Εργασία1 Θεωρία παιγνίων NE509_LAB1_2025_1093306
Κωνσταντίνος Αλεξόπουλος

Ασκηση1

[Z1.0]

Ορισμός και Μετασχηματισμός παιγνίου.

Ακολουθώντας την μαρτυρία της εικόνας2, αντιλαμβανόμαστε οτι το παιχνίδι "Πυργοι και Σκακιέρα" έχει κλειστή λύση καθώς διακρίνουμε οτι ανάγεται απεικονιστικά $<_m$ στο παιχνίδι Poker Nim.
Η παραδοχή αυτή γίνεται αποδεκτή αν θέσουμε τους εξής ισομορφισμούς:
- Οι αποστάσεις μεταξύ των πύργων κάθε στήλης ισοδυναμούν με τον αριθμό των σπίρτων σε μια στίβα του poker nim.
- Μείωση της απόστασης ισοδυναμεί με αφαίρεση σπίρτων ενώ αύξηση με πρόσθεση.
- Η απόσταση που έχει κάθε πιόνι απο το border της σκακιέρας ισοδυναμεί με τον αριθμό των σπίρτων που έχει στην μπάνκα του. (το επιτρεπτό border εννοείται).

Αξιολόγηση Κατάστασης.

Η απόφαση για το αν μια παρτίδα είναι ΠΝ/ΠΗ αποδίδεται στον Charles Leonard Bouton, ο οποιος διατύπωσε την λύση του παιχνιδιού με την μέθοδο NIM-sum.
1) Μετέτρεψε το πλήθος των σπίρτων κάθε στήλης στο δυαδικό
2) Κάνε XOR όλες τις στίβες μεταξύ τους και βρές το συνολικό value
3) Το αποτέλεσμα ταυτίζεται με το G (G==*n).
   Αν *n = *0 τότε ΠΗ αλλιώς ΠΝ

Στην περίπτωση που ο παίκτης προσθέσει σπίρτο (aka απομακρύνει το πιόνι του) ο επόμενος αρκεί να ανεραίσει αυτή την ενέργεια (να αφεραίσει τον ίδιο αριθμό σπίρτων). Αυτό αναγκάζει τον δεύτερο παίκτη να παιξει παρτίδα ήττας, αφού κάποια στιγμή θα ξεμείνει απο σπίρτα (aka εχει φτάσει το border). Αν ακολουθήσουν και οι δύο παίκτες αυτή τη στρατηγική, τότε το Poker Nim ανάγεται σε απλό Nim χωρίς επιπλέον προυποθέσεις και περιορισμούς.

Προσδιορισμός Νικηφόρας Στρατηγικής.

Η νίκη του πρώτου παίκτη διασφαλίζεται αν υπάρχει περιττός αριθμός παρτίδων μέχρι G *0 και του δεύτερου αν υπάρχει άρτιος αριθμός παρτίδων μέχρι G *0. Δεδομένου οτι ο κάθε παίκτης θα προσπαθήσει φέρει το παιχνίδι σε περιττό ή άρτιο πλήθος εναπομείναντων καταστάσεων, το παιχνίδι μεσω προς τα πίσω επαγωγής έχει προδιαγεγραμμένο τέλος απο την αρχή. Στόχος του κάθε παίκτη είναι να πασάρει στον αντίπαλο μια ΠΗ και αυτό θα το κάνει αφαιρώντας τόσα σπίρτα όσα χρειάζεται για να ερθει το παιχνίδι σε ισορροπία. Οποιος το καταφέρει πρώτος, θα μπορεί να το καταφέρνει συννέχεια για το υπόλοιπο του παιχνιδιού. 'Οποιος λαμβάνει το άτακτο παιχνίδι, πάντα θα μπορεί με μια κίνηση να το ισορροπεί.

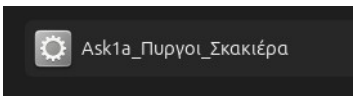
Αλγόριθμος Προσδιορισμού Βέλτιστης Επόμενης Κίνησης:

1) Εξέφρασε την απόσταση μεταξύ των πύργων μιας στήλης ως το *n της στήλης (gap)
2) Κάνε το συνολικό XOR χωρίς κρατούμενο (nim_sum), αν είναι *0 τότε δεν υπάρχει τρόπος να νικήσεις κοντρα σε αντίπαλο που παίζει βέλτιστα οπότε παίξε τυχαία
3) Διαφορετικά, για κάθε στήλη που ικανοποιεί
   (gap XOR nim_sum) < gap
   υπολόγισε το remove = gap – (gap XOR nim_sum), δηλαδή πόσα σπίρτα χρειάζεται να "αφαιρέσουμε".
Σημείωση: Αν πολλές στήλες εμφανίζουν τον ίδιο αριθμό τότε όλες είναι αποδεκτές ΠΝ

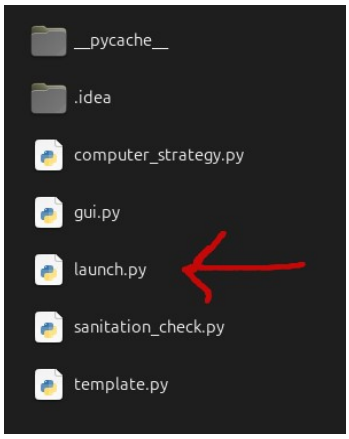
Υλοποίηση Προγράμματος Ανταγωνισμού Με τον Υπολογιστή.

Το πρόγραμμα υλοποιεί όλα τα ζητήματα [Z1.1], [Z1.2], [Z1.3] των εκφωνήσεων. Παρακάτω υπάρχει ενα συνοπτικό user manual για το πως να παίξετε.

User Manual:
Το πρόγραμμα διαθέτει γραφική διεπαφή (gui) και τρέχει ως standalone πρόγραμμα με double click. (ισως υπάρχει χρόνος αναμονής)



Σε περίπτωση που δεν δουλέψει, στα παραδοτέα υπάρχουν τα source αρχεία python.



Τα σχόλια καλύπτουν πλήρως την λειτουργία του κώδικα ιδιέτερα στο computer_strategy.py που υλοποιεί την λογική που κάνει κινήσεις ο υπολογιστής.

1) Ανοίξτε κάποιο ide για python της επιλογής σας, κάντε open το directory NE509_LAB1_2025_1093306_Ask1_1_Code

2) Τρέξτε το αρχείο launch.py

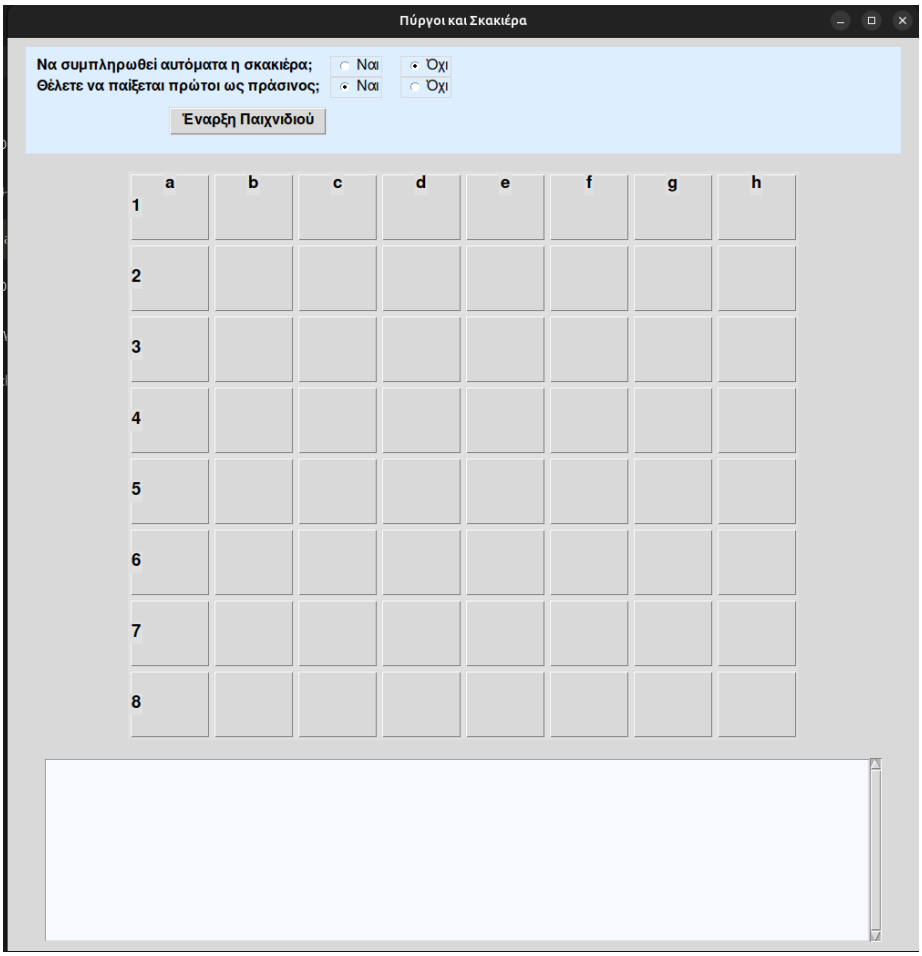
template.py → Υλοποιεί τους κανόνες του παιχνιδιού και διαχειρίζεται την λογική για το setup της σκακιέρας. Μπορεί να λειτουργήσει ανεξάρτητα και μας επιτρέπει να παίζουμε χωρίς γραφική διεπαφή απο το terminal. Χρησιμοποιήθηκε στα πρώτα σταδια της ανάπτυξης του κώδικα για debugging.

sanitation_check.py → Ευθύνεται για τον έλεγχο ορθής εισόδου, το υλοποιούμε ξεχωριστά για ευκολότερο debugging και seperation of concerns. Τα μυνήματα λάθους αφορούν κυρίως το terminal version του παιχνίδιού.

computer_strategy.py → Το “μυαλό” του υπολογιστή, λαμβάνει ως είσοδο την κατάσταση της σκακιέρας και δίνει ως έξοδο την κίνηση του υπολογιστή (μια συντεταγμένη στη σκακιέρα). Για παρτίδες ήττας υλοποιεί τον τυχαίο αλγόριθμο και για παρτίδες νίκης τον βέλτιστο poker nim αλγόριθμο.

gui.py → Δεν ζητούταν ρητά, όμως διευκόλυνε πολύ την διαδικασία του debugging της λογικής του υπολογιστή.

Η εφαρμογή πρέπει να μοιάζει κάπως έτσι



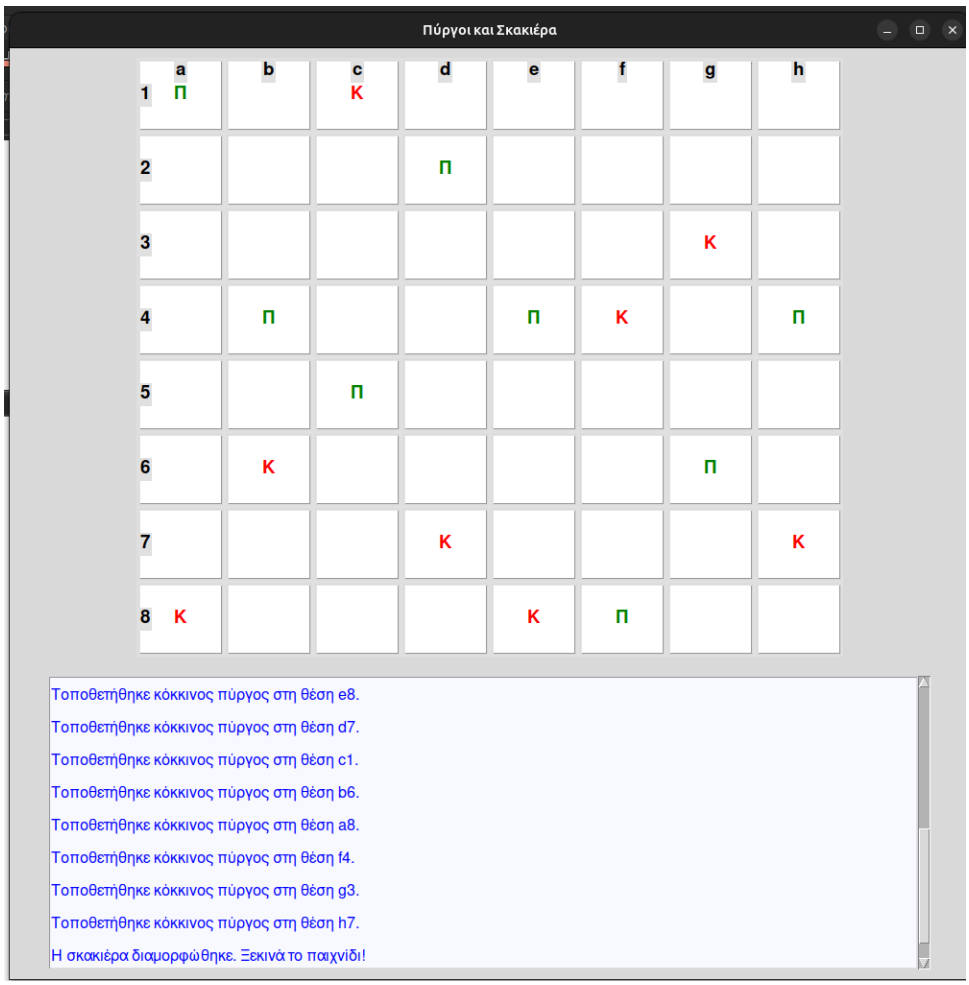
Επιλέγεται πάνω πάνω αν θέλετε να συμπληρωθεί αυτόματα η σκακιέρα, αν θέλετε να παίξετε πρωτος ως πράσινος και πατάτε έναρξη παιχνιδιού.

Κάτω υπάρχει text box με μυνήματα του συστήματος. Για να τοποθετήσετε ενα πύργο ή να τον μετακινήσετε απλά πατάτε στο άδειο κουτί που σας ενδιαφέρει και το σύστημα θα ανταποκριθεί κατάλληλα.

Οταν το παιχνίδι τελειώσει, θα εμφανιστεί κάτω κουμπί για να παίξετε ξανά.

Απλά πατήστε πάνω στο κελί που θέλετε να πάει ο πύργος δεν χρειάζεται να πατήσετε το αρχικό κελί.

Δεν εχει σημασια αν θα τοποθετησετε πανω τους πρασινους πυργους και κατω τους κοκκινους.



Για να αποφύγετε λάθος απαντήσεις απο το σύστημα, ΠΕΡΙΜΕΝΕΤΕ πρώτα ο υπολογιστής να κάνει την κίνησή του!

Αν δεν προλάβατε να δείτε την κίνηση του υπολογιστή, υπάρχει μύνημα στο TextBox που θα σας ενημερώσει κατάλληλα.

***Αν στο τέλος της παρτίδας δεν εμφανιστεί το κουμπί "Νεο Παιχνίδι" τότε τρέξτε το πρόγραμμα απο την αρχή (πιθανώς είναι bug)**

Ασκηση 2

Το παιχνίδι του δούρειου ίππου παρουσιάζει τις εξής ομοιότητες με το παραπάνω πρόβλημα:
- Δύο παίκτες παίζουν εναλλάξ.

-Δεν υπάρχει τύχη.

- Υπάρχει τέλεια πληροφόρηση. (ολοι οι παίκτες γνωρίζουν ολες τις διαθέσημες πληροφορίες)

- Το παιχνίδι τελειώνει κάποια στιγμή και πάντα.

Παιχνίδια που επιδικνύουν τα παραπάνω χαρακτηριστικά λέγονται Αμερόληπτα Συνδιαστικά και έχουν μια πολύ χρήσιμη ιδιότητα: Κάθε θέση τους μπορεί να αναπαρασταθεί με μια ισοδύναμη στοίβα Nim, δηλαδή με έναν αριθμό Grundy. (θεώρημα Sprague-Grundy).

Αυτό είναι εξεραιτικά χρήσημο καθώς έτσι μπορούμε να γνωρίζουμε αν ο παίκτης βρίσκεται σε θέση νίκης ή ήττας. Αν βρίσκεται σε νίκη κάνουμε τις κατάλληλες κινήσεις και πάλι για να δώσουμε στον αντίπαλο παρτίδα ήττας.

Πως όμως μπορούμε να προσδιορίσουμε αν είναι παρτίδα νίκης και μάλιστα ποιά συγκεκριμμένη παρτίδα νίκης είναι η κάθε θέση στη σκακιέρα?

Για να το κατανοήσουμε αυτό πρέπει να επικαλεστούμε το πόρισμα των sprague-grundy που λέει οτι “κάθε παρτίδα nim μπορεί να αναπαρασταθεί απο ένα σύνολο εναλλακτικών παρτίδων”

Ομως δεν έχουμε την γνώση της παρτίδας nim, αυτή ψάχνουμε! Αμέσως αμέσως οδηγούμαστε στην αναζήτηση λύσης με την ανάποδη λογική. Δηλαδή απο ένα σύνολο εναλλακτικών παρτίδων (γειτονικά τετραγωνάκια της σκακιέρας), να βρούμε τον αριθμό grundy της βασικής παρτίδας (το τετραγωνάκι που είμαστε τώρα). Λύση σε αυτό δίνει το MEX (maximum excluded principle), οπου για ενα σύνολο εναλλακτικών παρτίδων, ο αριθμός της βασικής παρτίδας είναι ο ελάχιστος που λείπει.

Αρκεί λοιπόν να προσδιορίσουμε την πρώτη εναλλακτική παρτίδα (ξεκινώντας απο κάποια γνωστή παρτίδα ήττας) και να εφαρμόσουμε αναδρομικά τον κανόνα mex σε κάθε κελί της σκακιέρας.

Συγκεκριμμένα για σκακιέρα δύο διαστάσεων και πιόνι που εκτελεί τις συγκεκριμμένες κινήσεις ισχύει ότι:

Trojan(r,c) = mex {Trojan(r-2, c-1), Trojan(r-1, c-2)}

(αναπαριστάται η πρώτη κίνηση ως μετακίνηση δυο θέσεις αριστερά και μια πάνω (πρασσινη), ενω η δευτερη ως δυο θέσεις πάνω και μια αριστερά (κοκκινη)). Τα κελιά για τα οποια δεν υπάρχει επιτρεπτή κίνηση παίρνουν απευθείας *0

Ο βέλτιστος αλγόριθμος λοιπόν είναι ο εξής.

start Βρές τον grundy του κάθε κελιού

α) hardcoded για το 8χ8 (προτιμότερο για απλότητα υλοποίησης)

β) Αλγόριθμος αναδρομικής ανακάλυψης:

- Ξεκίνα απο το κελί στην γωνία πάνω αριστερά και ανέθεσε *0

- Προχώρα στο αμέσως επόμενο δεξιά κελί, αν αυτό δεν υπάρχει επέστρεψε στην πρώτη στήλη και μετακινήσου μια γραμμή κάτω

- Ελεγξε τις εναλλακτικές παρτίδες του κελιού που εξετάζεις

- Αν δεν υπάρχουν εναλακτικές παρτίδες γράψε *0 αλλιώς γράψε τον ελάχιστο αριθμό που λείπει.

- Οταν αναθέσεις επιτυχώς αριθμούς grundy σε κάθε κελί, ξεκίνα το παιχνίδι.

- Απο τις εναλλακτικές σου παρτίδες πάντα θα επιλέγεις αυτή με *0, αν και οι δύο είναι *0 ή αν και οι δύο είναι ΠΝ επέλεξε τυχαία.

Με βάση αυτά φτιάχναμε τον κώδικα. Τον τρέχουμε με τρόπο παρόμοιο με τη προηγούμενη άσκηση, απλά τώρα πατάμε το “Δούρειος_Ιππος_App”. Παλι μεσα στον αντίστοιχο folder υπάρχει η ανάλυση του κώδικα.

	a	b	c	d	e	f	g	h
0	*0	*0	*0	*0	*0	*0	*0	*0
1	*0	*0	*1	*1	*1	*1	*1	*1
2	*0	*1	*1	*1	*2	*2	*2	*2
3	*0	*1	*1	*0	*0	*0	*0	*0
4	*0	*1	*2	*0	*0	*1	*1	*1
5	*0	*1	*2	*0	*1	*1	*1	*2
6	*0	*1	*2	*0	*1	*1	*0	*0
7	*0	*1	*2	*0	*1	*2	*0	*0