ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών
ΙΔΡΥΘΕΝ ΤΟ 1837

Εισαγωγή στον Προγραμματισμό

Εργασία #3

Ιανουάριος 2026

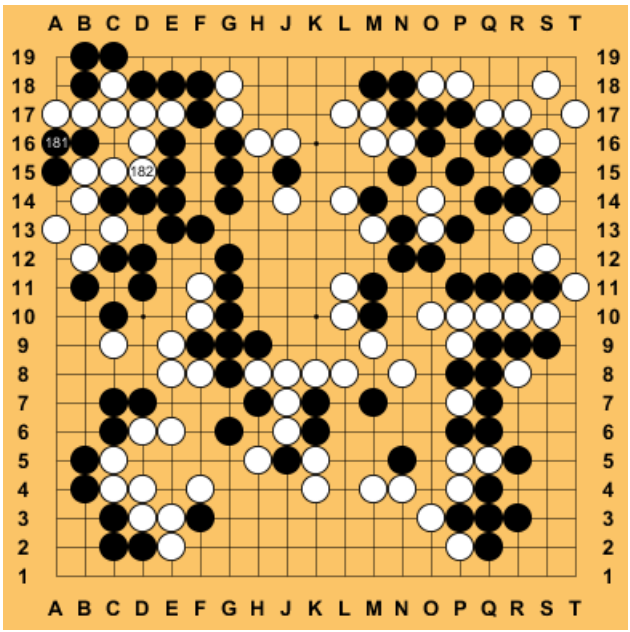
Στόχος της τρίτης και τελευταίας εργασίας είναι να εξοικειωθούμε με την υλοποίηση μεγαλύτερων και πιθανώς πιο πολύπλοκων προγραμμάτων, ολοκληρώνοντας την εισαγωγή μας στην γλώσσα C. Συγκεκριμένα, στοχεύουμε σε εμπειρία με τα ακόλουθα:

1. Γραφή μεγαλύτερων προγραμμάτων, αποτελούμενα από πολλά αρχεία.
2. Χρήση όλων των βασικών προγραμματιστικών δομών.
3. Αναζήτηση σε χώρο καταστάσεων.

Υποβολή Εργασίας. Όλες οι υποβολές των εργασιών θα γίνουν μέσω GitHub και συγκεκριμένα στο github.com/progintro [2]. Προκειμένου να ξεκινήσεις, μπορείς να δεχτείς την άσκηση με αυτήν την: πρόσκληση [3]. Σε αντίθεση με τις προηγούμενες εργασίες, αυτή είναι **προαιρετικά ομαδική με ομάδες μέχρι 2 άτομα**. Εάν θέλετε να δουλέψετε ως ομάδα, πρέπει ένας/μία από εσάς να αποδεχτεί την πρόσκληση και να δημιουργήσει την ομάδα σας **με συγκεκριμένο όνομα**. Αφού η ομάδα δημιουργηθεί ο/η συνεργάτης σας μπορεί να αποδεχτεί την πρόσκληση και να επιλέξει να προστεθεί στην ομάδα σας. **Προσοχή: μην επιλέξετε λάθος ομάδα, βεβαιωθείτε ότι κάνατε join την σωστή.** Αν επιλέξετε να εργαστείτε ως ομάδα, το repository για την εργασία θα είναι κοινό ανάμεσά σας και θα μπορέσετε να συνεργαστείτε σαν επαγγελματίες software engineers - προσοχή στα conflicts! Τέλος, για να είναι ξεκάθαρο ποια άτομα συνεργάστηκαν για την εργασία, **κάθε repository πρέπει να έχει ένα AUTHORS αρχείο** το οποίο θα περιέχει τα στοιχεία σας στην ακόλουθη μορφή:

```
$ cat AUTHORS
sdi2500998,mourmourakis-2007,ΘΑΝΟΣ ΜΟΥΡΜΟΥΡΑΚΗΣ
sdi2500999,lavrakis-2007,ΤΑΚΗΣ ΛΑΒΡΑΚΗΣ
```

δηλαδή μία γραμμή για κάθε άτομο, με πρώτο το sdi σας, μετά το github username και τέλος το όνομά σας. **Υποβολές χωρίς σωστό AUTHORS αρχείο δεν θα εξεταστούν**. Αυτό ισχύει και για ατομικές υποβολές.



Σχήμα 1: Το παραδοσιακό παιχνίδι Go, με ιστορία άνω των 2.000 ετών, παίζεται σε ένα τετράγωνο ταμπλό 19x19 γραμμών (δηλαδή 361 σημεία τοποθέτησης), με δύο παίκτες που εναλλάσσονται τοποθετώντας μαύρες και άσπρες πέτρες. Οι κανόνες του είναι εντυπωσιακά απλοί και μπορούν να εξηγηθούν μέσα σε λίγα λεπτά [16, 7], όμως η στρατηγική του πολυπλοκότητα είναι τεράστια: ο αριθμός των πιθανών θέσεων στο Go εκτιμάται ότι ξεπερνά το 10^{170}, πολύ περισσότερο από τον αντίστοιχο του σκακιού.

1. Νέα Μηχανή Go - goteam (Μονάδες: 360)

Τα περισσότερα παιχνίδια έχουν παρεμφερή δομή: (1) το παιχνίδι ξεκινάει σε μια αρχική κατάσταση, (2) ένας από τους παίκτες παίζει πρώτος επιλέγοντας μια από τις δυνατές κινήσεις που έχει στην διάθεσή του αλλάζοντας την κατάσταση του παιχνιδιού και (3) στην συνέχεια δίνει την σειρά του στον επόμενο παίκτη. Η εναλλαγή των παικτών συνεχίζεται μέχρι κάποιο κριτήριο (διαφορετικό για κάθε παιχνίδι) να μας υποδείξει ότι το παιχνίδι τελείωσε με νίκη, ισοπαλία ή ήττα για την κάθε πλευρά.

Η πολυπλοκότητα του κάθε παιχνιδιού [14] έχει άμεση σχέση με δύο παραμέτρους: (1) πόσες δυνατές κινήσεις έχει σε κάθε γύρο ο παίκτης και (2) πόσους γύρους μπορεί να έχει ένα παιχνίδι. Αυτές οι δύο παράμετροι μας επιτρέπουν να υπολογίσουμε τον χώρο καταστάσεων του παιχνιδιού (δηλαδή σε πόσες διαφορετικές καταστάσεις μπορεί να βρεθεί το παιχνίδι μας). Ένας τρόπος να φανταστούμε αυτόν τον χώρο καταστάσεων είναι σαν ένα m-αδικό δέντρο βάθους n, όπου m οι επιλογές μας σε κάθε θέση και n ο αριθμός των επιλογών (γύρων) του παιχνιδιού. Ένα m-αδικό δέντρο βάθους n έχει περίπου m^n διαφορετικές καταστάσεις—όμως πόσο γρήγορα μεγαλώνει ένα τέτοιο δέντρο; Για να δούμε μερικά παραδείγματα παιχνιδιών.

Ίσως έχετε παίξει τρίλιξα. Στην τρίλιξα ξεκινάμε με ένα πλέγμα 3x3 και στην αρχή έχουμε 9 επιλογές για το που θα τοποθετήσουμε το ο ή το x. Στην χειρότερη περίπτωση, μετά από 9 γύρους το παιχνίδι μας θα έχει τελειώσει καθώς και οι 9 θέσεις θα έχουν γεμίσει. Επομένως, ένα απλοϊκό άνω όριο για τον χώρο καταστάσεων είναι $9^9 = 387.420.489$ (στην πραγματικότητα ο χώρος καταστάσεων είναι πολύ μικρότερος—γύρω στις 765 [4]—καθώς σε κάθε κίνηση μειώνεται ο αριθμός των επιλογών μας, τα παιχνίδια μπορεί να ολοκληρωθούν πολύ πιο γρήγορα από 9 κινήσεις και κάποια παιχνίδια καταλήγουν στην ίδια κατάσταση με διαφορετική αρχική σειρά κινήσεων). Παρόλο που ο αριθμός αυτός είναι αρκετά μεγάλος για έναν άνθρωπο—θα μας έπαιρνε πολλές ώρες να παίξουμε όλες τις δυνατές παρτίδες τρίλιξας και να υπολογίσουμε τα αποτελέσματά τους—ο ίδιος υπολογισμός μπορεί να γίνει σε λιγότερο από ένα δευτερόλεπτο από έναν σύγχρονο υπολογιστή. Εάν διατρέξουμε ολόκληρο τον χώρο καταστάσεων ενός παιχνιδιού τότε μπορούμε να πούμε στην επιστήμη των υπολογιστών ότι το "λύσαμε" [19], δηλαδή για οποιαδήποτε θέση του παιχνιδιού μπορούμε να πούμε με βεβαιότητα ποια είναι η βέλτιστη κίνηση.

Χάρη στην εξέλιξη των υπολογιστών, έχουμε καταφέρει να λύσουμε αρκετά δημοφιλή παιχνίδια με μεγάλους χώρους καταστάσεων. Για παράδειγμα η "ντάμα" (Checkers / Draughts) με χώρο καταστάσεων γύρω στο 10^{20} έχει λυθεί επίσημα από το 2007, όπου ερευνητές κατάφεραν να ολοκληρώσουν τον υπολογισμό ύστερα από δεκαετίες προσπάθειας [9]. Φυσικά κανένας δεν θυμάται την καλύτερη κίνηση για όλες τις 10^{20} καταστάσεις οπότε παρόλο που το πρόβλημα λύθηκε οι άνθρωποι μπορούν ακόμα να απολαμβάνουν αυτό το παιχνίδι σε παρτίδες μεταξύ τους.

Ένα άλλο παιχνίδι που για δεκαετίες ερευνητές στον χώρο της τεχνητής νοημοσύνης προσπαθούσαν να λύσουν είναι το σκάκι. Ένας από τους λόγους που ακόμα δεν έχουμε λύση είναι το τεράστιο μέγεθος του χώρου καταστάσεων (με ένα πλέγμα μόλις 8x8!). Ο διάσημος μαθηματικός Claude Shannon εκτίμησε τις δυνατές παρτίδες στο 10^{120} και τις πιθανές θέσεις της σκακιέρας στο 10^{40} [18]—μεγέθη που είναι ακόμα άπιαστα ακόμα και σήμερα για τα ισχυρότερα υπολογιστικά συστήματα. Παρά την αδυναμία εξερεύνησης όλου του χώρου καταστάσεων, αξιοποιώντας κατάλληλες ευριστικές, οι πρώτες επιτυχίες ήρθαν το 1997 με τον DeepBlue [11] όπου κέρδισε τον τότε πρωταθλητή κόσμου Garry Kasparov.

Με ένα πλέγμα διαστάσεων 19x19 και έναν χώρο καταστάσεων που εκτιμάται στο 10^{170} (!) και τις δυνατές παρτίδες στο 10^{1048} (‼) όμως, υπήρχε ακόμα ένα παιχνίδι που θεωρούνταν άφταστο για την τεχνητή νοημοσύνη—το παιχνίδι Go [15]. Το Go είναι ένα από τα αρχαιότερα και βαθύτερα επιτραπέζια παιχνίδια στρατηγικής που έχουν επινοηθεί ποτέ. Η καταγωγή του τοποθετείται στην Κίνα πριν από περισσότερα από 2.500 χρόνια, γεγονός που το καθιστά παλαιότερο ακόμη και από το σκάκι.

Περισσότερο από παιχνίδι, το Go θεωρείται πολιτισμικό και φιλοσοφικό σύστημα σκέψης. Στην Ιαπωνία, την Κορέα και την Κίνα αναπτύχθηκαν επί αιώνες σχολές, επαγγελματικοί τίτλοι και επίσημα πρωταθλήματα, με επαγγελματίες παίκτες να αφιερώνουν δεκαετίες μελέτης. Το παιχνίδι διδάσκει έννοιες όπως η ισορροπία, η υπομονή και η μακροπρόθεσμη σκέψη: συχνά μια θυσία λίγων πετρών οδηγεί σε

Σχήμα 2: Το 2016, ο AlphaGo της DeepMind κέρδισε 4-1 τον Lee Sedol—τον κορυφαίο Κορεάτη επαγγελματία παίκτη Go σε ένα συναρπαστικό αγώνα [20]. Το ματς ανέδειξε τη ραγδαία πρόοδο της τεχνητής νοημοσύνης και άνοιξε ευρύτερες συζητήσεις για τη σχέση ανθρώπου και μηχανής στη λήψη αποφάσεων και τη δημιουργική σκέψη.

στρατηγικό πλεονέκτημα σε ολόκληρο το ταμπλώ. Γι' αυτό και το Go συνεχίζει, μετά από χιλιάδες χρόνια, να γοητεύει μαθηματικούς, φιλοσόφους και επιστήμονες υπολογιστών σε όλο τον κόσμο.

Μπορούν όμως οι αλγόριθμοι να αναμετρηθούν με την ανθρώπινη σκέψη σε ένα τέτοιο παιχνίδι που ούτως ή άλλως δεν μπορείς να εξαντλήσεις τον χώρο καταστάσεων και αυτό που έχει σημασία είναι η "διαίσθηση"; Η απάντηση ήταν γενικά όχι μέχρι το 2016. Το 2016, ο Κορεάτης Lee Sedol—ένας από τους κορυφαίους παίκτες Go στον κόσμο—αναμετρήθηκε με τον AlphaGo [8], το σύστημα τεχνητής νοημοσύνης της DeepMind [12]. Ο AlphaGo τελικά κέρδισε με σκορ 4–1 και ανέτρεψε την πεποίθηση ότι παιχνίδια με τεράστια πολυπλοκότητα και έντονη διαίσθηση βρίσκονταν εκτός των δυνατοτήτων των υπολογιστών. Παρά την ήττα του, ο Lee Sedol κέρδισε παγκόσμιο σεβασμό, ιδιαίτερα για τη μία του νίκη, η οποία ανέδειξε τόσο τη δημιουργικότητα του ανθρώπου όσο και τα όρια των αλγορίθμων. Ο Ντέμης Χασάμπης [13]—ο άνθρωπος πίσω από την ομάδα του AlphaGo—συνέχισε να δουλεύει σε αντίστοιχα προβλήματα τεχνητής νοημοσύνης (όπως η αναδίπλωση πρωτεϊνών) και πέρυσι μάλιστα βραβεύθηκε με βραβείο Νόμπελ (Χημείας!) για την προσφορά του [21].

Πόσο εύκολο είναι να φτιάξουμε μια μηχανή Go σήμερα; Αυτό θα είναι και το αντικείμενο αυτής της άσκησης, όπου καλείστε να υλοποιήσετε τον πυρήνα μιας μηχανής Go ικανό να παίζει με αντίπαλο είτε έναν άνθρωπο, τον χρήστη του προγράμματος, είτε ένα άλλο πρόγραμμα, μέσω ενός ελεγκτή/διαιτητή. Ευτυχώς, δεν χρειάζεται να ξεκινήσετε από το μηδέν, μπορείτε να βρείτε στο διαδίκτυο πολλές πληροφορίες για το πως δομούνται μηχανές Go [10]. Οι τεχνικές προδιαγραφές ακολουθούν.

2. Τεχνικές Προδιαγραφές

- Repository Name: progintro/hw3-<YourTeamName>
- README Filepath: README.md
- Όλα τα αρχεία κώδικα (.c και .h) που θα υλοποιήσετε πρέπει να τοποθετηθούν μέσα σε έναν φάκελο `src`.
- Το project C που θα υποβληθεί πρέπει να μεταγλωττίζεται χωρίς ειδοποιήσεις για λάθη και με κωδικό επιστροφής (exit code) που να είναι 0. Συγκεκριμένα, το αρχείο σας **πρέπει** να μπορεί να μεταγλωττιστεί επιτυχώς με την ακόλουθη εντολή σε ένα από τα μηχανήματα του εργαστηρίου (linuxXY.di.uoa.gr):

  ```
  make
  ```

  Στο repository της άσκησης σας δίνεται μια πιθανή οργάνωση κώδικα μαζί με ένα Makefile [17] το οποίο μπορείτε να συμπληρώσετε όπως θέλετε. Μπορείτε να αλλάξετε τα πάντα στο project σας, αλλά θέλουμε να επιβεβαιώσετε πως η εντολή make συνεχίζει να δουλεύει και να παράγει εκτελέσιμα για την Go μηχανή σας. **Το εκτελέσιμό σας πρέπει να λέγεται "goteam"**—άλλα ονόματα δεν θα γίνουν δεκτά. Ακολουθήστε τις συμβάσεις που δίνονται στο αρχικό Makefile.
- Το πρόγραμμά σας πρέπει να υποστηρίζει τις εντολές του πρωτοκόλλου Go Text Protocol (GTP) [6] προκειμένου να επικοινωνεί με τον χρήστη ή το πρόγραμμα διαιτητή. Δείτε τις ενδεικτικές εκτελέσεις στην συνέχεια για τις βασικές δυνατότητες που πρέπει να υποστηρίζει.
- Υπάρχουν διαφορές εκδοχές κανόνων για το Go, στα πλαίσια της άσκησης θα ακολουθήσουμε την πιο δημοφιλή εκδοχή, δηλαδή τους κινέζικους κανόνες [1].
- Προκειμένου να συμμετέχετε στον διαγωνισμό Go που θα γίνει στο τέλος του εξαμήνου, οι εντολές του πρωτοκόλλου GTP που θα υλοποιήσετε πρέπει να είναι επαρκείς για να είναι σε θέση το πρόγραμμά σας να συνεργάζεται με τους ελεγκτές/διαιτητές που θα βρείτε στην διεύθυνση:

  ```
  https://github.com/progintro/goref
  ```

  Προκειμένου να κατεβάσετε την τελευταία έκδοση (**προσοχή: θα υπάρξουν αρκετές ανανεώσεις του goref** μέχρι το τέλος του διαγωνισμού προκειμένου να διαχειριστούμε την όποια ... εφευρετικότητα έχουν οι λύσεις σας) του goref τοπικά, μπορείτε να τρέξετε τις ακόλουθες εντολές:

  ```
  curl -O https://raw.githubusercontent.com/progintro/goref/main/goref.py
  chmod +x goref.py
  ```

Μία ενδεικτική εκτέλεση του προγράμματος (έστω ότι το εκτελέσιμο ονομάζεται goteam) είναι η εξής:

```
$ ./goteam
boardsize 5
=

komi 0.5
=

clear_board
=

showboard
=
   A B C D E
 5 . . . . . 5
 4 . . . . . 4
 3 . . . . . 3
 2 . . . . . 2
 1 . . . . . 1
   A B C D E

genmove black
= C3

showboard
=
   A B C D E
 5 . . . . . 5
 4 . . . . . 4
 3 . . X . . 3
 2 . . . . . 2
 1 . . . . . 1
   A B C D E

play white C4
=

genmove black
= B4

showboard
```

```
=
   A B C D E
 5 . . . . . 5
 4 . X O . . 4
 3 . . X . . 3
 2 . . . . . 2
 1 . . . . . 1
   A B C D E

play white B3
=

genmove black
= B2

showboard
=
   A B C D E
 5 . . . . . 5
 4 . X O . . 4
 3 . O X . . 3
 2 . X . . . 2
 1 . . . . . 1
   A B C D E

play white D3
=

genmove black
= A3

showboard
=
   A B C D E
 5 . . . . . 5
 4 . X O . . 4
 3 X . X O . 3
 2 . X . . . 2
 1 . . . . . 1
   A B C D E

play white D5
```

```
=

genmove black
= E4

showboard
=
   A B C D E
 5 . . . O . 5
 4 . X O . X 4
 3 X . X O . 3
 2 . X . . . 2
 1 . . . . . 1
   A B C D E

play white C2
=

genmove black
= B3

showboard
=
   A B C D E
 5 . . . O . 5
 4 . X O . X 4
 3 X X X O . 3
 2 . X O . . 2
 1 . . . . . 1
   A B C D E

play white D1
=

genmove black
= D4

showboard
=
   A B C D E
 5 . . . O . 5
 4 . X O X X 4
```

```
 3 X X X O . 3
 2 . X O . . 2
 1 . . . O . 1
   A B C D E

play white E3
=

genmove black
= C5

showboard
=
   A B C D E
 5 . . X O . 5
 4 . X . X X 4
 3 X X X O O 3
 2 . X O . . 2
 1 . . . O . 1
   A B C D E

play white E2
=

genmove black
= B1

showboard
=
   A B C D E
 5 . . X O . 5
 4 . X . X X 4
 3 X X X O O 3
 2 . X O . O 2
 1 . X . O . 1
   A B C D E

play white C1
=

genmove black
= PASS
```

```
showboard
=
   A B C D E
 5 . . X O . 5
 4 . X . X X 4
 3 X X X O O 3
 2 . X O . O 2
 1 . X O O . 1
   A B C D E

play white pass
=

final_score
= B+8.5

quit
=

$
```

Κύπελλο Go (BONUS βαθμολογία). Οι μηχανές που θα υποβληθούν θα συμμετάσχουν σε κύπελλο Go που θα διεξαχθεί μετά το τέλος της προθεσμίας. Για τις τρεις καλύτερες υλοποιήσεις, θα υπάρξει επιβράβευση στη βαθμολογία τους κατά 100%, 70% και 40%, κατά σειρά. Μπορείτε να δοκιμάσετε τις δυνατότητες της υλοποίησής σας χρησιμοποιώντας προγράμματα διαιτητές όπως το goref παραπάνω ως εξής:

```
$ ./goref.py --black "gnugo --mode gtp --chinese-rules" \
     --white "./goteam" --games 10 --main-time 60 --verbose
[blackcmd:B >>] name
[blackcmd:B <<] = GNU Go
[blackcmd:B <<]
...
[whitecmd:W >>] name
[whitecmd:W <<] = Go Team
[whitecmd:W <<]
[whitecmd:W(Go Team) >>] version
[whitecmd:W(Go Team) <<] = 1.0
[whitecmd:W(Go Team) <<]
...
[whitecmd:W(Go Team) v1.0 >>] time_settings 60 0 0
[whitecmd:W(Go Team) v1.0 <<] =
```

```
[whitecmd:W(Go Team) v1.0 <<]
...
[whitecmd:W(Go Team) v1.0 >>] time_left B 60 0
[whitecmd:W(Go Team) v1.0 <<] =
[whitecmd:W(Go Team) v1.0 <<]
[blackcmd:B(GNU Go) v3.8 >>] genmove B
[blackcmd:B(GNU Go) v3.8 <<] = Q16
[blackcmd:B(GNU Go) v3.8 <<]
[whitecmd:W(Go Team) v1.0 >>] play B Q16
[whitecmd:W(Go Team) v1.0 <<] =
[whitecmd:W(Go Team) v1.0 <<]
...
[whitecmd:W(Go Team) v1.0 >>] time_left W 60 0
[whitecmd:W(Go Team) v1.0 <<] =
[whitecmd:W(Go Team) v1.0 <<]
...
[whitecmd:W(Go Team) v1.0 >>] genmove W
[whitecmd:W(Go Team) v1.0 <<] = D4
[whitecmd:W(Go Team) v1.0 <<]
...
Games: 10
Overall (by command slot):
  black_cmd: 8
  white_cmd: 2
  draws: 0
...
```

Η παραπάνω εκτέλεση ελέγχει την υλοποίησή μας με λευκά ενάντια στην μηχανή GNU Go με μαύρα (`apt install gnugo` σε σύστημα Ubuntu) και μας επιτρέπει να ελέγξουμε την πληρότητα αλλά και την αποδοτικότητα της υλοποίησής μας. Να αναφέρουμε εδώ ότι αν θέλετε να οπτικοποιήσετε τα αποτελέσματα της μηχανής σας (ή να παίξετε ένα παιχνίδι με graphical user interface) υπάρχουν αρκετές υλοποιήσεις ανοιχτού κώδικα που μπορείτε να χρησιμοποιήσετε [5].

Επίλογος

Τέλος, πριν την υποβολή της εργασίας, μην ξεχάσετε το README.md, μέσα στο οποίο πρέπει να συμπεριλάβετε τις όποιες παρατηρήσεις σας κατά την διεκπεραίωση της άσκησης! Ο κώδικας απαιτείται να είναι καλά τεκμηριωμένος με σχόλια καθώς αυτό θα είναι μέρος της βαθμολόγησης.

Καλές Γιορτές ευχόμαστε!

Αναφορές

[1] Carnegie Mellon University. The chinese rules of go. https://www.cs.cmu.edu/~wjh/go/rules/Chinese.html.

[2] DIT. Οργανισμός για το μάθημα (GitHub progintro) . https://github.com/progintro.

[3] DIT. Πρόσκληση για Εργασία 3 . https://classroom.github.com/a/UWzEAYS3.

[4] Gary Fredericks. Tic Tac Toe Visualizations. https://gfredericks.com/blog/76.

[5] GitHub. Graphical User Interface for Go. https://github.com/Remi-Coulom/gogui.

[6] Linköpings Universitet. Go Text Protocol. https://www.lysator.liu.se/~gunnar/gtp/.

[7] Sensei's Library. Introduction to Go. https://senseis.xmp.net/?RulesOfGoIntroductory.

[8] Wikipedia. AlphaGo. https://en.wikipedia.org/wiki/AlphaGo.

[9] Wikipedia. Checkers. https://en.wikipedia.org/wiki/Checkers.

[10] Wikipedia. Computer Go. https://en.wikipedia.org/wiki/Computer_Go.

[11] Wikipedia. Deep Blue. https://en.wikipedia.org/wiki/Deep_Blue_(chess_computer).

[12] Wikipedia. DeepMind. https://en.wikipedia.org/wiki/Google_DeepMind.

[13] Wikipedia. Demis Hassabis. https://en.wikipedia.org/wiki/Demis_Hassabis.

[14] Wikipedia. Game Complexity. https://en.wikipedia.org/wiki/Game_complexity.

[15] Wikipedia. Go Game. https://en.wikipedia.org/wiki/Go_(game).

[16] Wikipedia. Go Rules. https://en.wikipedia.org/wiki/Rules_of_Go.

[17] Wikipedia. Make and Makefiles. https://en.wikipedia.org/wiki/Make_(software).

[18] Wikipedia. Shannon Number. https://en.wikipedia.org/wiki/Shannon_number.

[19] Wikipedia. Solved Game. https://en.wikipedia.org/wiki/Solved_game.

[20] YouTube. AlphaGo vs Lee Sedol Documentary. https://www.youtube.com/watch?v=WXuK6gekU1Y.

[21] YouTube. The Thinking Game (αν ξεκινήσατε την άσκηση μια μέρα - ή ώρα - πριν την υποβολή συνιστούμε να μην πατήσετε το λινκ). https://www.youtube.com/watch?v=d95J8yzvjbQ.