ONDERZOEK

Vier op een rij

Datum: 8 oktober 2021

Opleiding: HBO-ICT, Zuyd Hogeschool

Auteurs: Manon Schüle(1805193)
Clara Schoelitsz(1427709)
Anthony Schuman(1939319)

Docent: Drs. Snijders

Versie: 4

Inhoud

Versie beheer.

Versie	Aanpassing	Toelichting
1	Opzet, gevonden bronnen, informatie van bronnen.	Het document is opgezet en de zover gevonden bronnen zijn toegevoegd. Per bron staat een korte toelichting en zijn de gebruikte zoektermen beschreven.
2	Bronnen score, interview informatie	Het interview en de resultaten zijn toegevoegd. De bronnen hebben hun eerste score gekregen.
3	Referentielijst verbeterd, onderbouwingen bijgewerkt	De referentielijst is APA beschreven en de onderbouwing zijn bijgewerkt om de keuzes van de bronnen te verhelderen.
3.1	Verbeteringen doorgevoerd	Spelfouten, achtergrond verhaal verbreedt,
4	Laatste aanpassingen	Lijst verbreedt, zoekmethodes toegelicht, interview toegelicht.

1. Achtergrond.

Het firma wilt een spel ontwikkelen om vier op een rij tegen een andere speler te kunnen spelen, maar ook tegen een computer. De computer moet ongeveer vijf stappen vooruit kunnen denken.

Voor de firma is het belangrijk dat ze een product ontwikkelen dat voldoet aan de verwachtingen die het publiek heeft van het spel. Deze verwachtingen zijn gebaseerd op bestaande versies van het spel, zowel de online versies als de fysieke versies, zoals de klassieke uitvoering en de Hasbro uitvoering.
Een spel met regels laat niets open voor interpretaties. Daarom is het belangrijk dat de feiten over het spel op een rij worden gezet.
Een werkend algoritme komt wel met enige meerduidigheid. Wat is “slim” spelen? Wat is er nodig om te winnen? Hoe groot is de winkans? Hoe diep moet een algoritme gaan om uitdagend te zijn?
De motivatie voor dit onderzoek is om te begrijpen hoe het spel vier op een rij in elkaar zit en welke eisen de spelregels stellen aan de computer in het programma. Aan het eind van het onderzoek moet er een overzicht zijn van de spelregels, een verzameling aan ideeën over algoritmes voor spelletjes, en moet er een grip zijn over wat er eventueel in de database zou moeten zitten.

De onderzoeksvraag is welke oplossingen bestaan er al die kunnen bijdrage aan het ontwikkelen van het programma.
Een tweede onderzoeksvraag is hoe het spel vier op een rij precies in elkaar zit.
Het onderzoek is noodzakelijk om de verschillende soorten bestaande oplossingen te begrijpen, en zo te bepalen welke oplossing het beste als inspiratie dient voor het project.

2. Methodes en proces

Interview.

Om de wensen van de opdrachtgever te begrijpen, heeft er een interview plaatsgevonden. De vragen worden gebaseerd op welke informatie nog mist om de requirements compleet te maken. Er is dus bij het opstellen van de requirements gekeken welke informatie over de opdracht miste om de requirements compleet te kunnen maken. Aan de hand van de al gemaakte requirements is er een lijst met vragen gebrainstormd. Vervolgens zijn de vragen gefilterd op relevantie en tautologie.
Het interview dient om de verwachtingen van de opdrachtgever op een rij te krijgen. Deze verwachtingen dienen als basis voor de requirements, die worden uitgebreid door het bronnenonderzoek. Tijdens het interviewen is gebruikt gemaakt van LSD om dieper in te gaan op de antwoorden en de motivatie achter de antwoorden te begrijpen.

Bronnen onderzoeken.

Zoekmethodes.

Er worden bronnen (voornamelijk artikelen) verzamelt vanaf het internet.
Er word een lijst van zoektermen samengesteld die we verwachten de juiste resultaten te kunnen geven. Deze zoektermen worden in verschillende zoekmachines getypt. Deze zoekmachines zijn Google, Bing en de bibliotheek van Zuyd. Vanaf de zoektermen worden zo veel mogelijk artikelen en websites in een lijst verzameld. Artikelen waarvan het duidelijk is dat deze over iets anders gaan dan wat we zochten, of geen informatie bron is maar iets anders gaan direct naar de exclusielijst. Dit wel onder consensus van de groep.
Vervolgens worden deze bronnen geanalyseerd op de inhoud.

Selectieprocedure.

De inhoud en meerwaarde van een artikel of bron wordt beoordeeld aan de hand van een punten- en score methode. Deze methode is geïnspireerd door het idee van Kitchenham om een kwantitatieve score te hangen aan de bronnen. Zo wordt bepaald of ze op de inclusie lijst blijven. De tekst in de artikelen worden geskimmed om te bepalen of ze meerwaarde hebben voor de opdracht. De punten worden toegekend aan de hand van onderstaande criteria.

Punten bepaling

Criteria	Aantal punten
Zoekwoord in de titel	40
Relevante onderwerpen in de samenvatting	10
Relevante onderwerpen in de tekst	30
Onderwerp geeft bijdrage aan de oplossing over het algoritme.	20

Aan de hand van een aantal zoektermen zijn artikelen vanaf het internet opgezocht en verzameld als referentielijst. Deze artikelen zijn geskimmed en beoordeeld aan de hand van een score en een rank. Deze score wordt bepaald door de titel, door wat er eventueel in de samenvatting staat en wat er naar voren komt tijdens het skimmen van de tekst. De titel en de samenvatting geven een indicatie

van wat er mogelijk in het artikel staat. Als wat gezocht wordt niet in de eventuele samenvatting staat, dan is de kans veel groter dat het niet terugkomt in de tekst. Hierom heeft welke termen er in de samenvatting voorkomen meerwaarde en moet dus gescoord worden.
Een artikel wordt verworpen als het niet aan meer dan 1 van de bovengenoemde criteria voldoet. De totale score van een bron bepaalt welke rank een bron krijgt (laag, middenlaag, midden, middenhoog, hoog). Om als relevant beschouwd te worden, moet een de bron minstens 30 punten score. De hoogte van de score bepaald de hoogte van de rank. Is de score te laag, dan gaat de bron naar de exclusie lijst.

Overeenkoming.

Als een artikel door het selectie proces heen is gekomen, wordt deze doorgelezen. Relevante informatie wordt eruit genoemd en voorgelegd aan de groep. Er moet een consensus gevormd worden over hoe relevant de informatie is. De informatie die als relevant wordt gezien, wordt herhaald in het document. Als er geen consensus kan worden gevormd, wordt het samenwerkingscontract opgeroepen, waarin staat dat de aangewezen persoon de knoop doorhakt.

3. Artikelen

https://123bordspellen.com/hoe-speel-je-4-op-een-rij/
https://www.jijbent.nl/spelregels/4rij.php
https://www.spelregels.eu/vier-op-een-rij/
https://www.speluitleg.com/4-op-een-rij/

punten: 70
Deze vier artikelen halen elk dezelfde score.
Deze vier artikelen leggen het origine van het spel uit en hoe het bord in elkaar zit. De artikelen geven ook een overzicht van de spelregels. Het gebruik van meerdere bronnen valideert de informatie.
De spelregels vormen de basis van de scope en de requirements van het spel. Zonder deze basis zal het programma nooit voldoen aan de verwachtingen.

Score: Hoog.

https://www.gamesver.com/the-rules-of-connect-4-according-to-m-bradley-hasbro/

punten: 70
Dit vijfde artikel over de spelregels verteld over een andere versie van het spel dan de eerste vier artikelen. Deze versie heeft een andere bordindeling. Deze versie is bedacht door Hasbro. Deze informatie legt het verschil uit tussen de twee versies, en voorkomt fouten bij het opstellen van het bord. De verheldering geeft dit artikel een grote meerwaarde, omdat het zekerheid geeft dat de indeling van het bord juist is.

Score: Hoog.

---

https://www.askpython.com/python/examples/connect-four-game

Dit artikel laat zien hoe het vier op een rij spel kan worden geprogrammeerd in Python. De functies die het artikel omschrijft, kunnen als inspiratie dienen voor het project. Het nadeel is dat de functies moeten worden vertaald naar C#.
punten:60

Score: middenhoog.

---

https://medium.com/analytics-vidhya/artificial-intelligence-at-play-connect-four-minimax-algorithm-explained-3b5fc32e4a4f
In dit artikel wordt een toelichting gegeven op een decision tree in combinatie met het 4 op een rij spel, en een minimax algoritme. De schrijver omschrijft hoe het spel gewonnen wordt en het algoritme wordt toegepast. Dit kan als inspiratie dienen voor het algoritme in het project.
punten: 60
Score: middenhoog.

---

https://project.dke.maastrichtuniversity.nl/games/files/phd/Baier_thesis.pdf

Deze thesis legt de werking van het algoritme uiteen en refereert weer naar het minimax algoritme. De thesis legt een link naar de monte carlo tree search. De thesis ontleet het principe op een wetenschappelijk niveau en kan verdere inzichten bieden op de werking van de algoritmes. De thesis heeft minder meerwaarde van het andere artikel omdat het op een niveau zit dat mogelijk niet van toepassing is op een casus van drie weken. Wel kan deze bron als controle bieden voor de grijze literatuur in het onderzoek.
punten: 60

Score: middenhoog.

---

https://en.wikipedia.org/wiki/Monte_Carlo_tree_search

Op wikipedia wordt de origine en werking van de monte carlo tree search toegelicht. De pagina legt de monte carlo search tree uit en verteld waar en hoe deze toegepast wordt.
punten: 100.
Score: hoog.

https://codebox.net/pages/connect4

Dit artikel verteld over de toepassing van de monte carlo tree search bij het vier op een rij spel. Het bespreekt de voordelen, de nadelen en hoe het werkt. Ook deze vallen onder grijze literatuur en zouden aan de hand van de thesis gevalideerd kunnen worden. Er staat connect4 in de titel, de inhoud is anders dan wat er verwacht werd. De informatie draagt bij aan het algoritme.
punten: 60.

Score: middenhoog.

4. Resultaten.

Spelregels en spel origine.

Iedere speler heeft 21 stenen. Deze stenen heb meestal de kleuren rood of geel. Het doel is om als eerste 4 stenen op een rij te krijgen. Dit kan diagonaal, verticaal of horizontaal. Het spel eindigt als alle vakjes vol zijn en er geen vier op een rij zijn gemaakt. Het bord wordt dan weer leeggehaald. Degene met de gele (lichtere) stenen begint altijd als eerst. Een boord heeft 6 rijen en 7 kolommen, dus in totaal 42 vakjes. Je mag niet twee keer achter elkaar een steen leggen.

Mogelijkheden voor de code.

"np.zeros( ) function is used to create a matrix full of zeroes. This function in Python can be used when you initialize the weights during the first iteration in TensorFlow and other statistic tasks. ((6,7)) are the dimensions. 6 rows and 7 columns. Then we just return that board.

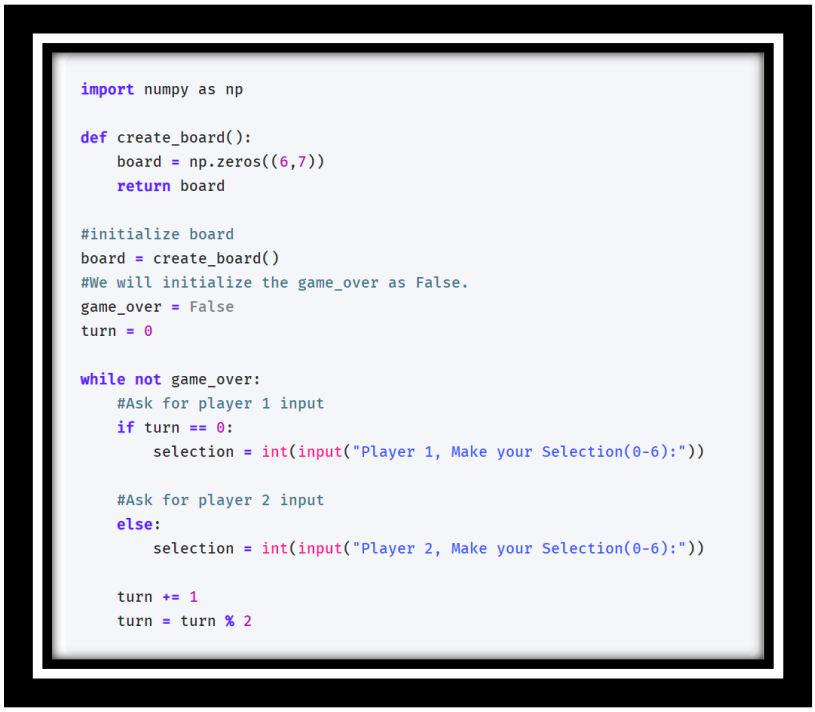
We will begin writing the main game loop now. We're going to create a loop as while not game_over. A while not loop repeatedly executes the body of the loop until the condition for loop termination is met. Our loop is going to run as long as this game_over variable is false. We will initialize the game_over as False. The only time its going to switch to true is if someone gets 4 circles in a row."

"We will begin writing the main game loop now. We're going to create a loop as while not game_over. A while not loop repeatedly executes the body of the loop until the condition for loop termination is met. Our loop is going to run as long as this game_over variable is false. We will initialize the game_over as False. The only time its going to switch to true is if someone gets 4 circles in a row.

To increase the turn by 1 we will use turn += 1. To make it switch between player 1 and 2 alternatively, we use turn = turn % 2."

Het artikel van askpython laat een methode van programmeren zien waarbij het spel wordt doorgezet met een Boolean en een while loop. De Boolean is game_over = false, en de code geeft aan dat het spel door moet gaan zolang de Boolean op false staat. Binnen deze while loop, kan het spel plaatsvinden.

De beurten kunnen worden bepaald aan de hand van een integer. Als de integer op 0 staat, is speler een aan de beurt, als de integer op 1 staat, is de andere speler aan de beurt, en dan wordt de integer teruggezet naar nul.
De speler kan kiezen tussen de 7 kolommen, waardoor zijn steentje kan worden toegevoegd aan een array, en standard naar de eerst volgende beschikbare index.
Er moet voorkomen worden dat indexen worden overschreven; het programma zou dan een foutmelding moeten geven dat de array vol zit.
Er moet een gelijkspel melding komen als alle array's vol zijn.



```python
import numpy as np

def create_board():
    board = np.zeros((6,7))
    return board

#initialize board
board = create_board()
#We will initialize the game_over as False.
game_over = False
turn = 0

while not game_over:
    #Ask for player 1 input
    if turn == 0:
        selection = int(input("Player 1, Make your Selection(0-6):"))

    #Ask for player 2 input
    else:
        selection = int(input("Player 2, Make your Selection(0-6):"))

    turn += 1
    turn = turn % 2
```

Figuur 1, functies voor het bord en het bepalen van de beurt

```python
def create_board():
    board = np.zeros((6,7))
    return board

def drop_piece(board,row,col,piece):
    board[row][col]= piece

def is_valid_location(board,col):
    #if this condition is true we will let the use drop piece here.
    #if not true that means the col is not vacant
    return board[5][col]==0

def get_next_open_row(board,col):
    for r in range(ROW_COUNT):
        if board[r][col]==0:
            return r

def print_board(board):
    print(np.flip(board,0))
```

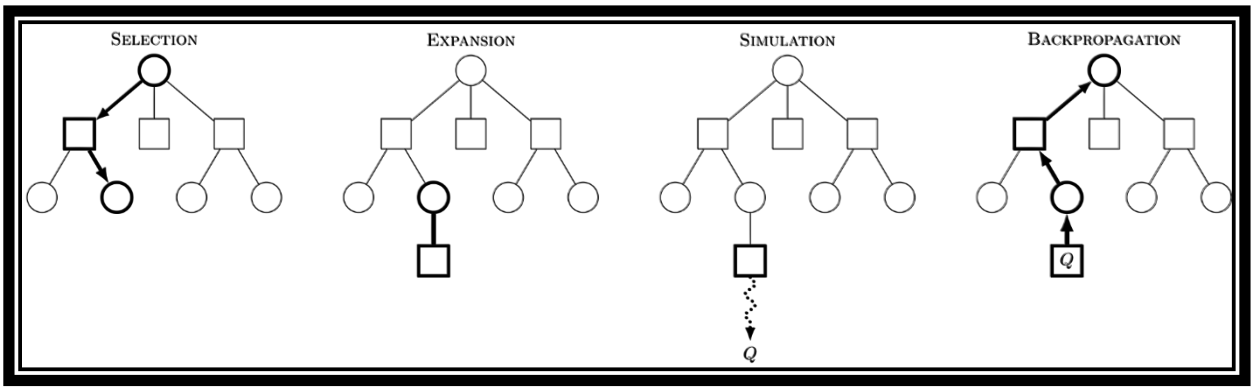
Figuur 2, mogelijke methodes in python

Aan de hand van een dictionary kan het systeem nakijken of een van de conditions voor een winnend spel van toepassing is. Elke keer als er een speler is geweest, dan kijkt het systeem de dictionary na.

Informatie over algoritmes.

Monte Carlo research tree.
"In computer science, Monte Carlo tree search (MCTS) is a heuristic search algorithm for some kinds of decision processes, most notably those employed in software that plays board games. In that context MCTS is used to solve the game tree.

MCTS was combined with neural networks in 2016 for computer Go. It has been used in other board games like chess and shogi, games with incomplete information such as bridge and poker, as well as in turn-based-strategy video games (such as Total War: Rome II's implementation in the high level campaign AI). MCTS has also been used in self-driving cars, for example in Tesla's Autopilot software."



"Monte-Carlo Tree Search
In order to assess the strength of Connect Zero I first developed a separate piece of software for it to play against. This software uses an algorithm called Monte-Carlo Tree Search (MCTS) which has the advantage that it can easily be tuned to play at different levels of ability. MCTS works by trying out each of the possible moves that could be played from the current position in lots of simulated games against itself. It starts off picking moves at random, but over time concentrates on those moves that seem to give it a better chance of winning. The longer the algorithm is allowed to run, the better moves it will find.
When referring to a player using this algorithm I use the abbreviation 'MCTS-' followed by the number of simulated games that are being used. For example a 'MCTS-1000' player simulates 1000 games before deciding on what move to play.

Self-Play Reinforcement Learning
The Neural Network was trained using 'self-play', which is exactly what it sounds like: two opponents play many games against each other, both selecting their moves based on the scores returned by the network. As such, the network is learning to play the game completely from scratch with no outside help. The games that it plays against other types of opponent, such as MCTS, are used only to check its progress and the results of those games are not used for training.
At the end of each training game the network receives some feedback about how well its recommended moves worked. The scores for the board positions recommended to the losing player are all reduced slightly, and those recommended to the winning player are increased.

The scores for board positions occurring towards the end of the game (i.e. closer to the eventual victory or loss) are adjusted by larger amounts than the scores for earlier positions, in other words they are apportioned a larger amount of the credit or blame for what happened."

Minimax.

"Minimax algorithm is a recursive algorithm which is used in decision-making and game theory especially in AI game. It provides optimal moves for the player, assuming that the opponent is also playing optimally. For example, considering two opponents: Max and Min playing. Max will try to maximize the value, while Min will choose whatever value is the minimum. The algorithm performs a depth-first search (DFS) which means it will explore the complete game tree as deep as possible, all the way down to the leaf nodes. The algorithm is shown below with an illustrative example."

```
function alphabeta(node, depth, α, β, maximizingPlayer) is
    if depth = 0 or node is a terminal node then
        return the heuristic value of node
    if maximizingPlayer then
        value := −∞
        for each child of node do
            value := max(value, alphabeta(child, depth − 1, α, β, FALSE))
            α := max(α, value)
            if α ≥ β then
                break (* β cut-off *)
        return value
    else
        value := +∞
        for each child of node do
            value := min(value, alphabeta(child, depth − 1, α, β, TRUE))
            β := min(β, value)
            if β ≤ α then
                break (* α cut-off *)
        return value
```

Het minimax algoritme berekend een score aan de hand van de zetten van zichzelf en de speler. Een goede zet van de speler verlaagd zijn score en dan zal het algoritme opzoek gaan naar de stap met de beste score om zijn score weer te verhogen. Zo zal hij bepalen welke zet hij moet zetten. De diepte zal bepalen hoe veel stappen hij voorruit gaat kijken om zijn beste zet te bepalen.

5. Interview

Interview met Dhr. Hans Snijders van 6 oktober 2021 om 11:00 uur.

Wat ziet u graag in de database terug?

- Ontwerp van de database onderbouwen. Wat er in moet komen mogen we eigenlijk zelf bepalen, denk na over wat het doel is van de data.
- Oplossing van database uitleggen. Waarom bepaalde keuzes gemaakt

Qua scores?

- Wie speelt wanneer en high scores bepalen.

Bent u het eens met ons principe van speler vs computer?

- Ja, dat kan.

Bent u het eens met ons principe dat het algoritme een dictionary bekijkt na elke zet?

- zoek het bord af, zoekend met een search method. 2 loopjes per “direction”.
- Bord af gaan zoeken in plaats van dictionary.
- Zoeken naar een patroon van 4 op een rij.
- Eerste rij controleren, 2 e rij. Dit in een loop laten doen.
- Dit ook voor verticaal doen en daarna voor diagonaal

Nog voorkeuren over de speelwijze?

- algo moet zien dat als er 3 op een rij staan, dat hij ervoor kiest om te gaan blokkeren. Computer moet de spelregels begrijpen en een verstandig besluit.
- Datastructuur wss list van list
- Dit zoeken naar een 4 op een rij is ook te gebruiken om te zoeken wat de handigste zet is voor de computer.
- Boom structuur wordt recursief aangeroepen
- Diepte 4 a 5 is dan goed (parameter)
- Denk algoritme moet er zijn voor de computer.
- Geen eis dat het algoritme wint

Hoe visueel moet het zijn?

- kruisjes, bolletjes, dat je het spel kunt spelen en zien,
- Kruisjes en bolletjes in een raster is genoeg
- Colom aangeven met een input getal zodat er dan een kruisje en of bolletje komt te staan
- CONSOLE APPLICATIE
- Clear screen en dan opnieuw het bord printen

Conclusies interview.

De belangrijkste conclusie is dat het idee om een dictionary in te zetten als wincontrole te veel stappen geeft voor het algoritme. De opdrachtgever suggereert dat er een snellere manier is om het bord te checken.

De opdrachtgever uit de wens voor een database, maar laat de beslissingen over wat er in komt, over aan het team. De database moet bijdrage aan het functioneren van het programma, en doet dit door te voorkomen dat er onnodig wordt gehardcoded. In het ontwerp kan worden nagedacht over de meerwaarde van de database door middel van een ERD.

Er zijn weinig tot geen wensen qua hoe visueel het spel moet zijn. UI en visuele weergave zijn de laagste prioriteit: het moet alleen duidelijk zijn voor de spelers wat er gebeurd en waar hun zetten zijn.

Als laatste zijn er eisen gesteld aan het algoritme. De computer moet een aantal stappen voorruit kunnen denken om het spel een uitdaging te maken voor de menselijke speler.

6. Bijlage en referentielijst.

Tijdens het onderzoek is gebruik gemaakt van verschillende key words om goede artikelen te vinden die tijdens deze casus gebruikt kunnen worden. Deze keywords zijn als volgt:

- 4 op een rij
- Connect 4
- Connect 4 algoritme
- 4 op een rij algoritme
- 4 op een rij python
- Connect 4 python
- 4 op een rij C#
- Connect 4 C#
- Spel algoritmen
- Game algoritmes
- Maastricht university connect 4
- Monte Carlo tree search

De zoektermen hebben de volgende resultaten geleverd. Deze links zijn gekozen op het feit dat de zoektermen voorkwamen in de titel, samenvatting of tekst, of dat de informatie een mogelijke bijdrage kan leveren aan het ontwikkelen van het algoritme van de computer speler.

Inclusielijst.

https://123bordspellen.com/hoe-speel-je-4-op-een-rij/ (Alles over bordpsllen, 2020)
https://www.jijbent.nl/spelregels/4rij.php (Egbert Brinks, 2021)
https://www.spelregels.eu/vier-op-een-rij/ (Kortenberg.bem onbekend)
https://www.speluitleg.com/4-op-een-rij/ (onbekende auteur)
https://www.gamesver.com/the-rules-of-connect-4-according-to-m-bradley-hasbro/ (Gamesver Team, 2021)
https://www.askpython.com/python/examples/connect-four-game (Ask Python, onbekend)
https://medium.com/analytics-vidhya/artificial-intelligence-at-play-connect-four-minimax-algorithm-explained-3b5fc32e4a4f (Jonathan C.T. Kuo, 2020)
https://project.dke.maastrichtuniversity.nl/games/files/phd/Baier_thesis.pdf (Johannes Sebastian Baier, 2015)
https://codebox.net/pages/connect4 (Rob Dawsen, 2021)

https://en.wikipedia.org/wiki/Monte_Carlo_tree_search
https://www.computermeester.be/ontspanning/vier-op-een-rij.htm (computermeester,2021)
https://www.youtube.com/watch?v=ypjrE40Y1cw (Skrenzy, 2017)
https://www.youtube.com/watch?v=MMLtza3CZFM&list=PLFCB5Dp81iNV_inzM-R9AKkZZlePCZdtV&index=7&t=4220s (Keith Galli, 2019)
https://github.com/KeithGalli/Connect4-Python/blob/master/connect4_with_ai.py (Keith Galli)
https://github.com/PascalPons/connect4 (Pascal Pons)

Exclusielijst.
https://connect-4.org/en (maker onbekend, 2021)
https://www.cbc.ca/kids/games/all/connect-4 (maker onbekend, 2021)
https://en.wikipedia.org/wiki/Connect_Four

https://www.mathsisfun.com/games/connect4.html (mathisfun.com, 2021)

https://www.amazon.com/Hasbro-A5640-Connect-4-Game/dp/B00D8STBHY
https://eds.s.ebscohost.com/eds/detail/detail?vid=2&sid=fe445a9a-0629-47cd-a0fd-c030d9a3ea60%40redis&bdata=Jmxhbmc9bmwmc2l0ZT1lZHMtbGl2ZQ%3d%3d#AN=edsnar.oai.tudelft.nl.uuid.093956f0.d04d.4b39.8d27.547b2a5e7f51&db=edsnar (Feyter, M.G. de Meijers, J.M. vers, G.E. TNO Arbeid, 2000)

https://eds.s.ebscohost.com/eds/detail/detail?vid=4&sid=fe445a9a-0629-47cd-a0fd-c030d9a3ea60%40redis&bdata=Jmxhbmc9bmwmc2l0ZT1lZHMtbGl2ZQ%3d%3d#AN=zuyd.232478&db=cat04966a (Hannah Fry, 2018)

https://eds.s.ebscohost.com/eds/detail/detail?vid=5&sid=fe445a9a-0629-47cd-a0fd-c030d9a3ea60%40redis&bdata=Jmxhbmc9bmwmc2l0ZT1lZHMtbGl2ZQ%3d%3d#AN=edshbo.sharekit.hr.oai.surfsharekit.nl.47d4ae49.727b.4ba7.8273.e28ac7202977&db=edshbo (Habers, M. 2020)

https://eds.s.ebscohost.com/eds/detail/detail?vid=7&sid=fe445a9a-0629-47cd-a0fd-c030d9a3ea60%40redis&bdata=Jmxhbmc9bmwmc2l0ZT1lZHMtbGl2ZQ%3d%3d#AN=2983949&db=nlebk (Guy van Lient, 2021)

https://eds.s.ebscohost.com/eds/detail/detail?vid=9&sid=fe445a9a-0629-47cd-a0fd-c030d9a3ea60%40redis&bdata=Jmxhbmc9bmwmc2l0ZT1lZHMtbGl2ZQ%3d%3d#AN=151308650&db=rzh