιωθείτε ότι η γεννήτρια τυχαίων αριθμών αρχικοποιείται σε διαφορετική κατάσταση σε κάθε εκτέλεση, π.χ. περάστε ως φύτρο την τρέχουσα ημερομηνία και ώρα του υπολογιστή σας. Επίσης, συμπεριλάβετε στο γραπτό σας και την καλύτερη λύση που μπορέσατε να βρείτε με την αναζήτησή σας για το δεδομένο στιγμιότυπο.

[10% Bonus] Για κάποια αντιπροσωπευτικά στιγμιότυπα ίδιου μεγέθους καταγράψτε την ποιότητα της τρέχουσας καλύτερης «λύσης» (αριθμός παραβιάσεων) κατά τη διάρκεια της αναζήτησης. Δημιουργήστε γραφικές παραστάσεις της μέσης ποιότητας ως προς τον αριθμό των τοπικών βημάτων για ένα δεδομένο (μεγάλο) μέγιστο αριθμό βημάτων αναζήτησης. Χρησιμοποιήστε δεδομένα από πολλές εκτελέσεις με όλα τα στιγμιότυπα για να έχετε πιο αξιόπιστη εικόνα. Πώς βελτιώνεται η ποιότητα ως προς τον αριθμό των τοπικών βημάτων; Πού παρατηρείται η μεγαλύτερη βελτίωση ή αλλιώς η μεγαλύτερη πρόοδος προς κάποια λύση; Υπάρχει ποιοτική διαφορά για διαφορετικά μεγέθη;

Καλή επιτυχία!