Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Προγραμματιστικές Γλώσσες Ι, 2020-2021, – 3η σειρά ασκήσεων

ΑΝΑΛΥΣΗ ΤΟΥ ΤΡΟΠΟΥ ΕΠΙΛΥΣΗΣ ΤΗΣ ROUND

Βασική Ιδέα: Πρέπει να βρούμε την πόλη εκείνη, από την οποία τα αυτοκίνητα θα απέχουν συνολικά (=άθροισμα αποστάσεων) τη μικρότερη απόσταση. Ταυτόχρονα πρέπει να ισχύει ο περιορισμός ότι το αμάξι που θα απέχει τη μεγαλύτερη απόσταση από την πόλη, θα απέχει το πολύ ένα βήμα παραπάνω από τη συνολική απόσταση όλων των άλλων πόλεων. Ο περιορισμός αυτός εξασφαλίζει ότι πάντοτε τα αυτοκίνητα θα κινούνται εναλλάξ, δηλαδή κανένα δε θα κάνει δύο διαδοχικές κινήσεις.

Standard ML

Για να βρούμε τη συγκεκριμένη πόλη εργαστήκαμε ως εξής στην ML:

1. Δημιουργήσαμε μια λίστα από λίστες με όλες τις δυνατές τελικές καταστάσεις των αυτοκινήτων. Για παράδειγμα αν $C = 4$ $(cars)$ και $T = 5$ $(towns)$ τότε οι τελικές δυνατές καταστάσεις είναι οι εξής:
   - $[0, 0, 0, 0]$ $(όλα\ τα\ αυτοκίνητα\ στην\ πόλη\ 0)$
   - $[1, 1, 1, 1]$
   - $[2, 2, 2, 2]$
   - $[3, 3, 3, 3]$
   - $[4, 4, 4, 4]$
2. Στη συνέχεια βρίσκουμε τη διαφορά **κάθε** δυνατής τελικής κατάστασης με την αρχική μας. Εδώ προσέξτε ότι αν η τελική πόλη είναι μεγαλύτερη από την αρχική, πχ. είμαστε στη 2 και θέλουμε να πάμε στην 4, τότε απλά αφαιρούμε $4 - 2 = 2$. Αν όμως είμαστε στην 2 και θέλουμε να πάμε στην 0 τότε πρέπει να κάνουμε την πράξεις $T - 2 + 0 = 5 - 2 + 0 = 3$ (όπου T το σύνολο των πόλεων). Αν η αρχική μας κατάσταση είναι η $[2, 0, 2, 2]$ τότε για την τελική κατάσταση [0, 0, 0, 0] η απόσταση είναι $[3, 0, 3, 3]$, ενώ για την $[4, 4, 4, 4]$ είναι η $[2, 4, 2, 2]$
3. Τώρα βρίσκουμε το άθροισμα και το μέγιστο κάθε διαφοράς που βρήκαμε στο προηγούμενο βήμα. Για παράδειγμα για τη διαφορά $[3, 0, 3, 3]$ το μέγιστο είναι το 3 και το άθροισμα το 9.
4. Ελέγχουμε αν ο συνδυασμός μέγιστου-αθροίσματος ικανοποιούν τον περιορισμό που εξηγήσαμε στην **βασική ιδέα**. Πχ. για max=3, sum=5 ο περιορισμός ικανοποιείται, όχι όμως για max=3, sum=4.
5. Αν λοιπόν ικανοποιείται η παραπάνω συνθήκη τότε το άθροισμα αποστάσεων για την πόλη που έχουμε βρει για τη συγκεκριμένη τελική κατάσταση είναι αποδεκτή λύση. Έτσι συγκρίνουμε το συγκεκριμένο άθροισμα για να διαπιστώσουμε αν είναι το ελάχιστο που έχουμε βρει προς το παρόν και συνεχίζουμε την αναζήτηση για την επόμενη πιθανή τελική κατάσταση. Προφανώς μαζί με την ελάχιστη απόσταση κρατάμε και το αντίστοιχο index.

Τα παραπάνω βήματα μπορούν να υλοποιηθούν σειριακά στην ML, όπως όμως θα διαπιστώσετε, αν προσπαθήσετε να τρέξετε το πρόγραμμα δε θα σας περνάει το τελευταίο testcase, οπότε θα πρέπει να συγχωνεύσετε κάποιες διαδικασίες, να μην παράγετε κάποια ενδιάμεσα «προϊόντα» που αποσκοπούν στην κατανόηση της λύσης. Στη δική μας περίπτωση περνούσε το τελευταίο testcase στα 2.2 sec.

Prolog

Η προσέγγιση που ακολουθήσαμε στην Prolog είναι αρκετά διαφορετική.

Φυσικά στην αρχή προσπαθήσαμε να μεταφράσουμε τη λύση της ML. Ωστόσο, όσο και αν τροποποιήσαμε και βελτιστοποιήσαμε τη λύση, δε μπορούσε με τίποτα να περάσει το τελευταίο testcase.

Έτσι καταλήξαμε να σκεφτούμε να αντιστρέψουμε τον αρχικό πίνακα που μας δίνεται. Ο αρχικός πίνακας δείχνει σε ποια πόλη βρίσκεται το κάθε αυτοκίνητο. Με βάση αυτή εμείς κατασκευάζουμε έναν νέο πίνακα που δείχνει πόσα αυτοκίνητα υπάρχουν σε κάθε πόλη. Πχ. αν υποθέσουμε ότι έχουμε 5 πόλεις, από τον παρακάτω **αρχικό πίνακα** φτιάχνουμε τον αντίστοιχο **πίνακα πόλης**:

$$[2, 0, 2, 2] \rightarrow [1, 0, 3, 0, 0]$$

Για να αντιστρέψουμε τον αρχικό πίνακα μπορούμε να σκεφτούμε διάφορους τρόπους. Ένας αρκετά αποδοτικός τρόπος είναι να ταξινομήσουμε τον αρχικό πίνακα και μετά να αθροίζουμε τις εμφανίσεις αυτοκινήτων για κάθε πόλη.

Βρίσκουμε τώρα την απόσταση, **sum**, όλων των αυτοκινήτων και τη μέγιστη απόσταση από τη μηδενική τελική κατάσταση [0, 0, … , 0]. Για να γίνει αυτό μπορούμε να χρησιμοποιήσουμε για μία φορά τον αρχικό πίνακα.

Γνωρίζοντας λοιπόν το αρχικό μας sum & max δημιουργούμε δύο δείκτες και σαρώνουμε τον πίνακα πόλης. Και οι δύο δείκτες κινούνται αυστηρά από τα αριστερά προς τα δεξιά. Μόλις ο **κύριος δείκτης** φτάσει στο τέλος της λίστας η διαδικασία τελειώνει. Ο **μέγιστος δείκτης** δείχνει κάθε φορά στο στοιχείο του πίνακα που είναι δεξιά του κύριου δείκτη και που δεν είναι 0. Προφανώς αν ο κύριος δείκτης είναι ίσος με τον μέγιστο, ο δεύτερος πρέπει να μετακινηθεί, και αν φτάσει στο τέλος της λίστα, θα πρέπει να κάνει σαν το Pacman και να πάει στην αρχή της λίστας. Με κάθε κίνηση του μέγιστου δείκτη ενημερώνουμε αντίστοιχα και τη μέγιστη απόσταση (δηλαδή τη μεγαλύτερη τρέχουσα απόσταση κάποιου αυτοκινήτου από την πιθανή τελική πόλη).

Τώρα το σημαντικό είναι ότι με κάθε κίνηση του κύριου δείκτη ανανεώνουμε αντίστοιχα το sum ως εξής:

$$NewSum = Sum + C - T \times ΠίνακαςΠόλης[i]$$

όπου C ο αριθμός των αυτοκινήτων, T ο αριθμός των πόλεων και i ο κύριος δείκτης.

Επίσης με κάθε κίνηση του κύριου δείκτη ενημερώνουμε κατάλληλα τη μέγιστη απόσταση.

Ελέγχουμε λοιπόν αν η μέγιστη απόσταση και το νέο sum τηρούν τη βασική προϋπόθεση του προβλήματος (δηλαδή $sum - max + 1 \leq max$). Αν ναι τότε η τιμή του sum είναι έγκυρη και, μπορούμε να τη χρησιμοποιήσουμε για να ανανεώσουμε τη Min τιμή, που κρατάμε για το τελικό αποτέλεσμα (μαζί και το index).

Συνεχίζουμε αναδρομικά μέχρι το τέλος της λίστας.

Τώρα όλα τα παραπάνω θα γίνουν πιο κατανοητά με ένα παράδειγμα. Θα χρησιμοποιήσουμε αυτό της εκφώνησης για το οποίο ήδη έχουμε βρει ότι το ο πίνακας πόλης είναι ο [1, 0, 3, 0, 0], ότι η συνολική

απόσταση από την τελική πιθανή κατάσταση [0, 0, 0, 0] είναι 9 και ότι η αρχική μέγιστη απόσταση είναι 3. Με **πράσινο** χρώμα είναι ο κύριος δείκτης και με **κόκκινο** ο μέγιστος δείκτης.

Πίνακας Πόλης	Sum	Μέγιστη Απόσταση	Sum-Max έγκυρο;	Min	Θέση Min
[**1** , 0 , **3** , 0 , 0]	**9**	3	ΝΑΙ	9	0
[1 , **0** , **3** , 0 , 0]	9+4 = **13**	4	ΝΑΙ	9	0
[**1** , 0 , **3** , 0 , 0]	13 + 4 – 3*4 = **2**	2	ΟΧΙ	9	0
[**1** , 0 , 3 , **0** , 0]	2+4=**6**	3	ΝΑΙ	6	3
[**1** , 0 , 3 , 0 , **0**]	6+4=**10**	4	ΝΑΙ	**6**	**3**

Η τελική απάντηση με το κίτρινο.

Να σημειωθεί ότι η συγκεκριμένη λύση περνάει άνετα και το τελευταίο testcase.

Καλό Καλοκαίρι! *(...αν και δε θυμάμαι τι είναι αυτό)*