COSC343 Assignment 1

X Nimmt! game playing agent

Weight:20% Lecturer: Lech Szymanski

For this assignment, you will create an agent that plays the game of X Nimmt!.

Task 1 (10 marks)

Implement in Python a minimax-like search-based agent that plays a the game of Nimmt!.

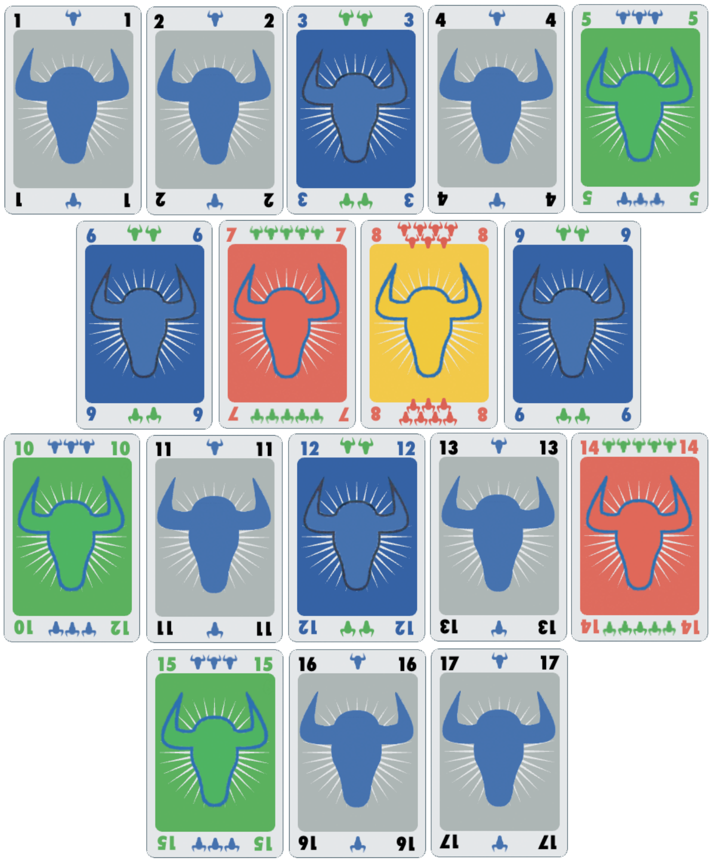


Figure 1: $K = 17$ card deck with the card number in the corner and the point value conveyed by the number of small bull heads at the top and bottom.

X Nimmt! is a game, where the players, following some rules, place cards in rows trying to avoid placing the Xth card which results in collectibg the row and the points from all the cards in that row. The objective is to take the least number of points.

The environment

The environment for this game is a deck of K cards, a table of R rows, and a starting hand of N cards to play with.

The cards in the deck are numbered consecutively from 1 to K. This numbering controls the rules of the placement of the cards on the table. The cards also carry point values, which are used for scoring when player is forced to take a row. Each cards is assigned the value of 1, except:

- every third number card is given a point value of 2,
- every fifth number card is given a point value of 3,
- every seventh number card is given a point value of 5,
- one middle number card is given a point value of 7.

The higher value rule trumps the lower value rule (so that, for instance, a card number divisible by 3 and 5 is assigned 3 points). A coloured visualisation of a $K = 17$ deck is shown in Figure 1.

Second part of the environment is the table, which contains R rows of face up/uncovered cards with the number and point values visible to all the players. At the start of the game one card is placed in each row from a shuffled deck.

Gameplay

Each player starts with N cards dealt randomly from the deck, which are kept secret from other players. The gameplay consists of N rounds, each round composed of two phases.

Phase 1 All players select one card from their hand to play; each player makes the selection in secret (from the others) and all the selected to play cards are revealed at the same time;

Phase 2 In the ascending oder of the number of the selected/revealed cards, each is added to the table following these rules:

- a card can be added to the end of the row if its number is larger than the number of the last card in the row;
- if there multiple rows that can legally house a card, the new card gets added to the row with the smallest difference between its number and the number of the last card in the row;

- if the added card is the X^{th} card in the row, all the $X^{\text{th}}-1$ cards are discarded and the new card starts a new clean row; the player who played that card collects all the points from the discarded cards;
- if there are no rows that can legally house a card, the row with the least number of total points is selected, the cards are discarded and the new card starts the row anew; the player who played the new card collects the points from discarded cards.

The game continues for at total of N rounds, at which point all the players are out of cards; the scores are compared and the player with least points wins.

Game parameters

For the purpose of development and testing, the game environment in this assignment has the following configurable parameters: P number players, K-card deck, R rows on the table, N cards in the starting hand, and X^{th} card taking a row. All these can be configured to different integer values, though it's necessary that $K \geq P \cdot N + R$.

The default settings for this assignment are $P = 2$ players, $K = 17$-card deck, $R = 3$ rows, $N = 5$-card starting hand and $X = 4^{\text{th}}$ card taking a row.

Your agent will be scored over a series of games against another player with the average points collected per game.

The agent

The agent, whose behaviour you have to implement for this assignment, is an X Nimmt! player. On initialisation the agent is provided with a game configuration including the card deck, R number of rows on the table, N cards in the starting hand of each player and the number X that takes a row. The number of players, in this assignment, will always be $P = 2$ (i.e. your agent plays a game against one opponent).

The agent function

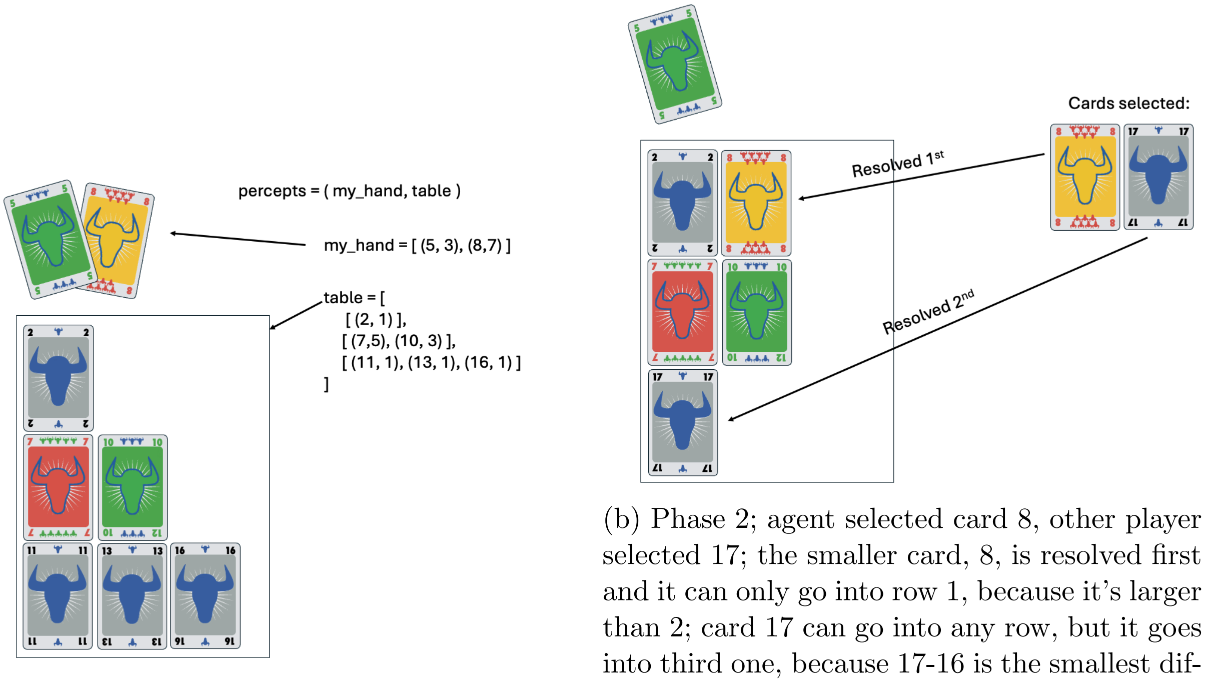
The agent select a card to play from its hand in its gent function. The argument to the function is a tuple of percepts, which provide information about agent's hand and the state of the table. The agent function must returns an integer, the card number from its hand to play next. The last round of the game is played automatically (as no choice needs to be made, since each player is left with one card in hand).

Percepts

The percepts of the X Nimmt!-playing agent is a tuple with several pieces of information:

- `my_hand` – list of cards (represented as tuples (card number, point value)) indicating the cards from which to make the selection – for example [(4,1), (12,2)] indicates two cards numbered 4 and 12 with corresponding values of 1 and 2;
- `table` – a list of R lists of cards relating the state of the table – for example [[(1,1)], [(8, 7), (13, 1), (15, 3)], [(6, 2), (11, 1)]] is a table with three lists/rows: one card in the first row, 3 cards in the second row and 2 cards in the third row;

For an illustrative example of how the percepts work, see Figure 2.



(a) Phase 1 of $K = 17$, $R = 3$, $X = 4$ game; the agent has 2-cards left in hand, there are three rows on the table of 1, 2 and 3 cards respectively..

(b) Phase 2; agent selected card 8, other player selected 17; the smaller card, 8, is resolved first and it can only go into row 1, because it's larger than 2; card 17 can go into any row, but it goes into third one, because 17-16 is the smallest difference, and since this is a game of take $X = 4$, this being 4th card, that player takes the row (and 3 points); card 17 is now the only one in row three.

Figure 2: Two phases of XNimmt! gameplay with percepts

Actions

The action of the agent is the choice of card (specified by its number) from `my_hand`. The last card selection is made automatically (agents have no choice, since they're left with one card). To detect start of a new game the agent can check the length of the `my_hand` – length of N in N card game means this is the first round.

The engine

A framework with Python code implementing the environment for this assignment is provided for you. For files and instructions on how to use it see the "How to use the XNimmt! Engine" in the "Assignments" section on Blackboard.

Task 2 (10 marks)

You must also write a report explaining the strategy of your agent and providing some results of its evaluation. The report should include:

- a brief introduction – you don't have to repeat the entire description of the game and environment given, but a short intro to what your report is about is required;
- explanation of how your agent works – the strategy, high level algorithm, not explanation of you code line by line;
- analysis of your agent's performance with some results – ideally conveyed via graphs and/or tables;
- brief conclusion – summary/discussion of the results/conclusions;
- citations – if needed;
- information on how to run your code – if anything special is required (if you use any extra libraries over what's in the environment comment that out – for this assignment cosc343 environment should be enough)

Screenshots of the text in your terminal and/or photos of hand-drawn diagrams are not the best way to add figures to your report. Figures and diagrams are great to have, but draw them properly (in Inkscape/PowerPoint or digital pad for instance).

To give you a bit of guidance for the report structure, a LaTeXtemplate is provided (you can find it on Blackboard in the "Assignments" section under "Report template in LaTeX"). You are encouraged to use LaTeX to write your report (but don't have to). Regardless, it might be a good idea to follow the general structure provided in the template.

Finally, to pre-empt inevitable questions about the length of the report, let's say: "between 1500 and 2000 words (which is roughly 3-4 pages)". But this is not an absolute number – a bit more is fine; more pages, especially if you have lots of figures, is not a problem.

Marking scheme

This is an individual assignment and the marks will be allocated as follows:

- Task 1: **10 marks**. This task will be assessed by running your code and inspecting the performance of your agent. I am looking for search-based solution and evidence of "better than random" behaviour under different game settings. A low bar target is barely beating the random agent. Your agent must handle (i.e. not crash) under various game configuration for different (valid) choices of K, X, N and R; you can assume that number of players is always $P = 2$. Also, there is a time limit. Under configuration $K = 17$, $X = 4$, $N = 5$, $R = 3$ playing against random your agent MUST play 100 games under 5min.
- Task 2: **10 marks**. Marks will be awarded for clarity, quality of explanation of the approach taken, and at some evaluation of your algorithm with some analysis.

The percentage of the obtained marks out of 20 will be converted to a fraction of 20% that constitutes the weighting of this assignment in COSC343.

Submission

The assignment is due at **23:59 on Tuesday, 12 Aug 2025**. You should submit two files via Blackboard: my_agent.py and the pdf file containing the report. Don't zip these files into one attachment – add them separately to the submission.

Late submissions will incur a 5% penalty per day.

Academic Integrity and Academic Misconduct

Academic integrity means being honest in your studying and assessments. It is the basis for ethical decision-making and behaviour in an academic context. Academic integrity is informed by the values of honesty, trust, responsibility, fairness, respect and courage. Students are expected to be aware of, and act in accordance with, the University's Academic Integrity Policy.

Academic Misconduct, such as plagiarism or cheating, is a breach of Academic Integrity and is taken very seriously by the University. Types of misconduct include plagiarism, copying, unauthorised collaboration, taking unauthorised material into a test or exam, impersonation, and assisting someone else's misconduct. A more extensive list of the types of academic misconduct and associated processes and penalties is available in the University's Student Academic Misconduct Procedures.

Use of generative software such as ChatGPT is allowed as long as it is for the purpose of aiding, not supplanting the effort of development code and/or improving the writing. If generative software is used, students must specify (in the report's appendix) how it was used and on what aspects of the assignment.

It is your responsibility to be aware of and use acceptable academic practices when completing your assessments. To access the information in the Academic Integrity Policy and learn more, please visit the University's Academic Integrity website or ask at the Student Learning Centre or Library. If you have any questions, ask your lecturer.

- Academic Integrity Policy
- Student Academic Misconduct Procedures