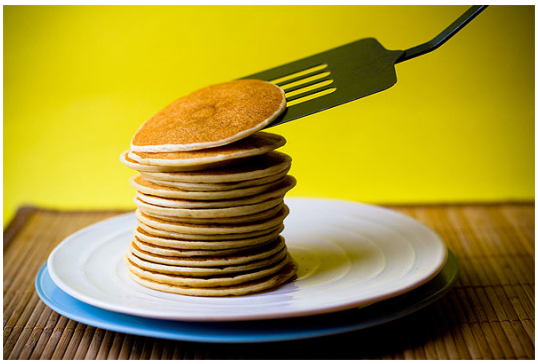
Αρχές Γλωσσών Προγραμματισμού
Άσκηση 3η - Pancake Sorting
Ημερομηνία ανάρτησης: 8/1/2019

Το Πρόβλημα



Σε ένα εστιατόριο, ένας απρόσεκτος σεφ ετοιμάζει μια στοίβα από pancakes που όλα έχουν διαφορετικό μέγεθος. Ο σερβιτόρος, για να ευχαριστήσει τους πελάτες, προσπαθεί να ταξινομήσει τα pancakes έτσι ώστε το μικρότερο να βρίσκεται στην κορυφή ενώ το μεγαλύτερο στην βάση. Το μόνο που μπορεί να κάνει είναι να βάλει μια σπάτουλα κάτω από κάποιο pancake και να αναποδογυρίσει την στοίβα απο εκείνο το σημείο μέχρι την κορυφή. Επειδή ο χρόνος του είναι περιορισμένος θέλει να βρει τον ελάχιστο αριθμό κινήσεων (και ποιές είναι αυτές) για να τα καταφέρει. Βασίζεται πάνω σας να γράψετε ένα πρόγραμμα Haskell που θα τον σώσει!

Απλή ταξινόμηση (15)

Σε πρώτο βήμα αρκεί να υλοποιήσετε μια συνάρτηση που δοσμένης μίας στοίβας από τα μεγέθη των pancakes, θα επιστρέφει μια οποιαδήποτε ακολουθία από flips που οδηγούν σε ταξινομημένη στοίβα.

Πιο συγκεκριμένα, θεωρήστε ότι στη λίστα των ακεραίων που αναπαριστά τη στοίβα, το πρώτο στοιχείο αντιστοιχεί στην κορυφή της στοίβας, καθώς και ότι ένα flip αναπαρίσταται από τη θέση του pancake κάτω από το οποίο μπαίνει η σπάτουλα για το αναποδογύρισμα (ξεκινώντας από το 1).

Η ζητούμενη συνάρτηση είναι: `naive :: [Int] -> [Int]`
Για παράδειγμα: `naive [2,7,3,1] = [2,4,2,3,2]` (5 κινήσεις)

Εξήγηση: `[2,7,3,1] -- 2 --> [7,2,3,1] -- 4 --> [1,3,2,7] -- 2 --> [3,1,2,7] -- 3 --> [2,1,3,7] -- 2 --> [1,2,3,7]`

Επίσης καλείστε να φτιάξετε και μία συνάρτηση που με είσοδο μια αρχική στοίβα και μια ακολουθία κινήσεων θα επιστρέφει την λίστα από στοίβες που οπτικοποιούν τα flips.

Η ζητούμενη συνάρτηση είναι: `visualize :: [Int] -> [Int] -> [[Int]]`
Για παράδειγμα: `visualize [2,7,3,1] [2,4,2,3,2] = [[2,7,3,1],[7,2,3,1],[1,3,2,7],`
`[3,1,2,7],[2,1,3,7],[1,2,3,7]`

Ελάχιστος αριθμός κινήσεων (20)

Στη συνέχεια θα πρέπει να βρίσκετε τις ελάχιστες κινήσεις που απαιτούνται για την ταξινόμηση της στοίβας. Για να το κάνετε αυτό θα πρέπει να μοντελοποιήσετε το Pancake Sorting σαν ένα πρόβλημα Shortest Path και να υλοποιήσετε τον αλγόριθμο BFS (Breadth First Search) για να βρείτε την πιο σύντομη διαδρομή στο γράφημα με τις στοίβες, δηλαδή την λύση με τον ελάχιστο αριθμό κινήσεων.

Υπόδειξη: Ένας τρόπος να υλοποιήσετε τον BFS είναι χρησιμοποιώντας μία FIFO queue ως σύνορο και ορίζοντας κατάλληλη δομή δεδομένων Node για τα στοιχεία της. Σε κάθε επανάληψη του BFS αφαιρείτε ένα Node από την ουρά και το επεκτείνετε (δηλαδή προσθέτετε τα successor Nodes του στην ουρά) μέχρι να βρείτε το Goal Node που περιέχει την ταξινομημένη στοίβα. Τότε θα πρέπει να χτίσετε την ακολουθία κινήσεων που σας οδήγησε εκεί και να την επιστρέψετε. Επίσης, θα πρέπει να ελέγχετε για διπλότυπες καταστάσεις καθώς κάνετε BFS, αλλίως η αναζήτηση θα είναι πολύ αργή.

Η ζητούμενη συνάρτηση είναι: `bfs :: [Int] -> [Int]`
Για παράδειγμα: `bfs [12,23,9,36,20,34,4] = [2,3,6,2,4,3,2,7]`

Οποιαδήποτε σωστή λύση είναι αποδεκτή αρκεί να κάνει τον ελάχιστο αριθμό κινήσεων. Αξίζει να σημειωθεί ότι το συγκεκριμένο πρόβλημα είναι NP-hard, οπότε πιστεύουμε ότι δεν θα καταφέρετε να βρείτε κάποια πολυωνυμική λύση (στο μέγεθος της στοίβας). Η λύση σας θα δοκιμαστεί με μικρές στοίβες, των οποίων η βέλτιστη λύση δεν απαιτεί πολλά flips.

Επιτρέπεται και προτείνεται να χρησιμοποιήσετε βιβλιοθήκες της Haskell (ghc) όπως οι `Data.List, Data.Set, Data.Map` στην υλοποίησή σας.

Bidirectional search (30)

Για να μειώσετε το search space μπορείτε να ξεκινάτε δύο BFS μαζί, έναν από την αρχική στοίβα (είσοδος) και έναν από την ταξινομημένη (στόχος). Εξερευνώντας ταυτόχρονα και απο τις δύο μεριές, μπορείτε να σταματάτε αν δείτε ότι κάποιος κόμβος του ενός BFS έχει βρεθεί ήδη στον άλλον. (σε κάθε βήμα του αλγορίθμου θα πρέπει να προχωράτε την κάθε αναζήτηση ένα επίπεδο πιο βαθιά).

Και εδώ θα σας φανούν χρήσιμες οι βιβλιοθήκες για να ελέγχετε αν υπάρχουν κοινά states στις δύο αναζητήσεις και να σταματάτε.

Η ζητούμενη συνάρτηση είναι: `bidirectional :: [Int] -> [Int]`
Για παράδειγμα: `bidirectional [32,3,19,10,40,15,6,21,1] = [7,5,3,9,3,8,3,2,6]`

Εδώ θα δοκιμάσουμε την λύση σας με δυσκολότερα instances (μεγαλύτερο βάθος της συντομότερης διαδρομής και περισσότερα pancakes).

Πολλές στοίβες μαζί (20)

Εχει πέσει πολλή δουλειά και ο σερβιτόρος έχει να ταξινομήσει πολλές στοίβες μαζί. Σας δίνει λοιπόν μια λίστα με το πολύ 10000 στοίβες (θεωρήστε ότι όλες έχουν το ίδιο μέγεθος) και θέλει να επιστρέψετε μια λίστα με ακολουθίες κινήσεων. Εκμεταλευτείτε αυτήν τη λεπτομέρεια για να λύσετε όλες τις στοίβες χωρίς να καλείτε BFS για κάθε μία ξεχωριστά. Οι στοίβες που θα δίνονται, θα είναι αντίστοιχης δυσκολίας με αυτές του $2^{ου}$ ερωτήματος.

Η ζητούμενη συνάρτηση είναι: `batch :: [[Int]] -> [[Int]]`
Για παράδειγμα:
```
batch [[6,12,1,7,3],[1,4,2,7,8],[10,2,11,8,9]] =
      [[4,3,5,3,4],[2,3,2],[2,3,5,3]]
```

Καμένα pancakes (15)

Τώρα ο σεφ εκτός των άλλων βγάζει pancaces που είναι καμένα απο τη μία τους πλευρά. Σκοπός του σερβιτόρου είναι μετά τα flips η στοίβα να είναι ταξινομημένη όπως πριν, και επιπλέον όλα τα καμένα μέρη να δείχνουν προς τον πάτο του πιάτου (για να μην φαίνονται). Τροποποιήστε τις προηγούμενες υλοποιήσεις σας ώστε να δουλεύουν και για καμένα pancakes. Εδώ η είσοδος θα είναι μια λίστα από ζεύγη που θα περιέχουν το μέγεθος, καθώς και το από ποια μεριά είναι καμένο το κάθε ένα (1=πάνω 0=κάτω).

Για παράδειγμα με είσοδο `[(4,1),(9,0),(1,0),(10,1),(17,0)]`, η ταξινομημένη στοίβα θα είναι `[(1,0),(4,0),(9,0),(10,0),(17,0)]`.

Τα ονόματα των συναρτήσεων θα είναι: `burnt_naive burnt_visualize burnt_bfs burnt_bidirectional burnt_batch` και τα όρια της εισόδου θα είναι μικρότερα λόγω της νέας πολυπλοκότητας.

Bonus (30)

Παρ΄ όλο που δεν έχει βρεθεί αποδοτικός αλγόριθμος για την εύρεση των ελάχιστων κινήσεων για την ταξινόμηση μια στοίβας, υπάρχουν αλγόριθμοι που βρίσκουν αρκετά καλές λύσεις σε πολυωνυμικό χρόνο. Ένας τέτοιος αλγόριθμος παρουσιάζεται στο [1], ως απόδειξη ότι ο ελάχιστος αριθμός κινήσεων για μία στοίβα μεγέθους n είναι $\leq (5n+5)/3$. Αυτό το φράγμα παρέμεινε το ελάχιστο δυνατό για περίπου 30 χρόνια! Μελετήστε προσεκτικά τον αλγόριθμο που παρουσιάζεται στο [1] και υλοποιήστε τον σε Haskell. Η υλοποίησή σας θα δοκιμαστεί με μεγάλες στοίβες όπως αυτές του $1^{ου}$ ερωτήματος (π.χ. $n \leq 1000$) και η λύση σας δεν θα πρέπει να ξεπερνάει τον αριθμό των κινήσεων του φράγματος.

Η ζητούμενη συνάρτηση είναι: `gates :: [Int] -> [Int]`
Για παράδειγμα: `gates [9,27,12,1,4] = [3,2,5,2]`

[1] William H. Gates, Christos H. Papadimitriou: Bounds for sorting by prefix reversal. Discrete Mathematics 27(1): 47-57 (1979)

Παρατηρήσεις

Όλες οι συναρτήσεις που θα υλοποιήσετε θα δοκιμαστούν με διάφορα test cases. Μια εκτέλεση θα θεωρείται σωστή αν βγάζει αποτέλεσμα μέσα σε διάστημα μερικών δευτερολέπτων, και επιπλέον η ακολουθία κινήσεων που επιστρέφει είναι έγκυρη και έχει το ελάχιστο δυνατό μήκος (εκτός από το 1^o και το bonus).

Υποβολή

Η άσκηση μπορεί να γίνει ατομικά ή σε ομάδες δύο ατόμων. Στη δεύτερη περίπτωση θα παραδοθεί μόνο από το ένα μέλος της ομάδας. Θα πρέπει να παραδώσετε ένα αρχείο `zip` με όνομα τους αριθμούς μητρώου σας, το οποίο θα περιέχει το `pancakes.hs` με τις λύσεις σας και ένα `readme` με μια σύντομη περιγραφή των επιλογών σας. Θα πρέπει να στείλετε το `zip` στις διευθύνσεις `sdi1500018@di.uoa.gr`, `sdi1500102@di.uoa.gr` και `prondo@di.uoa.gr`. Ερωτήσεις σχετικά με την άσκηση πρέπει να απευθύνονται στα δύο πρώτα emails (Μάρκος Βολίκας και Μιχάλης Μπιζίμης).