Τεχνητή Νοημοσύνη
Αναφορά Άσκησης 1:

Υλοποίηση αποδεκτής ευρετικής συνάρτησης για αναζήτηση Α*

Μέλη ομάδας:

Αντωνίου Χριστόδουλος	ΑΜ: 2641
Έφη Καρανίκα	ΑΜ: 2453
Γεράσιμος Πιτούλης	ΑΜ: 2803

Για να βρούμε **ευρετική συνάρτηση h(n)** για την αναζήτηση A* της άσκησης που μας δόθηκε, ουσιαστικά κληθήκαμε να εκτιμήσουμε την απόσταση που έχει μία δοθείσα κατάσταση από την πλησιέστερη τελική κατάσταση. Επιπλέον, για να είναι **αποδεκτή** η ευρετική μας συνάρτηση, έπρεπε να επιβεβαιώσουμε πως δεν επιστρέφει ποτέ τιμή μεγαλύτερη από την πραγματική απόσταση της δοθείσας κατάστασης από την πλησιέστερη τελική. Και φυσικά, να επιστρέφει μηδέν εάν πρόκειται για τελική κατάσταση.

Η προσέγγιση που ακολουθήσαμε ήταν πειραματική. Ορίσαμε μία μέθοδο υπεύθυνη για τον υπολογισμό της τιμής της ευρετικής συνάρτησης. Η μέθοδος δέχεται ως παράμετρο μία λίστα ακεραιών, η οποία περιγράφει την δοθείσα κατάσταση και επιστρέφει την τιμή της ευρετικής συνάρτησης για την κατάσταση αυτή. Αρχικά, δηλώσαμε την μεταβλητή **heuristic**, την οποία και επιστρέφει η συνάρτηση, και την αρχικοποιήσαμε στο μηδέν. Στη συνέχεια προσπελάζουμε την λίστα ξεκινώντας από το δεύτερο στοιχείο της, και σε κάθε μία από τις θέσεις αυτές ελέγχουμε αν υπάρχει ο αριθμός που θα αναμέναμε σε μία τελική κατάσταση. Η λογική για την εξαίρεση του πρώτου στοιχείου είναι πως, εάν το πρώτο στοιχείο δεν είναι στην αναμενόμενη θέση που θα ήταν σε τελική κατάσταση, τότε σίγουρα το ίδιο θα ισχύει και για κάποιο άλλο στοιχείο της λίστας. Ένας επιπλέον λόγος που ξεκινάμε την προσπέλαση από το δεύτερο στοιχείο είναι πως καταλήξαμε να κάνουμε περαιτέρω ελέγχους του κάθε στοιχείου της λίστας με το στοιχείο που προηγείται.

Για τον πρώτο αριθμό που θα βρεθεί (εάν βρεθεί) σε διαφορετική θέση από την αναμενόμενη, αυξάνουμε την μεταβλητή στο ένα. Με τον τρόπο αυτό εξασφαλίσαμε πως η ευρετική συνάρτηση επιστρέφει πάντα μηδέν για κάποια τελική κατάσταση, ενώ για μη τελική κατάσταση επιστρέφει ένα. Στο σημείο αυτό επιχειρήσαμε να βελτιώσουμε την ευρετική συνάρτηση πειραματικά, δοκιμάζοντας διάφορα **if** με διαφορετικές συνθήκες για να αυξήσουμε περαιτέρω την μεταβλητή heuristic. Εν τέλει, καταλήξαμε να προσθέσουμε μία ακόμη συνθήκη εμφολευμένη στην ήδη υπάρχουσα if που ελέγχει αν πρόκειται για τελική κατάσταση. Συγκεκριμένα, για κάθε στοιχείο στην επανάληψη, ελέγχουμε το προηγούμενο από αυτό. Ελέγχουμε εάν είναι μεγαλύτερο και εάν είναι διαφορετικό από τον αμέσως μεγαλύτερο ακέραιο. Δηλαδή, ουσιαστικά ελέγχουμε εάν τα δύο στοιχεία της λίστας βρίσκονται σε φθίνουσα κατάταξη. Θεωρούμε δεδομένο πως το πρόγραμμα δεν δέχεται μη έγκυρες λίστες, σύμφωνα με τους ελέγχους που έχουμε υλοποιήσει στην αρχή της εκτέλεσης. Έτσι, εάν ισχύουν και οι δύο συνθήκες, μπορούμε να συμπαιράνουμε πως θα χρειαστεί τουλάχιστον μία ακόμα μετάβαση και άρα αυξάνουμε την μεταβλητή κατά ένα. Αυτό επαναλαμβάνεται και για τα υπόλοιπα στοιχεία της προσπέλασης.

Με βάση την παραπάνω επεξήγηση, συμπεράναμε πως η ευρετική συνάρτηση που υλοποιήσαμε είναι και **αποδεκτή**. Ωστόσο, έχοντας κάποιες αμφιβολίες, προχωρήσαμε και σε πειραματική επαλήθευση. Συγκεκριμένα προσθέσαμε κάποια **print** στην συνάρτηση alphastar τα οποία μας παρέχουν πληροφορία για το κόστος g(n), την ευρετική h(n), το άθροισμα αυτών καθώς και την κατάσταση που τελικά επιλέχθηκε από το μέτωπο αναζήτησης. Με τον τρόπο

αυτό μπορέσαμε να επαληθέυσουμε και πειραματικά πως η ευρετική είναι αποδεκτή. Ακολουθεί στιγμιότυπο μίας ενδεικτικής εκτέλεσης, **java ask1 alphastar 3,4,1,2 > output**. Για ευκολία στην ανάγνωση έχουμε κάνει redirect το output σε αντίστοιχο αρχείο, προσθέτοντας το **> output** στο τέλος της εντολής. Σημειώνεται πως για μεγαλύτερο αριθμό θέσεων (N>6) και «πιο μπερδεμένες» λίστες, τα print φαίνεται να επιβραδύνουν σημαντικά την εκτέλεση και άρα είναι προτιμότερο να μπούνε σε σχόλια.

```
DEBUG: list: [3, 4, 1, 2]  h(n)=2  g(n)=0          g(n)+h(n)=2
> selected state:[3, 4, 1, 2]       min(g(n)+h(n)):2

Searching . . . 1 extends so far
DEBUG: list: [4, 3, 1, 2]  h(n)=2  g(n)=1          g(n)+h(n)=3
DEBUG: list: [1, 4, 3, 2]  h(n)=1  g(n)=1          g(n)+h(n)=2
DEBUG: list: [2, 1, 4, 3]  h(n)=1  g(n)=1          g(n)+h(n)=2
> selected state:[1, 4, 3, 2]       min(g(n)+h(n)):2

Searching . . . 2 extends so far
DEBUG: list: [4, 3, 1, 2]  h(n)=2  g(n)=1          g(n)+h(n)=3
DEBUG: list: [2, 1, 4, 3]  h(n)=1  g(n)=1          g(n)+h(n)=2
DEBUG: list: [4, 1, 3, 2]  h(n)=2  g(n)=2          g(n)+h(n)=4
DEBUG: list: [3, 4, 1, 2]  h(n)=2  g(n)=2          g(n)+h(n)=4
DEBUG: list: [2, 3, 4, 1]  h(n)=2  g(n)=2          g(n)+h(n)=4
> selected state:[2, 1, 4, 3]       min(g(n)+h(n)):2

Searching . . . 3 extends so far
DEBUG: list: [4, 3, 1, 2]  h(n)=2  g(n)=1          g(n)+h(n)=3
DEBUG: list: [4, 1, 3, 2]  h(n)=2  g(n)=2          g(n)+h(n)=4
DEBUG: list: [3, 4, 1, 2]  h(n)=2  g(n)=2          g(n)+h(n)=4
DEBUG: list: [2, 3, 4, 1]  h(n)=2  g(n)=2          g(n)+h(n)=4
DEBUG: list: [1, 2, 4, 3]  h(n)=1  g(n)=2          g(n)+h(n)=3
DEBUG: list: [4, 1, 2, 3]  h(n)=2  g(n)=2          g(n)+h(n)=4
DEBUG: list: [3, 4, 1, 2]  h(n)=2  g(n)=2          g(n)+h(n)=4
> selected state:[4, 3, 1, 2]       min(g(n)+h(n)):3

Searching . . . 4 extends so far
DEBUG: list: [4, 1, 3, 2]  h(n)=2  g(n)=2          g(n)+h(n)=4
DEBUG: list: [3, 4, 1, 2]  h(n)=2  g(n)=2          g(n)+h(n)=4
DEBUG: list: [2, 3, 4, 1]  h(n)=2  g(n)=2          g(n)+h(n)=4
DEBUG: list: [1, 2, 4, 3]  h(n)=1  g(n)=2          g(n)+h(n)=3
DEBUG: list: [4, 1, 2, 3]  h(n)=2  g(n)=2          g(n)+h(n)=4
DEBUG: list: [3, 4, 1, 2]  h(n)=2  g(n)=2          g(n)+h(n)=4
DEBUG: list: [3, 4, 1, 2]  h(n)=2  g(n)=2          g(n)+h(n)=4
DEBUG: list: [1, 3, 4, 2]  h(n)=2  g(n)=2          g(n)+h(n)=4
DEBUG: list: [2, 1, 3, 4]  h(n)=1  g(n)=2          g(n)+h(n)=3
> selected state:[1, 2, 4, 3]       min(g(n)+h(n)):3
```

```
Searching . . . 5 extends so far
DEBUG: list: [4, 1, 3, 2]  h(n)=2  g(n)=2          g(n)+h(n)=4
DEBUG: list: [3, 4, 1, 2]  h(n)=2  g(n)=2          g(n)+h(n)=4
DEBUG: list: [2, 3, 4, 1]  h(n)=2  g(n)=2          g(n)+h(n)=4
DEBUG: list: [4, 1, 2, 3]  h(n)=2  g(n)=2          g(n)+h(n)=4
DEBUG: list: [3, 4, 1, 2]  h(n)=2  g(n)=2          g(n)+h(n)=4
DEBUG: list: [3, 4, 1, 2]  h(n)=2  g(n)=2          g(n)+h(n)=4
DEBUG: list: [1, 3, 4, 2]  h(n)=2  g(n)=2          g(n)+h(n)=4
DEBUG: list: [2, 1, 3, 4]  h(n)=1  g(n)=2          g(n)+h(n)=3
DEBUG: list: [2, 1, 4, 3]  h(n)=1  g(n)=3          g(n)+h(n)=4
DEBUG: list: [4, 2, 1, 3]  h(n)=1  g(n)=3          g(n)+h(n)=4
DEBUG: list: [3, 4, 2, 1]  h(n)=2  g(n)=3          g(n)+h(n)=5
> selected state:[2, 1, 3, 4]       min(g(n)+h(n)):3

Searching . . . 6 extends so far
DEBUG: list: [4, 1, 3, 2]  h(n)=2  g(n)=2          g(n)+h(n)=4
DEBUG: list: [3, 4, 1, 2]  h(n)=2  g(n)=2          g(n)+h(n)=4
DEBUG: list: [2, 3, 4, 1]  h(n)=2  g(n)=2          g(n)+h(n)=4
DEBUG: list: [4, 1, 2, 3]  h(n)=2  g(n)=2          g(n)+h(n)=4
DEBUG: list: [3, 4, 1, 2]  h(n)=2  g(n)=2          g(n)+h(n)=4
DEBUG: list: [3, 4, 1, 2]  h(n)=2  g(n)=2          g(n)+h(n)=4
DEBUG: list: [1, 3, 4, 2]  h(n)=2  g(n)=2          g(n)+h(n)=4
DEBUG: list: [2, 1, 4, 3]  h(n)=1  g(n)=3          g(n)+h(n)=4
DEBUG: list: [4, 2, 1, 3]  h(n)=1  g(n)=3          g(n)+h(n)=4
DEBUG: list: [3, 4, 2, 1]  h(n)=2  g(n)=3          g(n)+h(n)=5
DEBUG: list: [1, 2, 3, 4]  h(n)=0  g(n)=3          g(n)+h(n)=3
DEBUG: list: [3, 1, 2, 4]  h(n)=2  g(n)=3          g(n)+h(n)=5
DEBUG: list: [4, 3, 1, 2]  h(n)=2  g(n)=3          g(n)+h(n)=5
> selected state:[1, 2, 3, 4]       min(g(n)+h(n)):3


====================================================
State: 1 Cost: 0        [3, 4, 1, 2]      Initial State
State: 2 Cost: 1        [4, 3, 1, 2]      T(2)
State: 3 Cost: 2        [2, 1, 3, 4]      T(4)
State: 4 Cost: 3        [1, 2, 3, 4]      T(2)

Total states: 19 Extends: 6
====================================================
```

Στιγμιότυπο της αντίστοιχης εκτέλεσης με αναζήτηση UCS: (για σύγκριση)

```
Searching . . . 12 extends so far
=====================================================
State: 1 Cost: 0        [3, 4, 1, 2]      Initial State
State: 2 Cost: 1        [4, 3, 1, 2]      T(2)
State: 3 Cost: 2        [2, 1, 3, 4]      T(4)
State: 4 Cost: 3        [1, 2, 3, 4]      T(2)

Total states: 37 Extends: 12
=====================================================
```

Με βάση τα παραπάνω, μπορούμε να συγκρίνουμε την τιμή της ευρετικής συνάρτησης με την πραγματική ελάχιστη απόσταση που επιστρέφει η αναζήτηση UCS και να επιβεβαιώσουμε πως δεν την υπερβαίνει και άρα η ευρετική συνάρτηση είναι αποδεκτή. Με την ίδια λογική, επιβεβαιώσαμε το ίδιο και σε διάφορα άλλα πειράματα και συγκρίναμε τα αποτελέσματα των δύο μεθόδων. Παρατηρήσαμε πως η αναζήτηση A* πάντα βρίσκει μονοπάτι με κόστος ίδιο με αυτό που βρίσκει η UCS, δηλαδή το αποτέλεσμα είναι βέλτιστο και έτσι επιβεβαιώσαμε περαιτέρω πως η ευρετική συνάρτηση είναι αποδεκτή. Επιπλέον, παρατηρήσαμε πως η A* επεκτείνει σημαντικά λιγότερες καταστάσεις από την UCS και άρα απαιτεί λιγότερη υπολογιστική ισχύ για να καταλήξει στο ίδιο αποτέλεσμα. Η διαφορά είναι ιδιαίτερα εμφανής για μεγαλύτερο αριθμό θέσεων και «πιο μπερδεμένες» λίστες. Έτσι, <u>συμπεραίνουμε πως η A* είναι αρκετά πιο αποδοτική από την UCS</u>. Ακολουθούν κάποιες ενδεικτικές συγκρίσεις των δύο μεθόδων:

```
chris@DESKTOP-ANT:~/JAVA/AI$ java ask1 alphastar 3,2,5,1,4
Searching . . . 21 extends so far
=====================================================
State: 1        Cost: 0         [3, 2, 5, 1, 4] Initial State
State: 2        Cost: 1         [1, 5, 2, 3, 4] T(4)
State: 3        Cost: 2         [5, 1, 2, 3, 4] T(2)
State: 4        Cost: 3         [4, 3, 2, 1, 5] T(5)
State: 5        Cost: 4         [1, 2, 3, 4, 5] T(4)

Total states: 85        Extends: 21
=====================================================
chris@DESKTOP-ANT:~/JAVA/AI$ java ask1 ucs 3,2,5,1,4
Searching . . . 87 extends so far
=====================================================
State: 1        Cost: 0         [3, 2, 5, 1, 4] Initial State
State: 2        Cost: 1         [1, 5, 2, 3, 4] T(4)
State: 3        Cost: 2         [5, 1, 2, 3, 4] T(2)
State: 4        Cost: 3         [4, 3, 2, 1, 5] T(5)
State: 5        Cost: 4         [1, 2, 3, 4, 5] T(4)

Total states: 349       Extends: 87
=====================================================
```

```
chris@DESKTOP-ANT:~/JAVA/AI$ java ask1 alphastar 3,2,4,5,6,1
Searching . . . 6 extends so far
=====================================================
State: 1        Cost: 0         [3, 2, 4, 5, 6, 1]      Initial State
State: 2        Cost: 1         [2, 3, 4, 5, 6, 1]      T(2)
State: 3        Cost: 2         [6, 5, 4, 3, 2, 1]      T(5)
State: 4        Cost: 3         [1, 2, 3, 4, 5, 6]      T(6)

Total states: 31        Extends: 6
=====================================================
chris@DESKTOP-ANT:~/JAVA/AI$ java ask1 ucs 3,2,4,5,6,1
Searching . . . 37 extends so far
=====================================================
State: 1        Cost: 0         [3, 2, 4, 5, 6, 1]      Initial State
State: 2        Cost: 1         [2, 3, 4, 5, 6, 1]      T(2)
State: 3        Cost: 2         [6, 5, 4, 3, 2, 1]      T(5)
State: 4        Cost: 3         [1, 2, 3, 4, 5, 6]      T(6)

Total states: 186       Extends: 37
=====================================================
```

```
chris@DESKTOP-ANT:~/JAVA/AI$ java ask1 alphastar 5,2,7,4,1,3,6
Searching . . . 964 extends so far
=====================================================
State: 1        Cost: 0         [5, 2, 7, 4, 1, 3, 6]   Initial State
State: 2        Cost: 1         [7, 2, 5, 4, 1, 3, 6]   T(3)
State: 3        Cost: 2         [6, 3, 1, 4, 5, 2, 7]   T(7)
State: 4        Cost: 3         [2, 5, 4, 1, 3, 6, 7]   T(6)
State: 5        Cost: 4         [1, 4, 5, 2, 3, 6, 7]   T(4)
State: 6        Cost: 5         [5, 4, 1, 2, 3, 6, 7]   T(3)
State: 7        Cost: 6         [3, 2, 1, 4, 5, 6, 7]   T(5)
State: 8        Cost: 7         [1, 2, 3, 4, 5, 6, 7]   T(3)

Total states: 5785      Extends: 964
=====================================================
chris@DESKTOP-ANT:~/JAVA/AI$ java ask1 ucs 5,2,7,4,1,3,6
Searching . . . 4680 extends so far
=====================================================
State: 1        Cost: 0         [5, 2, 7, 4, 1, 3, 6]   Initial State
State: 2        Cost: 1         [7, 2, 5, 4, 1, 3, 6]   T(3)
State: 3        Cost: 2         [6, 3, 1, 4, 5, 2, 7]   T(7)
State: 4        Cost: 3         [2, 5, 4, 1, 3, 6, 7]   T(6)
State: 5        Cost: 4         [1, 4, 5, 2, 3, 6, 7]   T(4)
State: 6        Cost: 5         [5, 4, 1, 2, 3, 6, 7]   T(3)
State: 7        Cost: 6         [3, 2, 1, 4, 5, 6, 7]   T(5)
State: 8        Cost: 7         [1, 2, 3, 4, 5, 6, 7]   T(3)

Total states: 28081     Extends: 4680
=====================================================
```

```
chris@DESKTOP-ANT:~/JAVA/AI$ java ask1 alphastar 8,3,7,2,4,1,6,5
Searching . . . 4937 extends so far
=====================================================
State: 1        Cost: 0         [8, 3, 7, 2, 4, 1, 6, 5]        Initial State
State: 2        Cost: 1         [5, 6, 1, 4, 2, 7, 3, 8]        T(8)
State: 3        Cost: 2         [1, 6, 5, 4, 2, 7, 3, 8]        T(3)
State: 4        Cost: 3         [4, 5, 6, 1, 2, 7, 3, 8]        T(4)
State: 5        Cost: 4         [7, 2, 1, 6, 5, 4, 3, 8]        T(6)
State: 6        Cost: 5         [1, 2, 7, 6, 5, 4, 3, 8]        T(3)
State: 7        Cost: 6         [3, 4, 5, 6, 7, 2, 1, 8]        T(7)
State: 8        Cost: 7         [7, 6, 5, 4, 3, 2, 1, 8]        T(5)
State: 9        Cost: 8         [1, 2, 3, 4, 5, 6, 7, 8]        T(7)

Total states: 34560     Extends: 4937
=====================================================
chris@DESKTOP-ANT:~/JAVA/AI$ java ask1 ucs 8,3,7,2,4,1,6,5
Searching . . . 31922 extends so far
=====================================================
State: 1        Cost: 0         [8, 3, 7, 2, 4, 1, 6, 5]        Initial State
State: 2        Cost: 1         [3, 8, 7, 2, 4, 1, 6, 5]        T(2)
State: 3        Cost: 2         [7, 8, 3, 2, 4, 1, 6, 5]        T(3)
State: 4        Cost: 3         [1, 4, 2, 3, 8, 7, 6, 5]        T(6)
State: 5        Cost: 4         [5, 6, 7, 8, 3, 2, 4, 1]        T(8)
State: 6        Cost: 5         [2, 3, 8, 7, 6, 5, 4, 1]        T(6)
State: 7        Cost: 6         [4, 5, 6, 7, 8, 3, 2, 1]        T(7)
State: 8        Cost: 7         [8, 7, 6, 5, 4, 3, 2, 1]        T(5)
State: 9        Cost: 8         [1, 2, 3, 4, 5, 6, 7, 8]        T(8)

Total states: 223455    Extends: 31922
=====================================================
```

```
chris@DESKTOP-ANT:~/JAVA/AI$ java ask1 alphastar 1,2,3,7,9,4,5,6,8
Searching . . . 814 extends so far
=====================================================
State: 1        Cost: 0         [1, 2, 3, 7, 9, 4, 5, 6, 8]     Initial State
State: 2        Cost: 1         [7, 3, 2, 1, 9, 4, 5, 6, 8]     T(4)
State: 3        Cost: 2         [6, 5, 4, 9, 1, 2, 3, 7, 8]     T(8)
State: 4        Cost: 3         [3, 2, 1, 9, 4, 5, 6, 7, 8]     T(7)
State: 5        Cost: 4         [9, 1, 2, 3, 4, 5, 6, 7, 8]     T(4)
State: 6        Cost: 5         [8, 7, 6, 5, 4, 3, 2, 1, 9]     T(9)
State: 7        Cost: 6         [1, 2, 3, 4, 5, 6, 7, 8, 9]     T(8)

Total states: 6513      Extends: 814
=====================================================
chris@DESKTOP-ANT:~/JAVA/AI$ java ask1 ucs 1,2,3,7,9,4,5,6,8
Searching . . . 22704 extends so far
=====================================================
State: 1        Cost: 0         [1, 2, 3, 7, 9, 4, 5, 6, 8]     Initial State
State: 2        Cost: 1         [3, 2, 1, 7, 9, 4, 5, 6, 8]     T(3)
State: 3        Cost: 2         [9, 7, 1, 2, 3, 4, 5, 6, 8]     T(5)
State: 4        Cost: 3         [8, 6, 5, 4, 3, 2, 1, 7, 9]     T(9)
State: 5        Cost: 4         [7, 1, 2, 3, 4, 5, 6, 8, 9]     T(8)
State: 6        Cost: 5         [6, 5, 4, 3, 2, 1, 7, 8, 9]     T(7)
State: 7        Cost: 6         [1, 2, 3, 4, 5, 6, 7, 8, 9]     T(6)

Total states: 181633    Extends: 22704
=====================================================
```