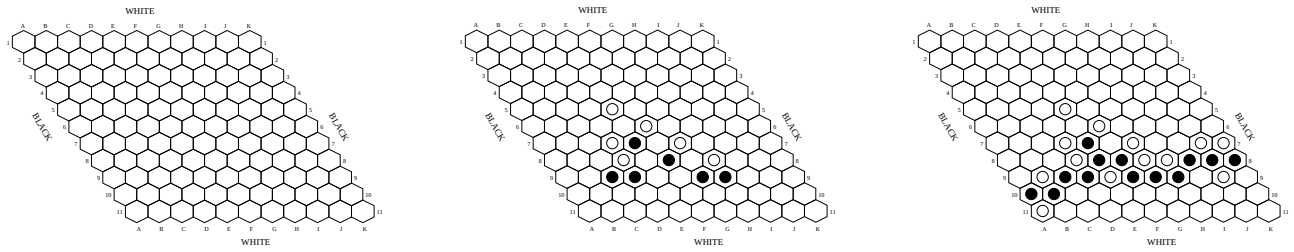
ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ (2018-19)

Εργασία 4

Στην εργασία αυτή, καλείσθε να υλοποιήσετε ένα πρόγραμμα C που θα παίζει, με αντίπαλο έναν άνθρωπο (τον χρήστη του προγράμματος), το παιγνίδι *Hex*.

Το Hex παίζεται σε ένα πλαίσιο από εξάγωνα, με διάσταση συνήθως 11×11, αλλά μπορεί να παιχθεί και σε οποιαδήποτε διάσταση $N \times N$, με $N \geq 4$. Ο ένας παίκτης είναι ο λευκός και ο άλλος ο μαύρος. Κάθε παίκτης έχει στη διάθεσή του έναν επαρκή αριθμό από κυκλικά πλακίδια του χρώματός του. Επίσης, κάθε ζευγάρι απέναντι πλευρών του πλαισίου ανήκει σε καθένα από τους δύο παίκτες. Στο παιγνίδι, οι παίκτες παίζουν εναλλάξ, με τον λευκό να παίζει πρώτος, τοποθετώντας ένα πλακίδιο του χρώματός τους σε κάποια ελεύθερη θέση του πλαισίου που επιθυμούν. Ο στόχος τους είναι να σχηματίσουν μία συνεχή ακολουθία από πλακίδια του χρώματός τους, σε γειτονικά εξάγωνα, που να συνδέει τις δύο πλευρές του πλαισίου που τους ανήκουν. Όποιος παίκτης το κατορθώσει πρώτος, είναι ο νικητής.

Στο σχήμα (α) παρακάτω φαίνεται το πλαίσιο του παιγνιδιού πριν αρχίσει η εξέλιξή του. Στο (β) φαίνεται η κατάσταση του παιγνιδιού, αφού οι παίκτες έχουν τοποθετήσει εναλλάξ κάποια πλακίδια. Στο (γ) έχει κερδίσει ο μαύρος παίκτης, γιατί έχει κατασκευάσει μία συνεχή ακολουθία από γειτονικά μαύρα πλακίδια που συνδέει τις πλευρές που του ανήκουν.



(α) (β) (γ)

Επειδή στο παιγνίδι αυτό ο παίκτης που παίζει πρώτος έχει πλεονέκτημα, μερικές φορές, συμφωνείται μεταξύ των δύο παικτών να εφαρμόσουν και ένα κανόνα που ονομάζεται *κανόνας ανταλλαγής* (swap rule). Σύμφωνα με τον κανόνα αυτό, μετά από την πρώτη κίνηση του λευκού παίκτη, ο μαύρος έχει το δικαίωμα, αν θέλει, να γίνει αυτός πρώτος παίκτης με πρώτη κίνηση τη συμμετρική αυτής που έπαιξε ο λευκός παίκτης. Με τον όρο "συμμετρική", εννοείται η αντιμετάθεση στήλης και γραμμής σε σχέση με την κίνηση του λευκού. Για παράδειγμα, αν ο λευκός είχε παίξει στο D2 την πρώτη του κίνηση, ο μαύρος έχει το δικαίωμα να την ακυρώσει, όταν ισχύει ο κανόνας της ανταλλαγής, και να παίξει εκείνος σαν πρώτη κίνηση στο B4. Στη συνέχεια, παίζει ο λευκός παίκτης, σαν δεύτερος.

Το Hex είναι ένα παιγνίδι στρατηγικής και το να αναπτυχθούν αλγόριθμοι ώστε να μπορεί ένας υπολογιστής να παίζει καλά αυτό το παιγνίδι είναι αντικείμενο ενός κλάδου της Πληροφορικής, που λέγεται *Τεχνητή Νοημοσύνη* (Artificial Intelligence). Περισσότερες πληροφορίες για το παιγνίδι μπορείτε να βρείτε στον σύνδεσμο `http://en.wikipedia.org/wiki/Hex_(board_game)`.

Το πρόγραμμα που θα γράψετε (έστω ότι το εκτελέσιμο ονομάζεται "`hex`") θα μπορεί να δέχεται από τη γραμμή εντολών κάποιες επιλογές, για τον καθορισμό αρχικών παραμέτρων, και αφού ξεκινήσει να εκτελείται, θα πρέπει να δέχεται εντολές από ένα προκαθορισμένο ρεπερτόριο που θα καθορίζουν τη λειτουργία του.

Οι επιλογές από τη γραμμή εντολών είναι οι εξής:

`-n <size>` : Το τυπικό Hex παίζεται σε πλαίσιο 11×11. Το πρόγραμμά σας θα πρέπει να είναι σε θέση να παίζει το παιγνίδι για οποιοδήποτε μέγεθος της διάστασης του πλαισίου (`<size>`) μεταξύ 4 και 26 (συμπεριλαμβανομένων), το οποίο καθορίζεται από την επιλογή αυτή. Αν δεν δοθεί η επιλογή, να θεωρείται ότι `<size> = 11`.

`-d <difficulty>` : Ορίζεται το επίπεδο δυσκολίας στο οποίο θα παίζεται το παιγνίδι από το πρόγραμμα που θα γράψετε. Αν δεν δοθεί η επιλογή, να θεωρείται `<difficulty> = 1`, που είναι το χαμηλότερο επίπεδο δυσκολίας. Περισσότερα για το επίπεδο δυσκολίας, στη συνέχεια.

`-b` : Ο χρήστης θα είναι ο μαύρος παίκτης, οπότε παίζει δεύτερος. Αν δεν δοθεί η επιλογή, ο χρήστης θα είναι ο λευκός παίκτης, οπότε παίζει πρώτος.

`-s` : Ο κανόνας ανταλλαγής είναι ενεργός. Αν δεν δοθεί η επιλογή, ο κανόνας ανταλλαγής δεν εφαρμόζεται.

Αφού ξεκινήσει να εκτελείται το πρόγραμμα που θα γράψετε, να δέχεται εντολές που καθορίζουν παραμέτρους, αλλά και τη λειτουργία του παιγνιδιού. Οι εντολές αυτές είναι:

`newgame [white|black [swapoff|swapon [<size>]]]` : Αρχίζει εξ αρχής νέο παιγνίδι. Αν δεν δοθεί καμία παράμετρος, το χρώμα του χρήστη, η ισχύς του κανόνα ανταλλαγής και το μέγεθος του παιγνιδιού προκύπτουν από το τι δόθηκε στη γραμμή εντολών. Αν δοθεί μία παράμετρος, αυτή πρέπει να είναι κάποιο από τα `white` ή `black`, που είναι το χρώμα που θα έχει ο χρήστης. Αν δοθεί, εκτός της πρώτης, και δεύτερη παράμετρος, αυτή πρέπει να είναι κάποιο από τα `swapoff` ή `swapon`. Στην πρώτη περίπτωση, δεν εφαρμόζεται ο κανόνας ανταλλαγής, ενώ στη δεύτερη εφαρμόζεται. Αν δοθεί, εκτός των δύο πρώτων, και τρίτη παράμετρος, αυτή είναι το μέγεθος του παιγνιδιού.

`play <move>` : Η εντολή είναι αποδεκτή όταν είναι η σειρά του χρήστη να παίξει. Το `<move>` είναι η κίνηση που επιθυμεί να κάνει ο παίκτης, στη μορφή `<letter><number>`, τις συντεταγμένες του εξαγώνου που επιθυμεί να τοποθετήσει πλακίδιο του χρώματός του, όπου `<letter>` είναι κάποιο από τα (`A`, `B`, `C`, ...) και `<number>` είναι κάποιο από τα (`1`, `2`, `3`, ...).

`cont` : Η εντολή είναι αποδεκτή όταν είναι η σειρά του προγράμματος να παίξει, οπότε γίνεται η κίνηση που αυτό αποφασίζει.

`undo` : Ο χρήστης αναιρεί την προηγούμενη κίνησή του. Αν έχει γίνει ήδη και η κίνηση του προγράμματος, αναιρείται και αυτή. Επιτρέπονται και συνεχόμενες `undo`.

`suggest` : Ο χρήστης ζητά από το πρόγραμμα να του υποδείξει την κίνηση που θα ήταν καλό να κάνει. Η εντολή είναι αποδεκτή όταν είναι η σειρά του να παίξει.

`level [<difficulty>]` : Ο χρήστης επιλέγει σαν επίπεδο δυσκολίας το `<difficulty>`. Αν δεν δοθεί `<difficulty>`, το πρόγραμμα εκτυπώνει ποιο είναι τρέχον επίπεδο δυσκολίας του παιγνιδιού.

`swap` : Αν ο χρήστης παίζει δεύτερος, με την εντολή αυτή μπορεί να υποδείξει, μόνο στην πρώτη του κίνηση, και εφ΄ όσον ισχύει ο κανόνας ανταλλαγής, ότι επιθυμεί να γίνει "ανταλλαγή".

`save <statefile>` : Αποθηκεύει την τρέχουσα κατάσταση του παιγνιδιού στο αρχείο `<statefile>`.

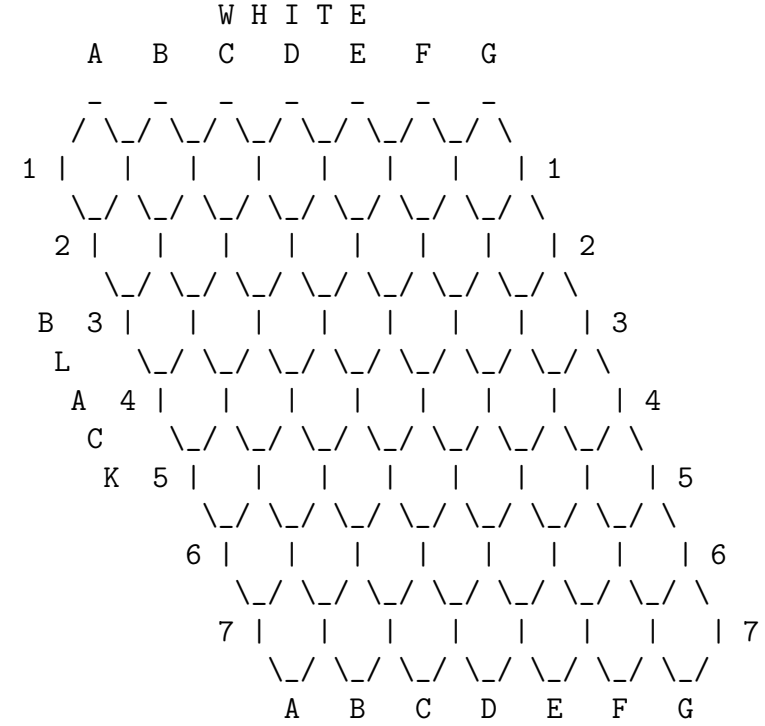
`load <statefile>` : Φορτώνει από το αρχείο `<statefile>` μία αποθηκευμένη κατάσταση παιγνιδιού. Περισσότερα για τη μορφή των αρχείων καταστάσεων, στη συνέχεια.

`showstate` : Εκτυπώνεται η τρέχουσα κατάσταση του παιγνιδιού.

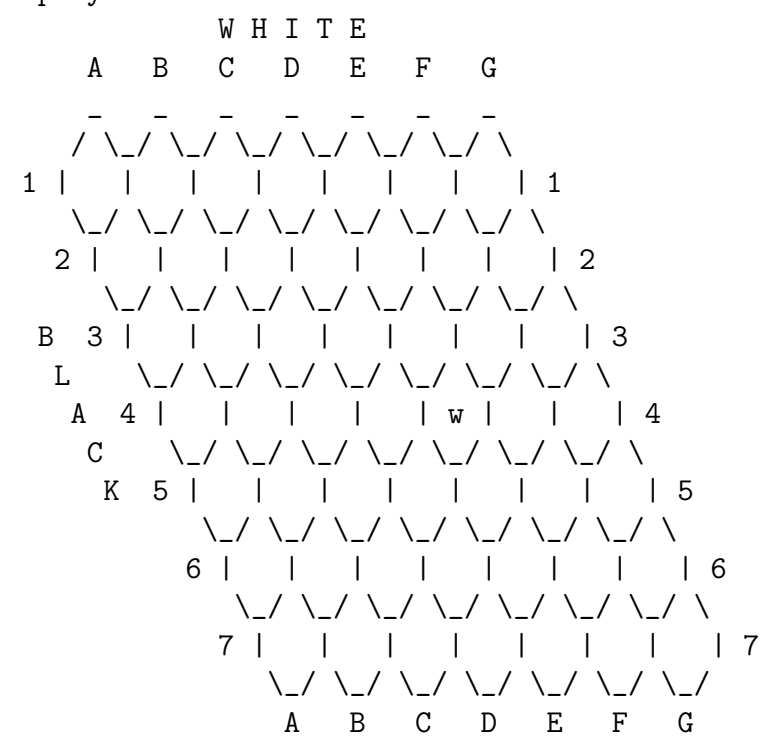
`quit` : Τερματίζει το πρόγραμμα.

Μία ενδεικτική εκτέλεση του προγράμματος φαίνεται στη συνέχεια:

```
$ ./hex -n 7 -d 2
> newgame
```

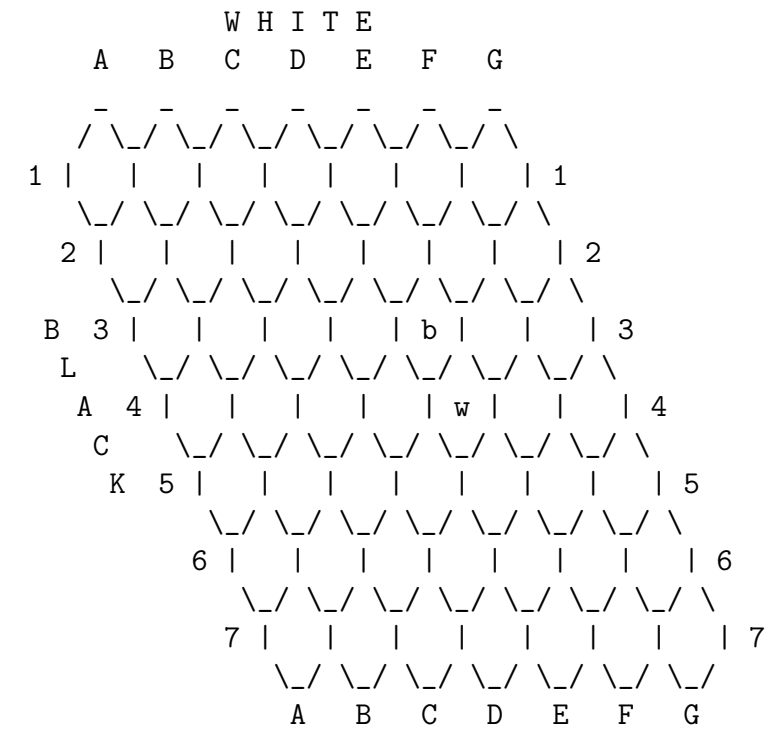



```
White player (human) plays now
> play E4
```

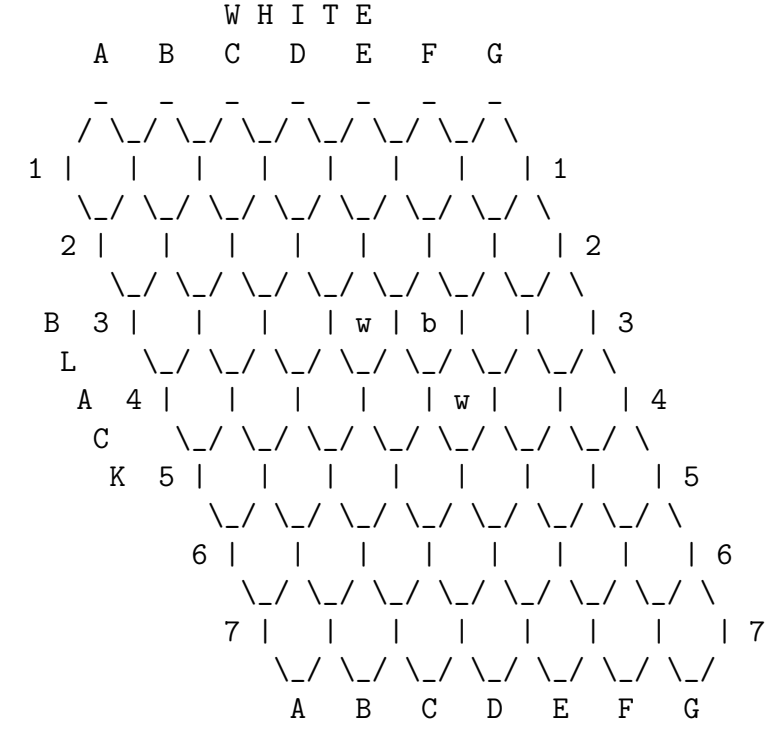



```
Move played: E4
Black player (computer) plays now
```

```
> cont
```

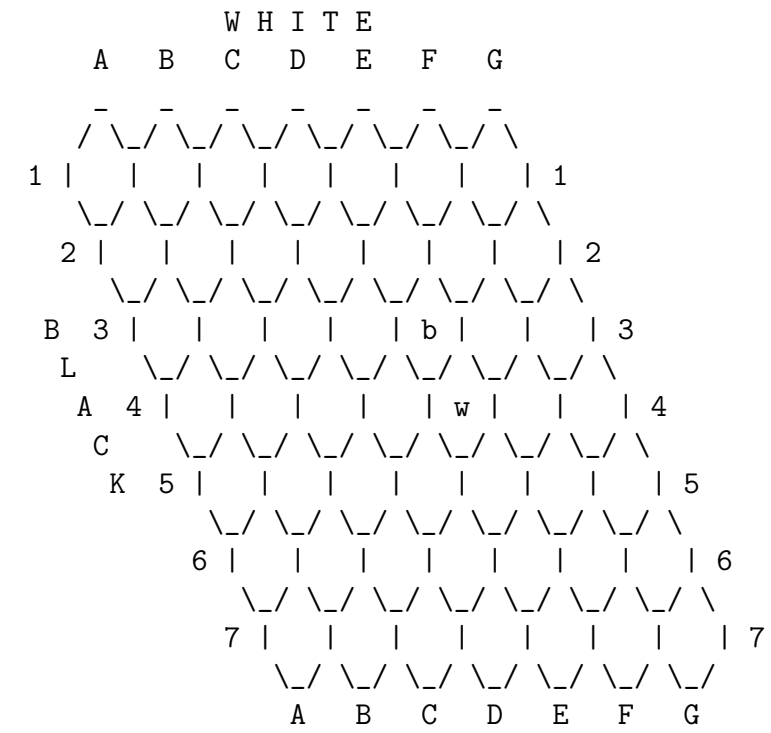



```
Move played: E3
White player (human) plays now
> play D3
```

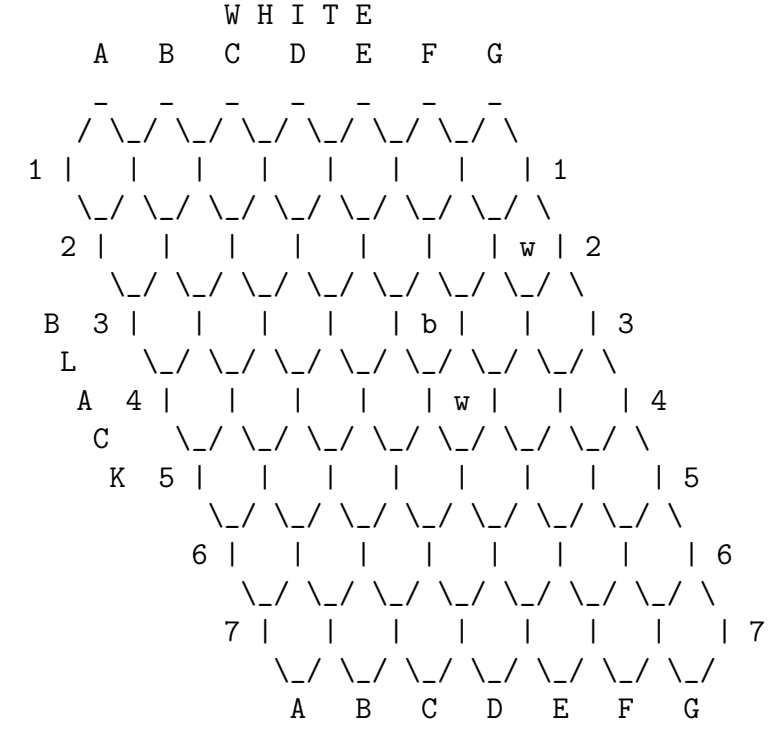



```
Move played: D3
Black player (computer) plays now
```

```
> undo
```




```
White player (human) plays now
> play G2
```




```
Move played: G2
Black player (computer) plays now
> save file.hex
> quit
```

```
$ ./hex
> load file.hex
> .................
```

Άλλη μία ενδεικτική εκτέλεση του προγράμματος, για παιγνίδι σε πλαίσιο 4×4, είναι η εξής:

```
$ ./hex
> newgame black swapon 4
       W H I T E
     A   B   C   D
     _   _   _   _
    / \_/ \_/ \_/ \
 1 |   |   |   |   | 1
B   \_/ \_/ \_/ \_/ \
 L 2 |   |   |   |   | 2
  A   \_/ \_/ \_/ \_/ \
   C 3 |   |   |   |   | 3
    K   \_/ \_/ \_/ \_/ \
       4 |   |   |   |   | 4
          \_/ \_/ \_/ \_/
           A   B   C   D

White player (computer) plays now
> level 3
> cont
       W H I T E
     A   B   C   D
     _   _   _   _
    / \_/ \_/ \_/ \
 1 |   |   |   |   | 1
B   \_/ \_/ \_/ \_/ \
 L 2 |   |   |   |   | 2
  A   \_/ \_/ \_/ \_/ \
   C 3 |   |   |   | w | 3
    K   \_/ \_/ \_/ \_/ \
       4 |   |   |   |   | 4
          \_/ \_/ \_/ \_/
           A   B   C   D

Move played: D3
Black player (human) plays now
```

```
> swap
       W H I T E
     A   B   C   D

     _   _   _   _
    / \_/ \_/ \_/ \
 1 |   |   |   |   | 1
B   \_/ \_/ \_/ \_/ \
 L 2 |   |   |   |   | 2
  A   \_/ \_/ \_/ \_/ \
   C 3 |   |   |   |   | 3
    K   \_/ \_/ \_/ \_/ \
       4 |   |   | b |   | 4
          \_/ \_/ \_/ \_/
           A   B   C   D

Move played: swap
White player (computer) plays now
> cont
       W H I T E
     A   B   C   D

     _   _   _   _
    / \_/ \_/ \_/ \
 1 |   |   |   |   | 1
B   \_/ \_/ \_/ \_/ \
 L 2 |   |   |   |   | 2
  A   \_/ \_/ \_/ \_/ \
   C 3 |   | w |   |   | 3
    K   \_/ \_/ \_/ \_/ \
       4 |   |   | b |   | 4
          \_/ \_/ \_/ \_/
           A   B   C   D

Move played: B3
Black player (human) plays now
> play B4
       W H I T E
     A   B   C   D

     _   _   _   _
    / \_/ \_/ \_/ \
 1 |   |   |   |   | 1
B   \_/ \_/ \_/ \_/ \
 L 2 |   |   |   |   | 2
  A   \_/ \_/ \_/ \_/ \
   C 3 |   | w |   |   | 3
    K   \_/ \_/ \_/ \_/ \
       4 |   | b | b |   | 4
          \_/ \_/ \_/ \_/
           A   B   C   D
```

```
Move played: B4
White player (computer) plays now
> cont
       W H I T E
     A   B   C   D
     _   _   _   _
    / \_/ \_/ \_/ \
 1 |   |   |   |   | 1
B   \_/ \_/ \_/ \_/ \
 L 2 |   |   |   |   | 2
  A   \_/ \_/ \_/ \_/ \
   C 3 |   | w |   |   | 3
    K   \_/ \_/ \_/ \_/ \
       4 | w | b | b |   | 4
          \_/ \_/ \_/ \_/
           A   B   C   D

Move played: A4
Black player (human) plays now
> play C1
       W H I T E
     A   B   C   D
     _   _   _   _
    / \_/ \_/ \_/ \
 1 |   |   | b |   | 1
B   \_/ \_/ \_/ \_/ \
 L 2 |   |   |   |   | 2
  A   \_/ \_/ \_/ \_/ \
   C 3 |   | w |   |   | 3
    K   \_/ \_/ \_/ \_/ \
       4 | w | b | b |   | 4
          \_/ \_/ \_/ \_/
           A   B   C   D

Move played: C1
White player (computer) plays now
```

```
> cont
       W H I T E
     A   B   C   D

     _   _   _   _
    / \_/ \_/ \_/ \
 1 |   |   | b |   | 1
B   \_/ \_/ \_/ \_/ \
 L 2 | w |   |   |   | 2
  A   \_/ \_/ \_/ \_/ \
   C 3 |   | w |   |   | 3
    K   \_/ \_/ \_/ \_/ \
       4 | w | b | b |   | 4
          \_/ \_/ \_/ \_/
           A   B   C   D

Move played: A2
Black player (human) plays now
> suggest
You may play at B1
> showstate
       W H I T E
     A   B   C   D

     _   _   _   _
    / \_/ \_/ \_/ \
 1 |   |   | b |   | 1
B   \_/ \_/ \_/ \_/ \
 L 2 | w |   |   |   | 2
  A   \_/ \_/ \_/ \_/ \
   C 3 |   | w |   |   | 3
    K   \_/ \_/ \_/ \_/ \
       4 | w | b | b |   | 4
          \_/ \_/ \_/ \_/
           A   B   C   D

Black player (human) plays now
> play B1
       W H I T E
     A   B   C   D

     _   _   _   _
    / \_/ \_/ \_/ \
 1 |   | b | b |   | 1
B   \_/ \_/ \_/ \_/ \
 L 2 | w |   |   |   | 2
  A   \_/ \_/ \_/ \_/ \
   C 3 |   | w |   |   | 3
    K   \_/ \_/ \_/ \_/ \
       4 | w | b | b |   | 4
          \_/ \_/ \_/ \_/
           A   B   C   D
```

```
Move played: B1
White player (computer) plays now
> cont
       W H I T E
     A   B   C   D
     _   _   _   _
    / \_/ \_/ \_/ \
 1 | w | b | b |   | 1
B   \_/ \_/ \_/ \_/ \
 L 2 | w |   |   |   | 2
  A   \_/ \_/ \_/ \_/ \
   C 3 |   | w |   |   | 3
    K   \_/ \_/ \_/ \_/ \
       4 | w | b | b |   | 4
          \_/ \_/ \_/ \_/
           A   B   C   D

Move played: A1
Black player (human) plays now
> play A3
       W H I T E
     A   B   C   D
     _   _   _   _
    / \_/ \_/ \_/ \
 1 | w | b | b |   | 1
B   \_/ \_/ \_/ \_/ \
 L 2 | w |   |   |   | 2
  A   \_/ \_/ \_/ \_/ \
   C 3 | b | w |   |   | 3
    K   \_/ \_/ \_/ \_/ \
       4 | w | b | b |   | 4
          \_/ \_/ \_/ \_/
           A   B   C   D

Move played: A3
White player (computer) plays now
```

```
> cont
       W H I T E
     A   B   C   D
     _   _   _   _
    / \_/ \_/ \_/ \
 1 | w | b | b |   | 1
B   \_/ \_/ \_/ \_/ \
 L 2 | w | w |   |   | 2
  A   \_/ \_/ \_/ \_/ \
   C 3 | b | w |   |   | 3
    K   \_/ \_/ \_/ \_/ \
       4 | w | b | b |   | 4
          \_/ \_/ \_/ \_/
           A   B   C   D

Move played: B2

End of game
White player (computer) wins with path A1-A2-B2-B3-A4
> quit
```

Όσον αφορά τη μορφή των αρχείων καταστάσεων, που χειρίζονται οι εντολές `save` και `load`, αυτή πρέπει να είναι η εξής: Αρχικά, μέσα στο αρχείο υπάρχει (σε δυαδική μορφή) ένας ακέραιος, του ενός `byte`, που είναι η διάσταση του πλαισίου (έστω n), μετά μία ένδειξη, σε ένα byte, για το ποιος παίκτης έχει σειρά να παίξει (`'w'` ή `'b'`, ανάλογα) και μετά n^2 bytes, τα οποία κωδικοποιούν τα περιεχόμενα των θέσεων του πλαισίου, κατά γραμμές. Για τις θέσεις με λευκό πλακίδιο υπάρχει το `'w'`, για εκείνες με μαύρο πλακίδιο το `'b'` και για τις κενές θέσεις το `'n'` (ASCII κωδικοί). Το αρχείο στο οποίο αποθηκεύτηκε η κατάσταση του παιγνιδιού, κατά την πρώτη από τις προηγούμενες ενδεικτικές εκτελέσεις, είναι το `http://www.di.uoa.gr/~ip/hwfiles/hex/file.hex`.

Επίπεδο δυσκολίας - Ο αλγόριθμος minimax

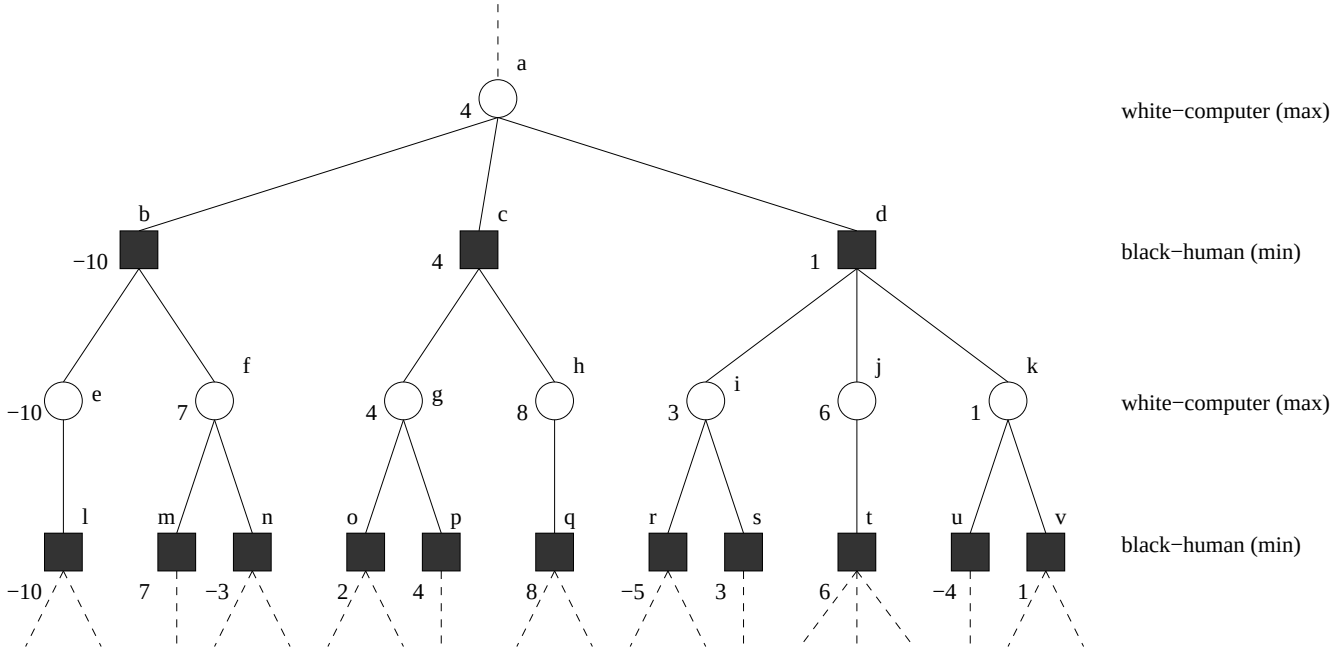
Ένα θέμα που θα πρέπει να συζητηθεί εδώ σε κάποιο βαθμό είναι αυτό του επιπέδου δυσκολίας στο οποίο θα καλείται να παίξει το πρόγραμμα που θα γράψετε.

Έστω ότι το πρόγραμμα βρίσκεται στο σημείο που πρέπει να αποφασίσει τι κίνηση να κάνει. Πώς όμως θα επιλέξει σε ποια ελεύθερη θέση του πλαισίου να τοποθετήσει πλακίδιο του χρώματός του; Προφανώς πρέπει να είναι εκείνη που θα το οδηγήσει στην ευνοϊκότερη δυνατή κατάσταση για το χρώμα του. Οπότε χρειάζεται ένα μέτρο του πόσο καλή είναι μία κατάσταση παιγνιδιού για κάποιο παίκτη δεδομένου χρώματος. Ένα πολύ απλό τέτοιο μέτρο για το Hex είναι η διαφορά του μήκους της μεγαλύτερης ακολουθίας από συνεχόμενα πλακίδια του χρώματος του παίκτη μείον το αντίστοιχο μήκος ακολουθίας για τον αντίπαλο. Είναι αντικείμενο της άσκησης το να σκεφτείτε χρήσιμα μέτρα της ποιότητας μίας κατάστασης, τα οποία να συνδυάσετε κατάλληλα ώστε να προκύψει μία όσο το δυνατόν περισσότερο αξιόπιστη συνισταμένη.

Επιστρέφοντας τώρα στο πρόβλημα της επιλογής από το πρόγραμμα της καταλληλότερης κίνησης μεταξύ όλων των δυνατών, μία, σχετικά απλοϊκή, προσέγγιση, είναι να επιλεγεί εκείνη η κίνηση που οδηγεί σε κατάσταση με την καλύτερη ποιότητα, για τον παίκτη-υπολογιστή. Θα μπορούσαμε τότε να πούμε ότι το πρόγραμμα παίζει σε επίπεδο δυσκολίας 1. Η προσέγγιση αυτή χαρακτηρίζεται σαν απλοϊκή, διότι δίνει κάποιο πρόσκαιρο πλεονέκτημα στο πρόγραμμα, όμως μπορεί να οδηγεί σε κατάσταση όπου ο παίκτησ-χρήστης να έχει διαθέσιμη κάποια πολύ καίρια κίνηση, η οποία να ανατρέπει

υπέρ του την ισορροπία του παιγνιδιού. Μήπως θα έπρεπε το πρόγραμμα να προβλέψει κάτι τέτοιο και να μην κάνει τελικά τη λάθος κίνηση; Με άλλα λόγια, δεν θα μπορούσε, αντί να λαμβάνει υπόψη του το τι μπορεί να συμβεί μετά από μία κίνηση στην τρέχουσα κατάσταση, να λαμβάνει υπόψη και τις δύο επόμενες κινήσεις (επίπεδο 2); Και γιατί να κάνει πρόβλεψη μόνο για δύο κινήσεις και όχι για τρεις; Καταλαβαίνετε ότι αυτή η λογική γενικεύεται, ώστε να μπορεί το πρόγραμμα να εξερευνά τα πιθανά εναλλακτικά για τις d επόμενες κινήσεις, οπότε τότε λέμε ότι παίζει σε επίπεδο δυσκολίας d.

Η μέθοδος που περιγράφηκε πιο πάνω σε γενικές γραμμές είναι ο αλγόριθμος minimax που έχει αναπτυχθεί για παιγνίδια δύο παικτών, στο πλαίσιο της Τεχνητής Νοημοσύνης. Για να καταλάβετε καλύτερα τη λογική αυτού του αλγορίθμου, δείτε το επόμενο σχήμα που παριστάνει τμήμα ενός *δέντρου παιγνιδιού* (game tree).



Έστω ότι κάποια στιγμή το παιγνίδι βρίσκεται στην κατάσταση a, στην οποία είναι η σειρά του παίκτη-υπολογιστή να παίξει με τα λευκά πλακίδια, και έστω ότι οι δυνατές κινήσεις που μπορεί να κάνει θα οδηγούσαν το παιγνίδι σε κάποια από τις καταστάσεις b, c ή d. Αν το πρόγραμμα έπαιζε σε επίπεδο δυσκολίας 1, απλά θα αποτιμούσε την ποιότητα αυτών των τριών καταστάσεων και θα έκανε την κίνηση που θα ήταν η πιο συμφέρουσα γι΄ αυτό. Έστω όμως ότι έχει καθορισθεί το πρόγραμμα να παίζει σε επίπεδο δυσκολίας 3. Τότε θα πρέπει να πάρει την απόφασή του προβλέποντας τι μπορεί να συμβεί μετά από τρεις κινήσεις. Οπότε, για κάθε πιθανή επόμενη κατάσταση από τις b, c ή d, στις οποίες είναι σειρά του παίκτη-χρήστη να παίξει με τα μαύρα πλακίδια, πρέπει το πρόγραμμα να βρει τις επόμενες πιθανές καταστάσεις, στις οποίες θα είναι η σειρά του παίκτη-υπολογιστή να παίξει και, τέλος, για κάθε μία από αυτές, να βρει τις πιθανές επόμενες, στις οποίες θα είναι η σειρά του παίκτη-χρήστη να παίξει. Αυτές οι καταστάσεις φαίνονται στο τελευταίο επίπεδο του σχήματος.

Αυτό που πρέπει να κάνει το πρόγραμμα είναι να αποτιμήσει τις καταστάσεις του τελευταίου επιπέδου και να διαδώσει προς τα επάνω τις αποτιμήσεις με την εξής λογική. Σε κάθε κατάσταση που είναι η σειρά του παίκτη-υπολογιστή να παίξει με τα άσπρα πλακίδια, η αποτίμησή της είναι το μέγιστο των αποτιμήσεων των καταστάσεων που μπορεί να προκύψουν από αυτήν (γι΄ αυτό, ο παίκτησ-υπολογιστής ονομάζεται max). Ο λόγος είναι ότι αν βρεθεί το παιγνίδι σε μία τέτοια κατάσταση και είναι η σειρά του παίκτη-υπολογιστή να παίξει, το πρόγραμμα θα πρέπει να επιλέξει την πιο συμφέρουσα κίνηση. Αντίστροφα, σε κάθε κατάσταση που είναι η σειρά του παίκτη-χρήστη να παίξει με τα μαύρα πλακίδια, η αποτίμησή της είναι το ελάχιστο των αποτιμήσεων των καταστάσεων που μπορεί να προκύψουν από αυτήν (γι΄ αυτό, ο παίκτησ-χρήστης ονομάζεται min). Η λογική είναι ότι ο παίκτησ-χρήστης θα κάνει

την λιγότερο συμφέρουσα κίνηση για τον παίκτη-υπολογιστή.[1]

Στο προηγούμενο σχήμα, οι αριθμοί στους κόμβους του τελευταίου επιπέδου είναι οι αποτιμήσεις των αντίστοιχων καταστάσεων, όπως αυτές υπολογίζονται από τη συνάρτηση που έχει επιλεγεί για να εκτιμά την ποιότητα των καταστάσεων από την πλευρά του παίκτη-υπολογιστή. Οι αριθμοί στους ενδιάμεσους κόμβους είναι οι αποτιμήσεις των αντίστοιχων καταστάσεων, πάντα από την πλευρά του παίκτη-υπολογιστή, όπως αυτές διαδίδονται από το τελευταίο επίπεδο προς τα ανώτερα. Δηλαδή, οι αριθμοί που υπάρχουν στους ενδιάμεσους κόμβους δεν είναι η αποτίμηση της δικής τους ποιότητας, αλλά προκύπτουν από τα παιδιά τους. Για παράδειγμα, i = max{r,s} = max{-5,3} = 3, k = max{u,v} = max{-4,1} = 1, οπότε d = min{i,j,k} = min{3,6,1} = 1. Τελικά, για το δεδομένο παράδειγμα, όταν η κατάσταση παιγνιδιού είναι η a, η κίνηση που συμφέρει το πρόγραμμα να κάνει είναι αυτή που το οδηγεί στην κατάσταση c.

Επίσης, για πληροφόρησή σας, ας αναφερθεί ότι υπάρχει και ένας αλγόριθμος για παιγνίδια δύο παικτών, που αποτελεί μία βελτιωμένη εκδοχή του minimax, ο αλγόριθμος των *άλφα-βήτα αποκοπών* (alpha-beta pruning).[2] Σημειώνεται ότι δεν είναι μέσα στις απαιτήσεις της άσκησης να υλοποιήσετε τον αλγόριθμό αυτό, αλλά αναφέρεται γιατί μπορεί να σας χρησιμεύσει αν επιθυμείτε να συμμετάσχει η άσκησή σας με αξιώσεις στο κύπελλο Hex που περιγράφεται στο τέλος της εκφώνησης.

Παραδοτέο

Θα πρέπει να δομήσετε το πρόγραμμά σας σε ένα σύνολο από **τουλάχιστον τρία πηγαία αρχεία** C (με κατάληξη `.c`) και **τουλάχιστον δύο αρχεία επικεφαλίδας** (με κατάληξη `.h`). Επίσης, δημιουργήστε ένα απλό αρχείο κειμένου με όνομα `README.txt`, στο οποίο να δίνετε οδηγίες για τη μεταγλώττιση του προγράμματος, καθώς και ό,τι άλλο κρίνετε σκόπιμο να επισημάνετε. Προαιρετικά, μπορείτε να παραδώσετε και ένα αρχείο `Makefile` που να αναλαμβάνει όλη τη διαδικασία της κατασκευής των τελικών εκτελέσιμων μέσω της εντολής "`make`" (δώστε "`man make`" για περισσότερες λεπτομέρειες).

Για να παραδώσετε το σύνολο των αρχείων που θα έχετε δημιουργήσει για την εργασία αυτή, ακολουθήστε την εξής διαδικασία. Τοποθετήστε όλα τα αρχεία μέσα σ΄ ένα κατάλογο που θα δημιουργήσετε σε κάποιο σύστημα Linux, έστω με όνομα `hex`. Όντας στον κατάλογο που περιέχει τον κατάλογο `hex`, δημιουργήστε ένα επιπεδοποιημένο `tar` αρχείο (έστω με όνομα `hex.tar`) που περιέχει τον κατάλογο `hex` και όλα του τα περιεχόμενα. Αυτό γίνεται με την εντολή "`tar cvf hex.tar hex`".[3] Συμπιέστε το αρχείο `hex.tar`, ώστε να δημιουργηθεί το αρχείο `hex.tar.gz`. Αυτό γίνεται με την εντολή "`gzip hex.tar`".[4] Το αρχείο `hex.tar.gz` είναι που θα πρέπει να υποβάλετε μέσω του eclass.[5]

Ομαδική εργασία

Η εργασία αυτή μπορεί να παραδοθεί και από **ομάδες των δύο ατόμων**. Στην περίπτωση αυτή, θα παραδοθεί μόνο από το ένα μέλος της ομάδας, αλλά μέσα στο αρχείο `README.txt` θα αναφέρονται σαφώς τα στοιχεία των δύο μελών. Ο στόχος της διαδικασίας αυτής είναι να ενισχυθεί η ιδέα της **ισότιμης** συνεργασίας σε μία ομάδα για την επίτευξη ενός στόχου. Αν τα μέλη της ομάδας έχουν υλοποιήσει διαφορετικά τμήματα της εργασίας, θα πρέπει στο αρχείο `README` να αναφέρεται ρητά τι

[1] Στους χαρακτηρισμούς των δύο παικτών, ως max και min, οφείλεται και το όνομα του αλγορίθμου (minimax).

[2] Για περισσότερες πληροφορίες, στο `http://en.wikipedia.org/wiki/Alpha-beta_pruning`.

[3] Αν θέλετε να ανακτήσετε την δενδρική δομή που έχει φυλαχθεί σε ένα επιπεδοποιημένο `tar` αρχείο `file.tar`, αυτό μπορεί να γίνει με την εντολή "`tar xvf file.tar`".

[4] Αν θέλετε να αποσυμπιέσετε ένα αρχείο `file.gz` που έχει συμπιεσθεί με την εντολή `gzip`, αυτό μπορεί να γίνει με την εντολή "`gzip -d file.gz`".

[5] Μην υποβάλετε ασυμπίεστα αρχεία ή αρχεία που είναι συμπιεσμένα σε άλλη μορφή εκτός από `tar.gz` (π.χ. `rar`, `7z`, `zip`, κλπ.), γιατί δεν θα γίνονται δεκτά για αξιολόγηση.

έχει υλοποιήσει κάθε μέλος, έτσι ώστε στην προφορική εξέταση που θα ακολουθήσει, να μην υπάρχει η απαίτηση να έχει κάποιο μέλος της ομάδας πλήρη γνώση του πώς έχουν υλοποιηθεί τα τμήματα στα οποία εκείνο δεν έχει εμπλακεί.

Κύπελλο Hex (BONUS βαθμολογία)

Οι εργασίες που θα υποβληθούν θα μπορούν να συμμετάσχουν, εφ΄ όσον το δηλώσουν οι συγγραφείς τους, σε κύπελλο Hex που θα διεξαχθεί, με σύστημα knock out, μετά την τελευταία ημέρα της προφορικής εξέτασης των εργασιών. Απαραίτητη προϋπόθεση για να διεξαχθεί το κύπελλο είναι να υπάρξουν τουλάχιστον οκτώ συμμετοχές. Για τις τρεις καλύτερες εργασίες, θα υπάρξει επιβράβευση στη βαθμολογία τους κατά 100%, 70% και 40%, κατά σειρά. Περισσότερες λεπτομέρειες για το κύπελλο θα ανακοινωθούν εν καιρώ στο φόρουμ του μαθήματος.

Τα προγράμματα που θα συμμετάσχουν στο κύπελλο Hex πρέπει να πληρούν τις ακόλουθες προϋποθέσεις:

- Το πρόγραμμα πρέπει, αφού εκτυπώσει ό,τι χρειάζεται μετά από κάθε κίνηση (π.χ. αμέσως πριν την ανάγνωση της επόμενης εντολής), να κάνει flush την έξοδό του, με την εντολή "`fflush(stdout);`".

- Το πρόγραμμα πρέπει να αναφέρει σε μία γραμμή της εξόδου του ποια κίνηση έγινε μετά από κάθε εντολή `cont` αλλά και `play`. Η γραμμή αυτή πρέπει να περιέχει τις δύο λέξεις `Move` και `played`, να έχει το σύμβολο της άνω και κάτω τελείας και μετά να περιέχει την κίνηση (π.χ. `Move played: D3`). Η κίνηση πρέπει να είναι ένα κεφαλαίο γράμμα ακολουθούμενο από ένα ή δύο ψηφία, χωρίς κενά ανάμεσα (σωστές κινήσεις: `A2`, `F23`, λάθος κινήσεις: `a2`, `F 23`, κλπ.). Επιπροσθέτως, το πρόγραμμα μπορεί, αν παίζει δεύτερο, να επιλέξει "ανταλλαγή" στην πρώτη του κίνηση. Τότε θα πρέπει στην αναφορά του, αντί για συντεταγμένες θέσης, να υπάρχει η λέξη `swap` (δηλαδή, `Move played: swap`).

- Το πρόγραμμα πρέπει οπωσδήποτε να υλοποιεί τις επιλογές `-n`, `-d`, `-b` και `-s` της γραμμής εντολών και τις εσωτερικές εντολές `newgame`, `play` και `cont`.

- Το πρόγραμμα πρέπει να είναι σε θέση να ανταποκρίνεται σωστά σε πιθανή εντολή "ανταλλαγής" από τον αντίπαλό του, ανεξάρτητα αν το ίδιο έχει υλοποιήσει τη δυνατότητα αυτή για τον εαυτό του.

- Σε περίπτωση που το πρόγραμμα χρησιμοποιεί τυχαιότητα για την επιλογή των κινήσεών του, θα πρέπει να ορίζει το φύτρο της γεννήτριας τυχαίων αριθμών σε μία σταθερή τιμή μέσα στον κώδικα, ώστε να είναι δυνατόν να αναπαραχθούν τα αποτελέσματά του.

- Τέλος, το πρόγραμμα πρέπει, τουλάχιστον όταν παίζει στο επίπεδο 1, να δίνει την κίνηση που αποφάσισε να κάνει σε λιγότερο από μισό λεπτό (πραγματικός χρόνος), όταν εκτελείται σε κάποιο Linux μηχάνημα του εργαστηρίου του Τμήματος, που δεν έχει φόρτο λειτουργίας από την εκτέλεση άλλων προγραμμάτων. Επιπρόσθετα, αν πάρουμε το άθροισμα των πραγματικών χρόνων που δαπάνησε το πρόγραμμα για να κάνει όλες τις κινήσεις του σε παιγνίδι Hex διάστασης N, αυτό δεν θα πρέπει να υπερβαίνει τα $N/2$ λεπτά.