Πανεπιστήμιο Ιωαννίνων

Πολυτεχνική Σχολή

Τμήμα Μηχανικών Η/Υ και Πληροφορικής

Προπτυχιακό Μάθημα: «Τεχνητή Νοημοσύνη»

Προγραμματιστική Εργασία Εξαμήνου

Μέλη Ομάδας – Α.Μ.:

Λάμπρος Βλαχόπουλος – 2948

Βασίλης Νίκας – 3143

Άσκηση 1

Για την άσκηση αυτή χρησιμοποιήσαμε μία μέθοδο A_astro_method(currentNodeChilds.get(i)) η οποία ‘υλοποιεί’ μια ευρετική συνάρτηση. Η ευρετική συνάρτηση αυτή εξασφαλίζει ότι στο μέτωπο αναζήτησης θα εισέρχονται στοιχεία που ικανοποιούν μια συνθήκη με σκοπό να επιταχύνει την εύρεση μιας τελικής κατάστασης. Σε αντίθεση με την ucs μέθοδο όπου στο μέτωπο αναζήτησης καταλήγουν όλα τα στοιχεία που προκύπτουν από υπολογισμούς.

Είναι αποδεκτή καθώς αν βρισκόμαστε σε έναν κόμβο h η απόσταση από τη τελική κατάσταση σύμφωνα με την ευρετική συνάρτηση **είναι μικρότερη ή το πολύ ίση με την πραγματική απόσταση.**

Πχ Έστω h=[1,4,2,3,5] μια ενδιάμεση κατάσταση. Επειδή το 5 είναι σε τελική κατάσταση , η ευρετικη συνάρτηση δε θα κάνει περιττούς υπολογισμούς για νέους κόμβους και θα υπολογίζει τη [**1,4,2,3**|5] για νέους κόμβους χωρίς να επηρεάζετε το 5.

Πιο συγκεκριμένα, στην άσκησή μας είχαμε να διαχειριστούμε λίστες με αριθμούς έστω πχ [2,4,1,5,3] και με συνεχόμενες μεταθέσεις των αριθμών (πχ για N=2 ,[4,2,1,5,3] , N=3 , [1,4,2,5,3], κτλ.) να καταλήξουμε στη λίστα [1,2,3,4,5] δλδ σε αύξουσα σειρά. Η ευρετική έρχεται εδώ να μας επιταχύνει τη διαδικασία εύρεσης καθώς αυτό που εξετάζει είναι αν για κάποια παραγόμενη λίστα αριθμών (πχ με αρχική κατάσταση [5,2,4,3,1], μια τυχαία(N=5) παραγόμενη [1,3,4,2,**5**]) <u>το τελευταίο ψηφίο συμπίπτει με αυτό της λίστας στην τελική κατάσταση</u>(εδώ [1,2,3,4,5]). Έτσι λοιπόν όταν ισχύει αυτό δεν χρειάζεται να κάνουμε άλλες μεταθέσεις (εδώ για N=5) καθώς το στοιχείο αυτό είναι στην τελική του κατάσταση και πρέπει να μεταθέτουμε τα στοιχεία της λίστας στις θέσεις από το αρχικό προς το προτελευταίο και να τσεκάρουμε πλέον αν το προτελευταίο στοιχείο της παραγόμενης λίστας ισούται με το προτελευταίο της τελικής κατάστασης των αριθμών και ούτω κάθε εξής.

Η ευρετική λοιπόν συνάρτηση έχει σκοπό πάντα να εξασφαλίζει ότι στο μέτωπο αναζήτησης θα προστεθεί μόνο η λίστα(αν βρεθέι) όπου έχει κάθε φορά **από** το τελευταίο προς το αρχικό στοιχείο ίδιο με αυτό της τελικής κατάστασης.

Στην υλοποίησή μας , αφού δημιουργήσουμε όλες τις παραγόμενες λίστες που προκύπτουν από μία αρχική , τις ελέγχουμε μία προς μία για το αν εξακριβώνετε η συνθήκη της ευρετικής και αν ισχύει χρησιμοποιούμε μια μεταβλητή (flag) για να σιγουρέψουμε ότι οι παραγόμενες λίστες στη συνέχεια δεν θα μεταθέτουν τα στοιχεία τους από το τελευταίο **ή** προς εξέταση στοιχείο που είναι στην τελική του κατάσταση. Στη συνέχεια σβήνουμε όλο το μέτωπο αναζήτησης και βάζουμε σε αυτό τη πρώτη λίστα που θα βρεθεί να ικανοποιεί την ευρετική συνάρτηση από την οποία θα προκύψουν λίστες με μεταθέσεις **εκτός** του τελευταίου ή **προς εξέταση** στοιχείου.

Ένα παράδειγμα λειτουγίας

Εστω αρχική κατάσταση [3,1,4,2]

Για N=2 -> [1,3,4,2]

Για N=3 -> [4,1,3,2]

Για N=4 ->[2,4,1,3]

Οι παραγόμενες λίστες δεν ικανοποιούν την ευρετικη οπότε όλες μπαίνουν μέτωπο αναζήτησης.

Στη συνέχεια από την [1,3,4,2] θα προκύψουν

`[3, 1, 4, 2] , θα σβηστεί καθώς είναι το ίδιο με αρχικό`
`[4, 3, 1, 2]`
`[2, 4, 3, 1]`

Οι παραγόμενες λίστες δεν ικανοποιούν την ευρετικη οπότε όλες μπαίνουν μέτωπο αναζήτησης.

Στη συνέχεια από την [4,1,3,2] θα προκύψουν

[1,4,3,2]

[3,1,4,2], `, θα σβηστεί καθώς είναι το ίδιο με αρχικό`

[2,3,1,**4**] ,finalstate=[1,2,3,**4**]

Εδώ βλέπουμε ότι η ευρετική ικανοποιείτε οπότε στο Μ. Αναζήτησης όπου πρώτα θα σβηστούν τα περιεχόμενα του θα μπει μόνο η [2,3,1,4] από την οποία θα προκύψουν

[3, 2, **1**, 4]
[1, 3, **2**, 4] , finalstate=[1,2,**3**,4]

Οι παραγόμενες λίστες δεν ικανοποιούν πλέον την ευρετικη οπότε όλες μπαίνουν μέτωπο αναζήτησης.

Στη συνέχεια από την [3, 2, 1, 4] θα προκύψουν

[2,3,1,4] **,** θα σβηστεί καθώς είναι το ίδιο με αρχικό

[1,2,3,4] ικανοποιεί την ευρετική καθώς finalstate=[1,2,**3**,4]

Αλλά αποτελεί λύση καθώς όλα τα στοιχεία έφτασαν στην τελική κατάσταση οπότε το πρόγραμμα τερματίζει.

Σημείωση:

Η ευρετική μας συνάρτηση θα γινόταν ακόμα πιο βέλτιστη αν:

Αντί για κάθε φορά που ελέγχουμε μία παραγόμενη λίστα για το αν ικανοποιεί την ευρετική(δλδ. τον έλεγχο για το τελευταίο στοιχείο ή αυτό που εξετάζουμε αν είναι σε τελική κατάσταση) και την πρώτη λίστα που βρούμε τη βάζουμε στο Μ. Αναζήτησης **, να ελέγχουμε** όλες τι παραγόμενες λίστες για περισσότερα ίδια ψηφία με την τελική κατάσταση και να βάζουμε τη λίστα αυτή στο Μ. Αναζήτησης.

Δλδ. αν από μία αρχική κατάσταση προέκυψαν

[2,4,3,1,5]

[4,5,3,1,2]

[2,1,3,4,5]

[3,2,1,4,5]

Να μην μπει η [2,4,3,1,5] που μπαίνει στο πρόγραμμα μας αλλά να μπει η [2,1,3,4,5] , καθώς περισσότερα ψηφία είναι σε τελική κατάσταση.

Παραδείγματα λειτουργίας Προγράμματος

List	4 2 1 3	2 1 4 5 3	5 1 4 3 6 2	2 1 4 5 3 6 7	4 5 1 2 3 7 6 8
Ucs	Cost: 3 Exexpansions:102	Cost: 3 Exexpansions:264	Cost: 5 Exexpansions:12565	Cost: 3 Exexpansions:804	Cost: 5 Exexpansions:31129
A*	Cost: 3 Exexpansions:9	Cost: 5 Exexpansions:28	Cost: 7 Exexpansions:50	Cost: 7 Exexpansions:54	Cost: 7 Exexpansions:77

Παρατηρούμε λοιπόν, πόσο πολύ πιο γρήγορη γίνετε η αναζήτησή μας με τη χρήση της ευρετικης συνάρτησης, καθώς οι επεκτάσεις των 2 αλγορίθμων όσο μεγαλώνει η λίστα εισόδου έχουν τεράστιες διαφορές σε υπολογιστικό κόστος.