SJÄLVSTÄNDIGA ARBETEN I MATEMATIK

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET

Backtracking och n-queens: hur radordning påverkar sökningens effektivitet

av

Amanda Borg

2026 - No L1

Backtracking och n-queens: hur radordning påverkar sökningens effektivitet

Amanda Borg

Självständigt arbete i matematik 15 högskolepoäng, grundnivå

Handledare: Per Alexandersson

2026

AI-statement

I denna uppsats har AI använts för formatering i LaTeX för att få figurer att se snygga ut. AI har även använts vid utformningen av Pythonprogrammen i Figur 20 och Figur 21. Idén bakom programmen, den förklarande texten i programmen, vilken input som ska ges, vilken data som ska skrivas ut, vilka funktioner som ska inkluderas, modifieringar av programmen, den förklarande texten i koden och dylikt har AI **inte** skapat.

Sammanfattning

Problemet n-queens är välkänt inom kombinatoriken och har studerats sedan mitten av 1800-talet. Syftet med problemet är att undersöka hur n damer kan placeras ut på ett schackbräde av storlek $n \times n$ så att inga damer hotar varandra. Sedan problemets början har det utvecklats åt flera håll och har bland annat kommit att inkludera tredimensionella perspektiv. Man har hittat flertalet användbara tillämpningar av n-queens. I den här uppsatsen behandlas problemets historiska bakgrund och utveckling, matematiska struktur, gruppverkan i förhållande till n-queens samt hur backtracking används för att finna lösningar.

Uppsatsens huvudfokus är att undersöka hur effektiviteten i backtrackingalgoritmen påverkas av ordningen som raderna söks. För att undersöka detta skapas permutationer som beskriver vilken ordning raderna i ett bräde ska besökas under backtrackingalgoritmen. Specifikt testas slumpmässiga och icke-slumpmässiga ordningar att besöka raderna i ett 7×7-bräde. Vidare används ett Pythonprogram för att undersöka maximala antalet backtracksteg för $n = 4, 5, 6, ..., 9$ och det analyseras om det finns samband eller struktur hos de permutationer som kräver flest backtracksteg.

Resultaten visar att det går att utforma permutationer effektivt utifrån en redan känd lösning och att hela banan till den lösningen kan genereras på detta sätt. Försöken med Pythonprogrammen visade att det fanns både struktur och mönster bland de permutationerna som krävde flest backtracksteg. En permutation och dess komplement har båda exakt lika många backtracksteg. Permutationer som inledde med att placera en dam i ett hörn visade sig ofta vara de som krävde flest backtracksteg. Genom att koppla resultaten till gruppverkan kunde uppsatsen demonstrera hur gruppteori kan användas för att effektivt söka efter lösningar till n-queens.

Jag vill rikta ett tack till min handledare Per Alexandersson som har varit till hjälp med arbetets utveckling.

Abstract

The n-queens problem is well known within the field of combinatorics and has been studied since the mid 19^{th} century. The purpose of the problem is to place n queens on a chessboard of size $n \times n$ in such a way that no queens attack each other. Since its beginning, the problem has evolved to also include a three-dimensional point of view. Several real-world applications have been found for the n-queens problem. This essay will cover the problem's historical background and development, mathematical structure, group action in relation to n-queens, and the implication of backtracking to find solutions.

The main focus of this essay is to analyze how the effectiveness of the backtracking algorithm is affected by the order in which the rows of the board are searched. In order to examine this, the 7×7 board was searched according to different permutations which represent row orders. To further look at backtracking and its effectiveness, a Python program is used to calculate the maximum number of backtracking steps for $n = 4, 5, 6, ..., 9$. The permutations that resulted in the largest amount of steps are then analyzed and any pattern or structure is noted.

The results of the study show that it is possible to create a permutation from a pre-existing solution and in doing so in a structured manner, one receives a permutation which will generate a new solution with zero backtracking steps needed. By doing this, one will also generate the entire orbit of the original solution. Usage of the Python program showed that there existed both a pattern and structure among the permutations that required the greatest number of backtracking steps. It showed that a permutation had exactly the same amount of backtracking steps as its complement. The results also showed that many of the permutations that were among the least effective ones, began by placing a queen in a corner. By connecting the results to group action the essay demonstrates how group theory can be used to effectively search for solutions to the n-queens problem.

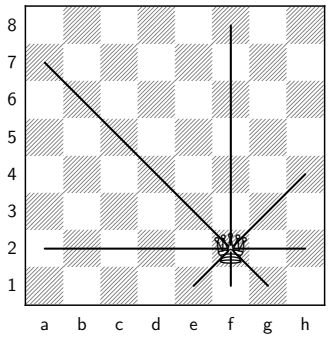
I would like to express my gratitude towards my supervisor Per Alexandersson who has provided me with valuable input during the course of the work.

Innehåll

1 Introduktion

1.1 Problemet n-queens

Problemet n-queens är ett matematiskt problem som väver samman kombinatorik, gruppverkan och backtracking för att undersöka hur n stycken damer kan placeras på ett schackbräde av storleken $n \times n$ utan att någon hotar en annan. Problemet har en gedigen historia med flera såväl matematiker som hobbytänkare som under åren har arbetat med n-queens och dess olika delproblem.

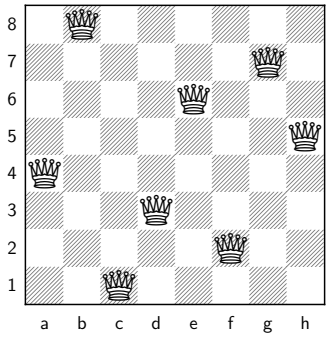


Figur 1: Alla rutor en dam i ruta `f2` hotar på ett schackbräde.

1.1.1 Historia

En av de tidigaste kända formuleringarna av problemet var i den tyska tidningen *Deutsche Schachzeitung*. Året var 1848 och en person under pseudonymen "Schachfreund" introducerade ett matematiskt schackproblem till läsarna. Problemet handlade om det vardagliga 8×8 schackbrädet och huruvida det var möjligt att ställa ut åtta damer på det utan att de kunde slå ut varandra [Bez48]. "Schachfreund" har senare visat sig vara den tyske schackspelaren Max Bezzel och hans publicering av problemet med de åtta damerna tros vara den första formuleringen av problemet som mer än 170 år senare fortfarande studeras och används. Även om Bezzel inte själv kunde bidra med några lösningar till sitt problem dröjde det inte länge innan andra schackentusiaster och matematikintresserade tog sig an problemet. I den direkt följande upplagan av schacktidningen publicerades två olika giltiga damplaceringar. Den som tros vara den första att hitta samtliga lösningar till det ursprungliga problemet är Franz Nauck. I tidningen *Leipziger Illustrierte Zeitung* kunde man år 1850 läsa om hur Nauck hade kommit fram till att det borde finnas 92 stycken olika giltiga placeringar av åtta damer på ett standardschackbräde. Han hade dock inget sätt att

bevisa att dessa lösningar faktiskt var de enda och samtliga lösningar som fanns. När allt kommer omkring kan åtta damer placeras ut på 8! olika sätt på ett schackbräde och därför är det inte konstigt att det fanns tvivel kring huruvida endast 92 av dessa är giltiga enligt problemformuleringen. År 1874 lyckades Emil Pauls bevisa att de 92 lösningarna som Nauck hade funnit var korrekta och fullständiga. Pauls antas därför vara den första att bevisa lösningsmängden för $n = 8$. [Bel09].



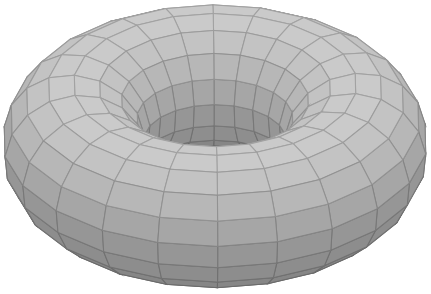
Figur 2: Ett exempel på en giltig placering av åtta damer på ett schackbräde.

Under åren har problemet n-queens utvecklats mycket och problemet har generaliserats till att gälla samtliga värden på n, alltså för både större och mindre bräden än standardschackbrädet. När n ökar växer antalet lösningar snabbt, likt en exponentiell ökning. För små värden på n kan samtliga lösningar hittas genom att testa sig fram för hand men för större värden blir det knepigare. Exempelvis har $n = 9$ 352 stycken lösningar, $n = 10$ har 724 stycken och $n = 20$ har hela 39 029 188 884 lösningar. Än så länge har antalet lösningar för $n < 28$ kunnat bestämmas, där $n = 27$ har 234 907 967 154 122 528 stycken lösningar [Slo25]. Det har varit känt nästan sedan tiden då problemet först uppkom att antalet giltiga lösningar alltid är delbart med två. Nyligen presenterade Nielsen bevis för att antalet lösningar alltid är delbart med fyra då $n \geq 6$. Beviset bygger på symmetrin hos schackbrädet och utforskar rotationssymmetri [MN26].

1.1.2 Det modulära schackbrädet

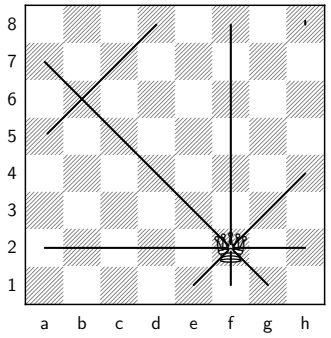
Utöver generaliseringen av problemet har n-queens även utvecklats till att involvera tredimensionella perspektiv och figurer. Det tredimensionella perspektivet på n-queensproblemet tar ett tvådimensionellt schackbräde och betraktar det på en torus. Genom att föra samman höger och vänster sida av schackbrädet och sedan

"vika in" övre och undre kanterna bildas en donut-liknande form. George Pólya har bevisat att det, om n inte är en multipel av två eller tre, alltid existerar minst en lösning [Pó18].



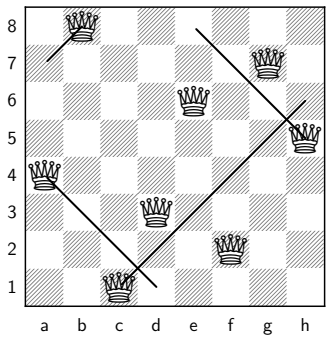
Figur 3: Det modulära schackbrädet illustrerat som en torus.

En intressant aspekt av denna variant av n-queens är att lösningsmängden inte är densamma som för det traditionella problemet. Anledningen är att lösningar som är giltiga på ett platt schackbräde inte nödvändigtvis är giltiga på det modulära, toroidala, schackbrädet. Genom att betrakta det modulära brädet på en torus noteras att rader och kolumner är desamma som för det traditionella brädet, alltså om två damer står i samma rad eller kolumn på det traditionella brädet kommer de även att göra det på det modulära brädet. Diagonalerna däremot är inte desamma, utan två damer som i det traditionella brädet inte hotar varandra via en diagonal kan göra det på det modulära brädet. Anledningen är att diagonalerna i det modulära brädet "fortsätter runt" torusen. I och med detta är n-queens på det modulära brädet fortfarande ett tvådimensionellt problem men en dams räckvidd via diagonalerna har modifierats[Bel09].



Figur 4: Alla rutor en dam på plats `f2` hotar på det modulära brädet, observera att ruta `h8` också är markerad.

Betrakta lösningen i Figur 2. Denna damplacering är giltig enligt ursprungsproblemets formulering men om lösningen betraktas på ett modulärt bräde upptäcks det att lösningen inte längre är giltig. Två par av damer hotar nämligen varandra.



Figur 5: En giltig lösning till standardproblemet n-queens som inte är giltig på det modulära brädet i och med att damerna `a4` och `h5` samt `b8` och `c1` hotar varandra.

1.2 Tillämpningar

Problemet n-queens är, trots sin till synes teoretiska och matematiska natur, ett problem som visat sig vara värdefullt och har idag flera användningsområden. Att problemet samt dess kriterier för en giltig lösning är lätta att definiera i kombination med att antalet lösningar ökar snabbt när värdet på n ökar, gör n-queens till ett bra riktmärke för att testa algoritmer. Dessa algoritmer kan exempelvis vara ämnade att lösa kombinatoriska optimeringsproblem. Utöver rent matematiska användningsområden utnyttjas n-queens vid schemaläggning och uppgiftsdelning där olika aktiviteter inte får krocka. Även inom datorteknik återfinns tillämpningarna av n-queens där principerna från problemet utnyttjas vid resurshantering på en dator för att undvika minnesblock [Abi23].

1.3 Syfte och forskningsfrågor

1.3.1 Syfte

Syftet med denna uppsats är att inledningsvis utforska problemet n-queens och dess historiska utveckling för att senare experimentera med backtrackingalgoritmen. Genom att besöka raderna i ett bräde i olika ordning undersöks det hur sökningen efter lösningar kan optimeras. Med hjälp av programmering kommer det även att undersökas hur många backtracksteg som maximalt tas för olika värden på n samt om det finns något samband mellan de radordningar som kräver flest backtracksteg.

1.3.2 Forskningsfrågor

1. Hur många backtracksteg krävs för att hitta en lösning genom att besöka raderna i en slumpmässig ordning?

2. Hur kan permutationer utformas utifrån en redan känd lösning för att dessa ska generera nya lösningar direkt?

3. Vad är maximala antalet backtracksteg för $n = 4, 5, 6, ..., 9$? Vad är genomsnittet?

4. Finns det något samband mellan de "sämsta" permutationerna och deras inverser samt komplement?

2 Definitioner och notation

I denna uppsats kommer inte klassisk schacknotation att användas när det kommer till hur rutorna på brädet betecknas. Istället för den vardagliga notationen bestående av bokstav + rad, kommer placeringarna bestå av radnummer och kolumnnummer, räknat från övre vänstra hörnet. Exempelvis skulle rutan `a5` i ett 5×5-bräde betecknas med $(1, 1)$.

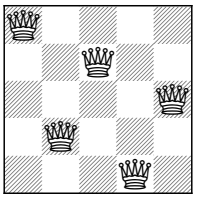
Definition 2.1. Låt ett bräde med n kolumner och n rader ha storlek $n \times n$. En ruta i brädet betecknas (r_a, c_a), där r_a anger raden och c_a kolumnen för dam a där $a = 1, 2, 3, \ldots, n$. Kraven för en giltig lösning blir således:

$$r_a \neq r_b, \quad c_a \neq c_b, \quad |r_a - r_b| \neq |c_a - c_b| \quad \text{för alla} \qquad a \neq b.$$

Definition 2.2. En permutation av talen $\{1, 2, \ldots, n\}$ är en omkastning av dessa tal, det vill säga en ordnad följd där varje tal förekommer exakt en gång. Permutationer kommer att betecknas med σ eller ρ och anges i listnotation:

$$\sigma = [\sigma(1), \sigma(2), \ldots, \sigma(n)]\,.$$

Permutationer kommer att användas för att representera damplaceringar i ett $n \times n$-bräde där $\sigma(i)$ anger raden för damen i kolumn i.



Figur 6: En giltig lösning för $n = 5$ som har permutation $\sigma_{n=5} = [1, 4, 2, 5, 3]$.

Definition 2.3. Identitetspermutationen e uppfyller att $e(i) = i$ för alla $0 < i \leq n$.

Definition 2.4. Varje permutation σ har en invers σ^{-1} som uppfyller att

$$\sigma \circ \sigma^{-1} = \sigma^{-1} \circ \sigma = e.$$

Definition 2.5. Låt $\sigma = [\sigma(1), \sigma(2), \ldots, \sigma(n)]$ vara en permutation av talen $\{1, 2, \ldots, n\}$. Komplementet till σ, betecknat σ^c, definieras som

$$\sigma^c(i) = (n+1) - \sigma(i) \text{ för alla } i = 1, \ldots, n.$$

Definition 2.6. En grupp G är en mängd element som kan vara ändlig eller oändlig. Gruppen G har den binära operatorn $\cdot$ och gruppen är sluten under operatorn. För att mängden element med sin operator ska klassas som en grupp ska kraven för associativitet vara uppfyllda samt att alla element ska ha en invers och identitetselement ska finnas i gruppen.

Definition 2.7. Låt G vara en grupp med identiteten e, gruppoperator α och X är en mängd element. Då definieras en gruppverkan av G på X som

$$\alpha : G \times X \to X, \qquad (g, x) \mapsto g \cdot x.$$

En gruppverkan uppfyller två krav:

$$\alpha(e, x) = x, \qquad g_2(g_1 x) = (g_2 g_1)x.$$

Definition 2.8. Låt G vara gruppen som verkar på mängden X. Banan, Gx för något element x i X betecknas som

$$Gx = \{g \cdot x \mid g \in G\}.$$

Definition 2.9. Den dihedrala gruppen D_{2n} är symmetrigruppen för en liksidig månghörning med n sidor där $n > 1$.

Elementen i en symmetrigrupp betecknas e för identiteten, r beskriver en medurs rotation med $\left(\frac{360}{n}\right)^\circ$ och s betecknar vertikal spegling.

Definition 2.10. Låt $x \in X$ vara en giltig lösning till problemet n-queens. Banan för x definieras som mängden av alla lösningar som kan erhållas genom att applicera symmetrierna d i den dihedrala gruppen för fyrhörningar, D_8, på x:

$$D_8 x = \{\, d \cdot x \mid d \in D_8 \,\}.$$

3 Lösning av n-queensproblemet

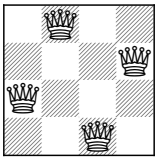
Det finns olika metoder som kan användas för att finna lösningar till n-queens. Denna uppsats kommer att lägga fokus på *backtracking*. Det är en enkel metod som kan användas för att lösa problemet för godtyckliga n som har åtminstone en giltig lösning. Processen för att finna lösningar med backtracking är densamma för alla n, n-queens går att lösa för $n \geq 4$.

3.1 Bevis för $n \geq 4$

Beviset för att det går att hitta minst en giltig lösning för $n \geq 4$ är uppdelat i tre fall beroende på formen av n. Dessa fall kommer i uppsatsen att benämnas Fall A, B och C. Där $j = 1, 2, \ldots, \frac{n}{2}$, k är ett godtyckligt heltal och $n \times n$ är storleken på brädet.

Fall A: n är jämnt och inte på formen $6k + 2$. För varje värde på j kommer två damer åt gången att placeras ut på följande platser:

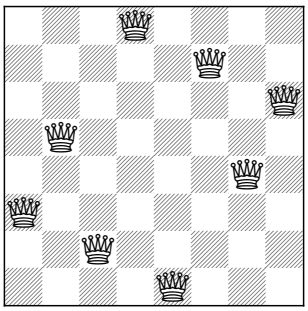
$$(j, 2j) \text{ och } \left(\frac{n}{2} + j, 2j - 1\right).$$



Figur 7: Exempel på lösning av fall A där $n = 4$.

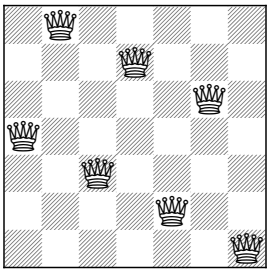
Fall B: n är jämnt och inte på formen $6k$. För att lösa fall B används kongruensräkning och damerna placeras på platserna

$$\left(j, 1 + [2\ (j-1) + \frac{n}{2} - 1 \bmod n]\right) \text{ och } \left(n + 1 - j, n - [2\ (j-1) + \frac{n}{2} - 1 \bmod n]\right).$$



Figur 8: Exempel på lösning av fall B där $n = 8$.

Fall C: n är udda. För udda värden på n används fall A eller B på $n - 1$ (som blir jämnt i och med att n är udda) och avslutningsvis placeras en dam på plats (n, n) för att färdigställa brädet.

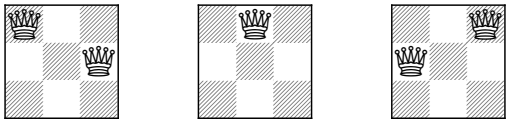


Figur 9: Exempel på lösning av fall C där $n = 7$.

De här tre fallen utgör beviset på att det går att hitta minst en explicit lösning till problemet n-queens för samtliga $n \geq 4$. För att verifiera lösningarna kan kolumner, rader och diagonaler enkelt kontrolleras för att se att ingen dam hotar en annan. När det kommer till Fall A och Fall B existerar det värden på n som varken är på formen $6k + 2$ eller $6k$ till exempel $n = 4$. Dessa värden passar in i båda fallens beskrivningar och för sådana n kan både Fall A och B användas.

För $n < 4$ gäller inte dessa tre fall och det blir därför en process av att testa sig fram för att försöka hitta lösningar eller bevisa att det inte finns några. Lösning för $n = 1$ är trivial, en dam på plats $(1, 1)$. För $n = 2$ och $n = 3$ finns inga giltiga lösningar. Att dessa värden saknar lösningar kan bevisas genom att titta på kraven för en giltig lösning i Definition 2.1. För $n = 2$ finns fyra möjliga placeringar för varje dam $(1, 1)$, $(1, 2)$, $(2, 1)$ och $(2, 2)$. Ingen kombination med två av dessa fyra

platser uppfyller villkoren och därmed går problemet inte att lösa. För $n = 3$ kan vi verifiera att inga giltiga lösningar finns genom att först anta att en dam placeras på plats $(1, 1)$, den enda giltiga platsen på rad två är $(2, 3)$. När vi sedan försöker placera en dam på rad 3 märker vi att ingen av platserna är giltiga och därmed kan en lösning inte hittas. Samma resonemang visar att det inte heller finns någon giltig lösning om den första damen placeras på platserna $(1, 2)$ eller $(1, 3)$.

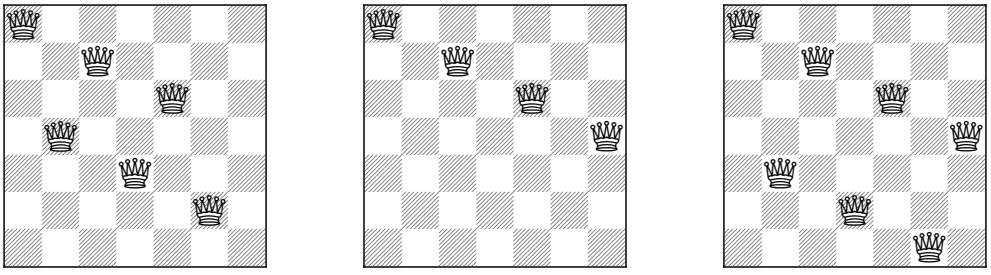


Figur 10: Figuren visar ett försök till att hitta en giltig lösning till ett 3×3-bräde.

3.2 Backtracking

3.2.1 Beskrivning av metoden

En av de mest grundläggande metoderna för att hitta lösningar till problemet n-queens är så kallad *backtracking*. Det är en användbar metod vid lösning av många problem, inte bara n-queens. Förenklat kan backtracking beskrivas som att metodiskt och strukturerat testa sig fram. Genom att gå igenom raderna en i taget och för varje rad placera en dam i den första giltiga rutan, letar man sig fram till en lösning. Om det inte finns någon giltig placering för en dam på en rad backar man tillbaka ett steg och flyttar föregående dam till nästa giltiga placering och testar på nytt. I denna uppsats kommer *ett backtracksteg* att räknas varje gång en rad inte har någon giltig placering alls för en dam.



Figur 11: Exempel på hur backtracking kan se ut för att hitta en lösning för $n = 7$.

Betrakta Figur 11. Genom att börja på plats $(1, 1)$ och gå nedåt en rad i taget placeras en dam på den första giltiga platsen. En lösning fås nästan direkt utan att några steg bakåt behövs men vid sista raden är inga rutor giltiga placeringar för dam nummer 7. Alltså tas nu ett backtracksteg. Den sjätte damen har inga andra giltiga placeringar än den som redan testats på rad sex och därför tas ett till backtracksteg. För den femte damen gäller samma sak och det har nu blivit tre backtracksteg. Dam nummer fyra däremot kommer att få testa en ny plats, den hoppar från $(4, 2)$ till $(4, 7)$. Placeringen av damer fortsätter och dam nummer $5, 6$ och 7 placeras ut på första giltiga placeringen på respektive rad. När detta är färdigt har en giltig lösning erhållits och alltså krävdes tre backtracksteg.

3.2.2 Begränsningar

Backtracking är en viktig metod vid lösning av problemet n-queens i den bemärkelse att metoden kan användas för godtyckliga värden på n och inte kräver komplexa matematikkunskaper. Metoden har däremot sina nackdelar, speciellt när det kommer till större n eller när målet är att finna samtliga lösningar. Det framgår relativt snabbt hur vägarna blir fler och stegen i backtracking ökar i både antal och längd vilket medför att metoden blir ineffektiv. För att hitta samtliga lösningar, även för mindre n, kommer många placeringar av damer att behöva testas vilket gör att det blir mer och mer orealistiskt att få fram samtliga lösningar genom att enbart använda backtracking för hand.

4 Gruppverkan och n-queens

4.1 Gruppteori

Som tidigare nämnts växer antalet lösningar till n-queens mycket snabbt då värdet på n ökar, i och med detta blir en viktig del i hanteringen av lösningarna att betrakta så kallade symmetrier. Varje giltig lösning till n-queens kan nämligen, genom rotationer och speglingar, ge upphov till fler lösningar. När man talar om att räkna hur många lösningar som finns för ett visst värde på n kan man antingen räkna samtliga lösningar eller endast alla unika lösningar, alltså en lösning och dess symmetrier "klumpas ihop" och räknas som en. En unik lösning och dess symmetrier bildar en mängd lösningar, $M_{\text{symmetrier}}$, där

$$M_{\text{symmetrier}} = \left\{ \begin{array}{l} \text{identitet}, \\ \text{rotation } 90^\circ \text{ medurs}, \\ \text{rotation } 180^\circ, \\ \text{rotation } 270^\circ \text{ medurs}, \\ \text{vertikal spegling}, \\ \text{vertikal spegling och rotation } 90^\circ \text{ medurs}, \\ \text{vertikal spegling och rotation } 180^\circ, \\ \text{vertikal spegling och rotation } 270^\circ \text{ medurs} \end{array} \right\}.$$

Denna mängd kan mer kompakt skrivas som

$$M_{\text{symmetrier}} = \{e, r, r^2, r^3, s, sr, sr^2, sr^3\}.$$

Mängden $M_{symmetrier}$ är faktiskt inte bara en mängd, utan det är en grupp. Mer specifikt är mängden symmetrier den dihedrala gruppen D_8. Alltså $M_{symmetrier} = D_8 = \{e, r, r^2, r^3, s, sr, sr^2, sr^3\}$.

4.1.1 Bevis av $M_{symmetrier} = D_8$

Beviset utgår ifrån kraven för en grupp i Definition 2.6.

Bevis av slutenhet: Beviset av slutenhet är uppdelat i tre delbevis. Eftersom det är ett kvadratiskt schackbräde som behandlas kommer en rotation att bestå av 90°. I och med detta resulterar fyra rotationer i samma utseende som noll rotationer,

alltså är $r^4 = e$. Rotationer kan därför beräknas modulo fyra. Generellt har vi även att $s^2 = e$. Utifrån elementen i D_8 kan vi också få sambandet att $sr = r^3 s$ som vidare innebär att $sr^k = r^{-k}s$. Alla de här sambanden kommer att användas nedan för att bevisa att oavsett vilka två element som sammansätts från gruppen kommer det nya elementet vara i gruppen.

Produkt av två speglingar är en rotation. Låt sr^i och sr^j vara speglingar där $i, j \in \{0, 1, 2, 3\}$. Då är

$$(sr^i)(sr^j) = sr^i sr^j = r^{-i} ssr^j = r^{-i} er^j = r^{-i} r^j = r^{j-i}.$$

Rotationen r^{j-i} ingår i mängden $\{e, r, r^2, r^3, s, sr, sr^2, sr^3\}$ i och med att rotationerna betraktas modulo fyra.

Produkt av en spegling och rotation är en spegling. Låt r^m vara en rotation och sr^n en spegling, där $m, n \in \{0, 1, 2, 3\}$, då är

$$r^m(sr^n) = r^m sr^n = sr^{-m} r^n = sr^{n-m}.$$

Eftersom rotationer beräknas modulo fyra är $sr^{n-m} \in \{e, r, r^2, r^3, s, sr, sr^2, sr^3\}$.

Produkt av två rotationer är en rotation. Låt r^a och r^b vara rotationer, där $a, b \in \{0, 1, 2, 3\}$. Då får vi att

$$r^a r^b = r^{a+b}$$

som finns i mängden $\{e, r, r^2, r^3, s, sr, sr^2, sr^3\}$ i och med rotationers tolkning modulo fyra.

Tack vare de här tre delbevisen är det nu klart att gruppen är sluten under sin operator. Eftersom gruppoperatorn är komposition av funktioner och dessa alltid är associativa, uppfyller $M_{symmetrier}$ kravet för associativitet. Identiteten e finns i gruppen och för att bevisa att alla element har precis en invers kan nedanstående Cayleytabell betraktas.

	e	r	r^2	r^3	s	sr	sr^2	sr^3
e	e	r	r^2	r^3	s	sr	sr^2	sr^3
r	r	r^2	r^3	e	sr^3	s	sr	sr^2
r^2	r^2	r^3	e	r	sr^2	sr^3	s	sr
r^3	r^3	e	r	r^2	sr	sr^2	sr^3	s
s	s	sr	sr^2	sr^3	e	r	r^2	r^3
sr	sr	sr^2	sr^3	s	r^3	e	r	r^2
sr^2	sr^2	sr^3	s	sr	r^2	r^3	e	r
sr^3	sr^3	s	sr	sr^2	r	r^2	r^3	e

Figur 12: Visar Cayley-tabell för dihedrala gruppen D_8.

På varje rad och i varje kolumn återfinns identiteten precis en gång, vilket innebär att alla element har precis en invers som, vid komposition, ger identiteten. Därmed är det bevisat att $M_{symmetrier}$ är gruppen D_8.

4.2 Permutationer och gruppverkan

En giltig lösning till n-queens har precis en dam i varje rad och kolumn. Denna konstruktion kan beskrivas med hjälp av en lista där varje element i listan beskriver vilken rad en dam står på och index, alltså vilken plats i listan elementet har, beskriver vilken kolumn respektive dam står i. Exempel på en giltig lösning i listform då $n = 4$ är $L_4 = [2, 4, 1, 3]$, där damerna är placerade i rutorna $(2, 1), (4, 2), (1, 3), (3, 4)$. Uppmärksamma att listan L_4 innehåller talen $1, 2, 3, 4$ precis en gång och kan betraktas som en permutation $\sigma \in S_4$, alltså en permutation av talen $1, 2, 3, 4$. Detta kan generaliseras och en giltig lösning till n-queens kan beskrivas av permutationen $\sigma_n = [\sigma(1), \sigma(2), ..., \sigma(n)]$. Givet att varje giltig lösning kan representeras som en permutation av siffrorna $1, 2, ..., n$ kan vi betrakta hur gruppen D_8 verkar på en sådan permutation. Ett element $d \in D_8$ definierar en bijektiv transformation av schackbrädet. Denna transformation bevarar giltigheten hos lösningen, i och med detta induceras en operation på permutationerna som beskrivs enligt:

$$\cdot : D_8 \times S_n \to S_n, \qquad (d, \sigma_n) \mapsto d \cdot \sigma_n.$$

Detta kommer att beskriva en gruppverkan av D_8 på mängden lösningar. Att det

är en gruppverkan kan verifieras genom att betrakta kraven i Definition 2.7. Kravet gällande identitet uppfylls i och med att identiteten innebär att inget förändras i D_8 och därmed stämmer $e \cdot \sigma_n = \sigma_n$ för alla $\sigma_n \in X$. Det andra kravet om kompatibilitet uppfylls i och med att gruppoperationen i D_8 är en funktionskomposition, som alltid är associativ. Eftersom båda kraven för gruppverkan uppfylls är verkan av D_8 på lösningsmängden X en gruppverkan. Därmed har varje giltig lösning $\sigma_n \in X$ en bana definierad som

$$D_8 \cdot \sigma_n = \{d \cdot \sigma_n \mid d \in D_8\}.$$

Banan representerar mängden av lösningarna som fås genom att applicera D_8 på den giltiga placeringen σ_n, alltså kan ytterligare sju lösningar fås från en giltig lösning.

5 Metod

5.1 Permutationer: slumpmässiga och avsiktliga

För att undersöka hur snabbt en lösning kan finnas, alltså hur många backtracksteg som behövs, givet att raderna söks i ordning av en viss permutation användes dels slumpmässiga permutationer, dels avsiktligt konstruerade permutationer. Storleken på brädet som behandlades var 7×7.

Till att börja med skapades två slumpmässiga permutationer med hjälp av Pythonprogrammet i Figur 19. Dessa permutationer utgjorde grunden för det första experimentet. En giltig lösning togs fram genom backtracking, där raderna besöktes i den ordning som permutationerna angav. Metoden som användes för att finna en lösning var backtracking där varje backtracksteg antecknades. När en giltig lösning hittades antecknades denna.

Den andra delen av experimentet gick ut på att undersöka hur permutationer kan utformas för att generera en giltig lösning direkt, alltså utan att några backtracksteg behövde utföras. För att utforma permutationer som ger en giltig lösning direkt utnyttjas gruppverkan av D_8 som finns på lösningsmängden till n-queens. Permutationen $\rho_1 = [4, 1, 5, 2, 6, 3, 7]$ beskriver damkonfigurationen i Figur 9, på denna permutation utfördes olika permutationsoperationer för att erhålla nya permutationer. Från ρ_1 skapades en invers och ett komplement samt att en omkastning utfördes där ordningen på elementen i permutationen vändes. Dessa tre operationer kombinerades också med varandra. Var och en av dessa permutationer utgjorde varsin ordning som raderna i 7×7-brädet skulle sökas i. Förhoppningen med att göra detta var att samtliga element som finns i banan till ρ_1 skulle fås av de nya permutationerna. Som för de slumpmässiga permutationerna antecknades även här alla backtracksteg och slutgiltiga lösningar.

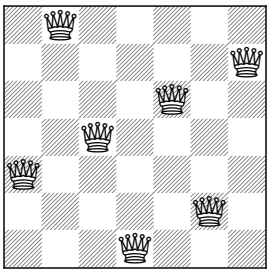
Den sista delen av experimenten involverade programmering för att undersöka maximala antal backtracksteg för $4 \leq n \leq 9$. Programmet som användes finns i Figur 20 och 21 i Appendix. Dessa två program skapades med hjälp av AI. All användning av och beräkning med hjälp av programmen har AI **inte** använts till.

6 Resultat

6.1 Slumpmässiga permutationer

6.1.1 $\sigma_1 = [5, 4, 1, 6, 7, 2, 3]$

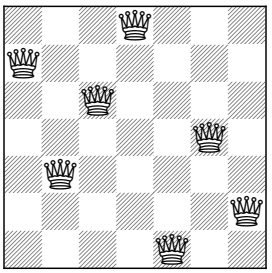
Den första slumpmässiga permutationen som skapades var $\sigma_1 = [5, 4, 1, 6, 7, 2, 3]$ och för att se hur snabbt denna gav en lösning placerades först en dam på rad fem i ruta $(5, 1)$. Nästa dam placerades på rad fyra och första giltiga placeringen var $(4, 3)$. Den tredje damen placerades på plats $(1, 2)$, den fjärde på plats $(6, 4)$, den femte damen i ruta $(7, 7)$ och sedan placerades den sjätte på plats $(2, 6)$. När det kom till att placera den sista damen på rad tre fanns inga giltiga placeringar och därmed fick ett backtracksteg tas. Varken damen på rad två eller sju hade ytterligare giltiga placeringar att testa. Därför togs två backtracksteg. Damen på rad sex flyttades till ruta $(6, 6)$. Efter detta kunde den femte, sjätte och sjunde damen placeras ut på respektive platser: $(7, 4)$, $(2, 7)$ och $(3, 5)$. Nu har en giltig lösning erhållits och det krävdes tre backtracksteg för att få fram en giltig lösning när raderna söktes enligt $\sigma_1 = [5, 4, 1, 6, 7, 2, 3]$.



Figur 13: Den giltiga lösningen som erhölls av permutationen σ_1.

6.1.2 $\sigma_2 = [2, 3, 5, 7, 4, 6, 1]$

Den andra slumpmässiga permutationen som skapades var $\sigma_2 = [2, 3, 5, 7, 4, 6, 1]$. Processen att använda backtracking för att finna en giltig lösning gick till på samma sätt som för σ_1. Vid sökning av raderna enligt $\sigma_2 = [2, 3, 5, 7, 4, 6, 1]$ erhölls en giltig lösning utan att några backtracksteg behövdes. Lösningen som gavs var:



Figur 14: Den giltiga lösningen som erhölls av permutationen σ_2.

6.2 Permutationer utifrån en känd lösning

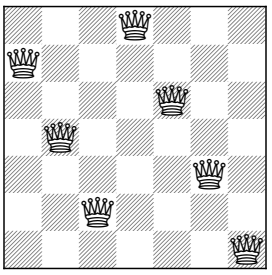
Den här delen i uppsatsen kommer att undersöka hur många backtracksteg som behövs för att finna en giltig lösning när raderna söks i ordning av permutationer skapade utifrån $\rho_1 = [4, 1, 5, 2, 6, 3, 7]$.

Permutationerna som skapades var:

- Inversen av ρ_1:
  $\rho_1^{-1} = [2, 4, 6, 1, 3, 5, 7]$
- Komplementet till ρ_1:
  $\rho_1^c = [4, 7, 3, 6, 2, 5, 1]$
- Omvänd ordning av ρ_1:
  $\rho_1^r = [7, 3, 6, 2, 5, 1, 4]$
- Komplementet till inversen ρ_1^{-1}:
  $\rho_{i+c} = [6, 4, 2, 7, 5, 3, 1]$
- Omvänd ordning av inversen:
  $\rho_{i+r} = [7, 5, 3, 1, 6, 4, 2]$
- Omvänd ordning av komplementet:
  $\rho_{c+r} = [1, 5, 2, 6, 3, 7, 4]$
- Omvänd ordning av komplementet till inversen:
  $\rho_{i+c+r} = [1, 3, 5, 7, 2, 4, 6]$.

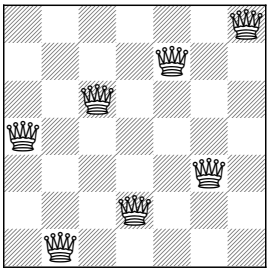
Genom att utföra operationer på permutationen $\rho_1 = [4, 1, 5, 2, 6, 3, 7]$ kunde sju ytterligare permutationer skapas. Genom att utföra operationerna i andra ordningar än de som nämndes ovan fås inga ytterligare permutationer. Nu till huvudexperimentet, hur snabbt ger dessa permutationer en giltig damplacering på ett 7×7-bräde?

Inledningsvis testades ursprungspermutationen $\rho_1 = [4, 1, 5, 2, 6, 3, 7]$ och denna gav lösningen i Figur 9 utan några backtracksteg. När ρ_1^{-1} testades visade det sig att även den gav en giltig damkonfiguration direkt, mer specifikt erhölls följande placering:

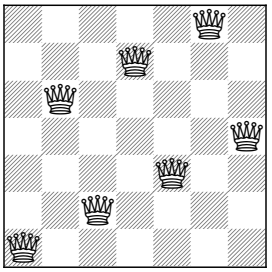


Figur 15: Lösning av $\rho_1^{-1} = [2, 4, 6, 1, 3, 5, 7]$.

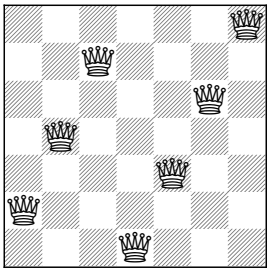
Uppmärksamma att denna placering av damer representerar en spegling av Figur 9 längs diagonalen som går från övre vänstra hörnet till nedre högra hörnet. Vidare undersöktes resterande sex permutationer och det visade sig att samtliga resulterade i att en giltig lösning erhölls utan att några backtracksteg behövdes.



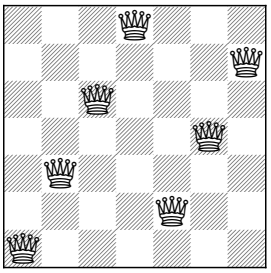
$\rho_1^c = [4, 7, 3, 6, 2, 5, 1]$



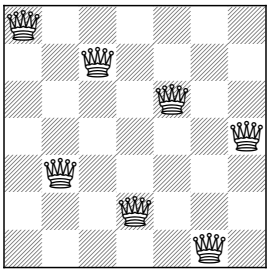
$\rho_1^r = [7, 3, 6, 2, 5, 1, 4]$



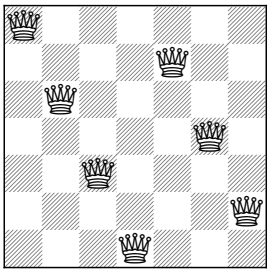
$\rho_{i+c} = [6, 4, 2, 7, 5, 3, 1]$



$\rho_{i+r} = [7, 5, 3, 1, 6, 4, 2]$



$\rho_{c+r} = [1, 5, 2, 6, 3, 7, 4]$



$\rho_{i+c+r} = [1, 3, 5, 7, 2, 4, 6]$.

Figur 16: Damplaceringarna som gavs av de resterande permutationerna skapade utifrån ρ_1.

En viss struktur tycks existera bland ovanstående damplaceringar, lösningarna liknar varandra men är vridna eller spegelvända i förhållande till varandra. Genom

att jämföra de sju damplaceringar som erhållits med den ursprungliga lösningen i Figur 9 är det tydligt att dessa är olika rotationer och speglingar av den lösningen.

6.3 Maximalt antal backtracksteg

n	maximalt antal backtracksteg	antal permutationer	snitt (backtracksteg)
$n = 4$	7	2	2.833
$n = 5$	4	6	0.667
$n = 6$	70	2	22.939
$n = 7$	43	4	6.990
$n = 8$	196	6	28.420
$n = 9$	401	2	34.593

Figur 17: Visar maximala antalet backtracksteg per n, antal permutationer som hade så många steg samt vad snittet antalet backtracksteg var per n.

I Figurerna 22 till 27 i Appendix återfinns permutationerna som krävde flest backtracksteg och hur många steg deras inverser samt komplement krävde för att finna en giltig lösning.

7 Diskussion

Resultaten i denna uppsats visar att ordningen i vilken raderna söks igenom har stor betydelse för hur snabbt en giltig lösning hittas. För ett bräde av storlek 7×7 behövs mellan noll och 43 stycken backtracksteg. Alltså är det inte effektivt att testa sig fram med hjälp av slumpmässiga permutationer för att försöka finna samtliga lösningar till n-queensproblemet.

En metod som däremot visade sig var effektiv för att finna ytterligare lösningar, om en lösning redan var känd, var att utforma permutationer för ordningen utifrån den kända lösningen. Utifrån en giltig lösning kunde åtta permutationer skapas, inklusive permutationen som beskrev den lösningen. När brädet söktes i ordningen av dessa permutationer gavs en giltig lösning utan några backtracksteg. Som nämndes i resultatdelen liknade de olika lösningarna varandra men var vridna och spegelvända. Detta är inte ett slumpmässigt fenomen utan har sin förklaring i gruppteori. När en invers eller ett komplement skapas eller när ordningen på elementen vänds i en permutation skapas nya permutationer som, geometriskt, representerar olika vridningar och speglingar.

I den här uppsatsen var lösningen i Figur 9 med permutation $\rho_1 = [4, 1, 5, 2, 6, 3, 7]$ utgångspunkt för experimenten med att skapa nya permutationer utifrån en känd giltig lösning. Genom att jämföra Figur 9 med lösningen i Figur 15 som gavs av inversen till ρ_1, upptäcks att inversen motsvarar en spegling längs diagonalen från ruta $(1, 1)$ till $(7, 7)$ eller en vertikal spegling längs kolumn 4 plus en rotation 270° medurs. Komplementet $\rho_1^c = [4, 7, 3, 6, 2, 5, 1]$ till ρ_1 motsvarar en horisontell spegling längs rad 4 eller en vertikal spegling plus medurs rotation 180°. Vänds ordningen på ρ_1, fås permutationen $\rho_1^r = [7, 3, 6, 2, 5, 1, 4]$ som motsvarar en vertikal spegling. Resterande permutationer motsvarar följande speglingar och rotationer av originallösningen ρ_1:

- komplement av inversen $\rho_{i+c} = [6, 4, 2, 7, 5, 3, 1]$ motsvarar medurs rotation med 270°,
- omvänd ordning av inversen $\rho_{i+r} = [7, 5, 3, 1, 6, 4, 2]$ motsvarar en medurs rotation med 90°,
- omvänd ordning av komplementet $\rho_{c+r} = [1, 5, 2, 6, 3, 7, 4]$ motsvarar 180° rotation medurs,
- omvänd ordning av komplementet till inversen $\rho_{i+c+r} = [1, 3, 5, 7, 2, 4, 6]$ motsvarar en vertikal spegling plus 90° medurs rotation.

Genom att jämföra dessa sju speglingar och rotationer med elementen i $M_{symmetrier}$ som togs upp i avsnitt 4.1 upptäcks det att samtliga återfinns i den mängden. Det som saknas är identiteten e men denna är ju ursprungslösningen i Figur 9. Mängden $M_{symmetrier}$ har tidigare fastställts att motsvara den dihedrala gruppen D_8 för kvadrater. Genom att utföra olika operationer på ursprungspermutationen har vi alltså lyckats bilda samtliga element i D_8. Nedan följer en tabell som visar specifikt vilka element i gruppen som de olika permutationerna gav.

permutation	element i D_8
$\rho_1 = [4,1,5,2,6,3,7]$	e
$\rho_{i+r} = [7,5,3,1,6,4,2]$	r
$\rho_{c+r} = [1,5,2,6,3,7,4]$	r^2
$\rho_{i+c} = [6,4,2,7,5,3,1]$	r^3
$\rho_1^r = [7,3,6,2,5,1,4]$	s
$\rho_{i+c+r} = [1,3,5,7,2,4,6]$	sr
$\rho_1^c = [4,7,3,6,2,5,1]$	sr^2
$\rho_1^{-1} = [2,4,6,1,3,5,7]$	sr^3

Figur 18: Visar alla permutationer som erhölls utifrån ρ_1 samt vilket element i D_8 deras lösning motsvarar.

Rent gruppteoretiskt kan detta tolkas som banan för ρ_1, alltså om gruppen D_8 får verka på $\rho_1 = [4,1,5,2,6,3,7]$ fås följande:

$$D_8 \cdot \rho_1 = \{[4,1,5,2,6,3,7],\ [7,5,3,1,6,4,2],\ [1,5,2,6,3,7,4],\ [6,4,2,7,5,3,1],\\ [7,3,6,2,5,1,4],\ [1,3,5,7,2,4,6],\ [4,7,3,6,2,5,1],\ [2,4,6,1,3,5,7]\}.$$

Även om de här resultaten i uppsatsen är framtagna för just $n = 7$ är det rimligt att anta att samma resultat kan åstadkommas för alla värden på n där giltiga lösningar existerar. Detta beror på att resonemangen kring symmetrier och permutationer inte bygger på brädets storlek utan på dess kvadratiska struktur. För varje $n \geq 4$ kan åtminstone en explicit lösning hittas och denna kan representeras som en permutation. Verkan av D_8 på denna lösning bevarar angreppsrelationerna mellan damer vilket innebär att nya lösningar som genereras med hjälp av D_8 kommer att vara giltiga.

När det kom till att undersöka vilka permutationer som var minst effektiva att söka efter en giltig damplacering kan en iakttagelse göras gällande deras struktur. I

majoriteten av de undersökta fallen inledde de "sämsta" permutationerna med att placera en dam i ett hörn. Att placera den första damen i ett hörn medför att största möjliga antal rutor elimineras för kommande damer i och med att den damen dels hotar hela sin rad och kolumn men även en maximal diagonal. Att slå ut ett stort antal rutor redan vid första placeringen leder till att backtrackingalgoritmen blir mer begränsad och därmed kan sökträdet få många återvändsgränder.

Vad som också var tydligt var att antalet långsammaste permutationer var jämnt för alla n som undersöktes. Anledningen till detta är symmetri. Varje permutation motsvarar en specifik ordning i vilken raderna söks. En horisontell spegling av schackbrädet kommer att byta rad i mot $n + 1 - i$ men bevara konfliktmönstret och därmed kommer en horisontell spegling av en permutation att ge upphov till ett sökträd som är isomorft med den ursprungliga permutationens och har exakt lika många backtracksteg. Därmed måste antalet permutationer med maximalt antal backtracksteg, eller egentligen med ett godtyckligt antal backtracksteg, uppträda parvis. Maximala antalet backtracksteg för olika värden på n verkade inte följa något mönster eller uppenbar struktur.

En intressant upptäckt kan göras ur tabellen i Figur 17 utifrån det genomsnittliga antalet backtracksteg. För $n = 5$ är snittet endast 0.667 vilket betyder att en stor andel av alla möjliga permutationer $\sigma \in S_5$ inte kräver några backtracksteg alls medan $n = 6, 8$ och 9 alla har värden över 20. Betrakta värdena för $n = 7$. Jämfört med $n = 6, 8$ har $n = 7$ både ett lågt snitt och litet maximalt antal backtracksteg. Detta antyder att permutationerna $\sigma \in S_7$ är mer tacksamma för n-queens än vad $\sigma \in S_6$ och $\sigma \in S_8$ är.

Betrakta tabellerna i Figurerna 22 till 27 i Appendix. Av de tabellerna framkommer att komplementen till de "sämsta" permutationerna också har maximalt antal backtracksteg och även de är alltså "sämst". Att komplementen till de minst effektiva permutationerna är lika "dåliga" på att hitta en giltig lösning snabbt är kopplat till det som nämndes i ett tidigare stycke gällande horisontella spegelbilder. I Figur 16 kunde vi även se geometriskt att komplementet motsvarade en horisontell spegling av originalplaceringen och det gör att en permutations komplement är lika "dålig" eller "bra" som originalpermutationen på att finna en lösning snabbt. När det kommer till inverser däremot, syns inte samma resultat utan inverser till de "sämsta" permutationerna kan kräva både få eller många backtracksteg innan en lösning hittas. Av Figur 15 framgår det att inversen motsvarar en vertikal spegling följt av 270° rotation medurs. Detta representerar en diagonal spegling som byter

plats på rader och kolumner, en handling som inte bevarar konfliktmönstret och därmed inte heller garanterar att en invers har samma antal backtracksteg som dess ursprungspermutation.

8 Slutsats

Syftet med den här uppsatsen var att utforska problemet n-queens och dess koppling till gruppteori genom att undersöka vilken påverkan sökordningen av raderna har för antalet backtracksteg som behövs innan en giltig lösning erhålls. Med fokus på ett 7×7-bräde testades att söka efter en lösning enligt en slumpmässig permutation av raderna ett till sju. Vidare analyserades även hur permutationer kan utformas utifrån en redan känd lösning och hur snabbt dessa permutationer finner en lösning.

Resultaten visade att slumpmässiga permutationer kan leda till komplicerade sökträd med flera backtracksteg och är alltså inte ett effektivt sätt att leta efter samtliga lösningar till n-queensproblemet. Att utforma permutationer utifrån en redan känd lösning däremot, det visade sig vara mycket effektivt. Permutationerna som skapades på detta sätt gav en giltig lösning direkt, alltså utan några backtracksteg. Anledningen till detta ligger hos den gruppverkan som finns där den dihedrala gruppen D_8 verkar på lösningsmängden. Genom att göra olika operationer på den kända lösningens permutation kunde samtliga element i dess bana genereras. Det är alltså mycket effektivt att använda symmetri och gruppverkan för att finna fler lösningar utifrån en känd giltig lösning.

Med hjälp av programmering har det kunnat visas att en permutation och dess komplement alltid kräver lika många backtracksteg för att finna en giltig lösning. Pythonprogrammen hjälpte även till att undersöka maximala antalet backtracksteg för olika n samt genomsnittliga antalet backtracksteg för dessa värden på n.

Denna uppsats har fokuserat på $n = 7$ i sina undersökningar men det kan anses högst troligt att resultaten och insikterna kan generaliseras till andra värden på n, både större och mindre, förutsatt att giltiga lösningar existerar. Anledningen till det är att resonemangen kring permutationer, symmetrier och gruppverkan inte är byggda på brädets storlek utan istället på dess kvadratiska struktur.

Även om resultaten rimligtvis kan antas generaliserbara till andra värden på n är det en begränsning hos studien att endast ett värde fokuserades på. Vidare studier skulle därför naturligt kunna vara att titta på fler värden på n och se om samma slutsatser kring lösningar och gruppverkan kan dras. Vidare skulle det vara intressant att byta perspektivet för studien och snarare än att fokusera på enskilda permutationer, utgå ifrån symmetrierna och undersöka hur olika symmetrier påverkar sökträdet och därmed vilka som kanske effektiviserar eller komplicerar sökningen. På så sätt kan en ineffektiv permutation eventuellt optimeras genom att först transformeras med

hjälp av en symmetri som effektiviserar sökträdet och sedan finna en lösning enklare med backtracking.

Det som undersöks gällande maximala antalet backtracksteg för olika n verkar vara ett problem som inte studerats tidigare och resultaten från denna studie kan i någon mening därför antas vara helt nya gällande det. Det hade varit intressant att titta på maximala antalet backtracksteg för större n än vad den här uppsatsen gjorde och se om någon struktur kan finnas eller om talföljden som beskriver maximala antalet steg har något mönster. Det nämndes att det verkar existera en skillnad mellan olika n på hur stor andel av alla permutationer som hittar en lösning direkt och detta hade varit intressant att titta närmare på.

Backtracking är en metod som inte bara används för n-queensproblemet och intressanta framtida studier skulle därför kunna inkludera hur gruppverkan och resultaten från den här studien skulle kunna appliceras på ett annat problem, exempelvis sudoku.

Sammanfattningsvis belyser uppsatsen hur matematiska verktyg som permutationer och gruppteori kan användas vid analysen av algoritmiska metoder.

Referenser

[Abi23] Bouneb Zine El Abidine. An incremental approach to the n-queen problem with polynomial time. *Journal of King Saud University – Computer and Information Sciences*, 2023. Available online 14 February 2023.

[Bel09] B Bell, J. Stevens. A survey of known results and research areas for n-queens. *Discrete Mathematics*, 309:1–31, 2009. Received 14 Sep 2005; revised 28 Nov 2007; accepted 13 Dec 2007; available online 4 Mar 2008. doi:10.1016/j.disc.2008.12.010.

[Bez48] M. Bezzel. Proposal of 8-queens problem. *Berliner Schachzeitung*, 3:363, 1848. Skriver under pseudonymen "Schachfreund".

[MN26] H. Møller Nielsen. The n-queens solution count $q(n)$ is divisible by 4. *arXiv preprint*, arXiv:2601.05856, 2026. arXiv:2601.05856 [math.CO]. URL: https://arxiv.org/abs/2601.05856.

[Pó18] G. Pólya. Über die doppelt-periodischenlösungen des n-damen-problems. In W. Ahrens, editor, *Mathematische Unterhaltungen und Spiele*, volume 2, pages 364–374. B.G. Teubner, 2 edition, 1918.

[Slo25] N. J. A. Sloane. The On-Line Encyclopedia of Integer Sequences, Sequence A000170: Number of solutions to the n-queens problem. https://oeis.org/A000170, 2025. Accessed: 2025-10-16.

9 Appendix

```
1 import random
2
3 num = [1,2,3,4,5,6,7]
4 random.shuffle(num)
5
6 print(num)
```

Figur 19: Pythonprogram som skapar slumpmässiga permutationer av siffrorna ett till sju.

```python
#!/usr/bin/env python3
from typing import List, Optional, Tuple, Dict
from collections import Counter
from itertools import permutations
import random, sys

def nqueens_by_row_order(pi: List[int]) -> Tuple[Optional[List[int]], int]:
    """
    Backtracking of n-queens by visiting the rows in the order of the permutation "pi" given in list notation and tries columns left-to-right.
    Returns (solution, backtracks).
    """
    n = len(pi)
    if sorted(pi) != list(range(1, n + 1)):
        raise ValueError("pi must be a permutation of 1..n")

    row_order = [r - 1 for r in pi]
    solution = [-1] * n
    cols, diag1, diag2 = set(), set(), set()
    backtracks = 0

    def dfs(depth: int) -> bool:
        nonlocal backtracks
        if depth == n:
            return True
        r = row_order[depth]
        placed_any = False
        for c in range(n):
            if (c in cols) or ((r - c) in diag1) or ((r + c) in diag2):
                continue
            placed_any = True
            solution[r] = c
            cols.add(c); diag1.add(r - c); diag2.add(r + c)
            if dfs(depth + 1):
                return True
            cols.remove(c); diag1.remove(r - c); diag2.remove(r + c)
            solution[r] = -1
        backtracks += 1
```

```
38          return False
39
40      ok = dfs(0)
41      if not ok:
42          return None, backtracks
43      return [c + 1 for c in solution], backtracks
44
45  def tally_backtracks(n: int):
46      """
47      Enumerates all permutations of the rows {1, ..., n} and runs
            the
48      backtracking algorithm for each of them.
49
50      The function notes how many permutations require a certain
51      amount of backtrack steps.
52      """
53      tally = Counter()
54      mapping = {}
55      for pi in permutations(range(1, n + 1)):
56          sol, bt = nqueens_by_row_order(list(pi))
57          tally[bt] += 1
58          mapping.setdefault(bt, []).append(pi)
59      return dict(tally), mapping
60
61  def sample_tally_backtracks(n: int, trials: int, seed: Optional[int
        ] = None):
62      """
63      This function is used when one does not want to try all
64      permutations of the rows {1, ..., n}. The function tries a
            random
65      sample from all n! possible permutations.
66
67      The function notes how many permutations require a certain
68      amount of backtrack steps.
69      """
70      if seed is not None:
71          random.seed(seed)
72      base = list(range(1, n + 1))
73      tally = Counter()
74      mapping = {}
75      for _ in range(trials):
76          pi = base[:]
```

```
77          random.shuffle(pi)
78          sol, bt = nqueens_by_row_order(pi)
79          tally[bt] += 1
80          mapping.setdefault(bt, []).append(tuple(pi))
81      return dict(tally), mapping
82
83  def main():
84      """
85      This function handles the user inputs and decides which other
86      functions to call.
87      """
88      while True:
89          try:
90              n = int(input("Enter board size n (>=1): "))
91              if n < 1:
92                  raise ValueError
93              break
94          except ValueError:
95              print("Invalid input. Please enter an integer >= 1.")
96
97      # sample
98      s = input("Enter number of random samples (0 or blank =
            exhaustive): ").strip()
99      sample = int(s) if s.isdigit() else 0
100
101     # seed
102     s = input("Enter seed (blank = none): ").strip()
103     seed = int(s) if s.isdigit() else None
104
105     # CSV
106     csv = input("Enter CSV filename (blank = do not write CSV): ").
            strip() or None
107
108     # Compute
109     if sample > 0:
110         tally, mapping = sample_tally_backtracks(n, sample, seed)
111         total = sample
112         mode = f"sample of {sample}"
113     else:
114         tally, mapping = tally_backtracks(n)
115         total = sum(tally.values())
116         mode = "exhaustive"
```

```
117
118 # Compute average backtracks per permutation
119     total_bt_sum = sum(
120         bt * count for bt, count in tally.items()
121     )
122
123     avg_bt = total_bt_sum / total if total > 0 else None
124
125     # Output
126     print(f"# n = {n} ({mode})")
127     print("# backtracks : permutations")
128     for k in sorted(tally):
129         print(f"{k:>11} : {tally[k]}")
130     print(f"# total permutations tallied = {total}")
131
132     if avg_bt is not None:
133         print(f"# Average backtracks per successful permutation = {
                avg_bt:.4f}")
134     else:
135         print(" No successful permutations, average undefined.")
136
137     # Worst case permutations
138     if sample == 0:
139         max_bt = max(tally.keys())
140         worst = mapping[max_bt]
141         print("\n# Determining worst (max-backtrack) permutations
                ...")
142         print(f"# Max backtracks = {max_bt}")
143         print(f"# Number of worst-case permutations = {len(worst)}"
                )
144         print("# Worst permutations:")
145         for pi in worst:
146             print(pi)
147
148     # Write to CSV
149     if csv:
150         if not csv.lower().endswith(".csv"):
151             csv = csv + ".csv"
152
153         with open(csv, "w", encoding="utf-8") as f:
154             f.write("backtracks,count,permutations\n")
155             for k in sorted(tally):
```

```
156                 perms = mapping[k]
157                 perm_str = ";".join("".join(map(str, p)) for p in
                        perms)
158                 f.write(f"{k},{tally[k]},{perm_str}\n")
159 
160             if avg_bt is not None:
161                 f.write(f"\nAverage_backtracks,{avg_bt},\n")
162 
163         print(f"# wrote CSV to {csv}")
164 
165 if __name__ == "__main__":
166     main()
```

Figur 20: Pythonprogram som testar alla permutationer av talen $1, 2, ..., n$ som sökordningar och noterar antal backtracksteg, antal permutationer med dessa antal steg samt vilka permutationer som har flest antal backtracksteg. Den skriver även ut genomsnittliga antalet backtracksteg.

```python
    #!/usr/bin/env python3
from itertools import permutations
from typing import List, Tuple, Optional

def nqueens_by_row_order(pi: List[int]) -> Tuple[Optional[List[int]], int]:
    """
    This function backtracks n-queens by visiting rows in the order of different permutations of the rows {1, ..., n}.

    The function returns a tuple where entries consists of comulmn positions for the queens as well as the amount of backtrack steps needed for that specific solution.
    """
    n = len(pi)
    if sorted(pi) != list(range(1, n + 1)):
        raise ValueError("pi must be a permutation of 1..n")

    row_order = [r - 1 for r in pi]
    solution = [-1] * n
    cols, diag1, diag2 = set(), set(), set()
    backtracks = 0

    def dfs(depth: int) -> bool:
        nonlocal backtracks
        if depth == n:
            return True
        r = row_order[depth]
        for c in range(n):
            if (c in cols) or ((r - c) in diag1) or ((r + c) in diag2):
                continue
            solution[r] = c
            cols.add(c); diag1.add(r - c); diag2.add(r + c)
            if dfs(depth + 1):
                return True
            cols.remove(c); diag1.remove(r - c); diag2.remove(r + c)
            solution[r] = -1
        backtracks += 1
        return False

```

```
    ok = dfs(0)
    if not ok:
        return None, backtracks
    return [c + 1 for c in solution], backtracks


def main():
    """
    This function handles user input. It then prints the "worst" permutation(s)
    for the given n as well as the amount of backtrack steps for the permutation,
    their inverse, and their complement.
    """
    n = int(input("Enter board size n: "))

    max_bt = -1
    worst_perms = []

    for pi in permutations(range(1, n + 1)):
        _, bt = nqueens_by_row_order(list(pi))
        if bt > max_bt:
            max_bt = bt
            worst_perms = [pi]
        elif bt == max_bt:
            worst_perms.append(pi)

    print(f"Max backtracks: {max_bt}")
    print(f"Number of worst-case permutations: {len(worst_perms)}\n")

    for idx, p in enumerate(worst_perms, 1):
        n_len = len(p)

        # Inverse
        p_inv = [0]*n_len
        for i, v in enumerate(p):
            p_inv[v-1] = i+1
        bt_inv = nqueens_by_row_order(list(p_inv))[1]

        # Complement
        p_comp = [n_len + 1 - v for v in p]
```

```
76          bt_comp = nqueens_by_row_order(list(p_comp))[1]
77
78          print(f"Worst permutation #{idx}: {p}, bt = {max_bt}")
79          print(f"   Inverse:  {p_inv}, bt = {bt_inv}")
80          print(f"   Complement: {p_comp}, bt = {bt_comp}\n")
81
82
83  if __name__ == "__main__":
84      main()
```

Figur 21: Pythonprogram som hittar de långsammaste permutationerna för ett givet n och printar sedan hur många backtracksteg den permutationen har samt gör detsamma för dess komplement och invers.

	permutation	backtracksteg
permutation	$[1, 4, 2, 3]$	7
invers	$[1, 3, 4, 2]$	4
komplement	$[4, 1, 3, 2]$	7
permutation	$[4, 1, 3, 2]$	7
invers	$[2, 4, 3, 1]$	0
komplement	$[1, 4, 2, 3]$	7

Figur 22: Långsammaste permutationerna och deras inverser samt komplement för $n = 4$.

	permutation	backtracksteg
permutation	$[1, 5, 2, 3, 4]$	4
invers	$[1, 3, 4, 5, 2]$	0
komplement	$[5, 1, 4, 3, 2]$	4
permutation	$[1, 5, 2, 4, 3]$	4
invers	$[1, 3, 5, 4, 2]$	0
komplement	$[5, 1, 4, 2, 3]$	4
permutation	$[2, 5, 4, 1, 3]$	4
invers	$[4, 1, 5, 3, 2]$	1
komplement	$[4, 1, 2, 5, 3]$	4
permutation	$[4, 1, 2, 5, 3]$	4
invers	$[2, 3, 5, 1, 4]$	0
komplement	$[2, 5, 4, 1, 3]$	4
permutation	$[5, 1, 4, 2, 3]$	4
invers	$[2, 4, 5, 3, 1]$	0
komplement	$[1, 5, 2, 4, 3]$	4
permutation	$[5, 1, 4, 3, 2]$	4
invers	$[2, 5, 4, 3, 1]$	1
komplement	$[1, 5, 2, 3, 4]$	4

Figur 23: Långsammaste permutationerna och deras inverser samt komplement för $n = 5$.

	permutation	backtracksteg
permutation	$[1, 3, 6, 4, 2, 5]$	70
invers	$[1, 5, 2, 4, 6, 3]$	42
komplement	$[6, 4, 1, 3, 5, 2]$	70
permutation	$[6, 4, 1, 3, 5, 2]$	70
invers	$[3, 6, 4, 2, 5, 1]$	6
komplement	$[1, 3, 6, 4, 2, 5]$	70

Figur 24: Långsammaste permutationerna och deras inverser samt komplement för $n = 6$.

	permutation	backtracksteg
permutation	$[1, 7, 5, 2, 6, 3, 4]$	43
invers	$[1, 4, 6, 7, 3, 5, 2]$	6
komplement	$[7, 1, 3, 6, 2, 5, 4]$	43
permutation	$[1, 7, 5, 2, 6, 4, 3]$	43
invers	$[1, 4, 7, 6, 3, 5, 2]$	0
komplement	$[7, 1, 3, 6, 2, 4, 5]$	43
permutation	$[7, 1, 3, 6, 2, 4, 5]$	43
invers	$[2, 5, 3, 6, 7, 4, 1]$	1
komplement	$[1, 7, 5, 2, 6, 4, 3]$	43
permutation	$[7, 1, 3, 6, 2, 5, 4]$	43
invers	$[2, 5, 3, 7, 6, 4, 1]$	0
komplement	$[1, 7, 5, 2, 6, 3, 4]$	43

Figur 25: Långsammaste permutationerna och deras inverser samt komplement för $n = 7$.

	permutation	backtracksteg
permutation	$[1, 2, 3, 8, 7, 5, 6, 4]$	196
invers	$[1, 2, 3, 8, 6, 7, 5, 4]$	189
komplement	$[8, 7, 6, 1, 2, 4, 3, 5]$	196
permutation	$[1, 2, 3, 8, 7, 6, 4, 5]$	196
invers	$[1, 2, 3, 7, 8, 6, 5, 4]$	189
komplement	$[8, 7, 6, 1, 2, 3, 5, 4]$	196
permutation	$[1, 2, 3, 8, 7, 6, 5, 4]$	196
invers	$[1, 2, 3, 8, 7, 6, 5, 4]$	196
komplement	$[8, 7, 6, 1, 2, 3, 4, 5]$	196
permutation	$[8, 7, 6, 1, 2, 3, 4, 5]$	196
invers	$[4, 5, 6, 7, 8, 3, 2, 1]$	3
komplement	$[1, 2, 3, 8, 7, 6, 5, 4]$	196
permutation	$[8, 7, 6, 1, 2, 3, 5, 4]$	196
invers	$[4, 5, 6, 8, 7, 3, 2, 1]$	10
komplement	$[1, 2, 3, 8, 7, 6, 4, 5]$	196
permutation	$[8, 7, 6, 1, 2, 4, 3, 5]$	196
invers	$[4, 5, 7, 6, 8, 3, 2, 1]$	3
komplement	$[1, 2, 3, 8, 7, 5, 6, 4]$	196

Figur 26: Långsammaste permutationerna och deras inverser samt komplement för $n = 8$.

	permutation	backtracksteg
permutation	$[1, 9, 3, 7, 2, 6, 8, 4, 5]$	401
invers	$[1, 5, 3, 8, 9, 6, 4, 7, 2]$	19
komplement	$[9, 1, 7, 3, 8, 4, 2, 6, 5]$	401
permutation	$[9, 1, 7, 3, 8, 4, 2, 6, 5]$	401
invers	$[2, 7, 4, 6, 9, 8, 3, 5, 1]$	37
komplement	$[1, 9, 3, 7, 2, 6, 8, 4, 5]$	401

Figur 27: Långsammaste permutationerna och deras inverser samt komplement för $n = 9$.

Källa till bevis 3.1 har lagts till: Bernhardsson, B. (1991). *Explicit solutions to the N-queens problem for all N.* Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Avsnitt 4.1 Gruppteori: Sista stycket "Mängden $M_{symmetrier}$ är ... Alltså $D_8 = \{e, r, r^2, r^3, s, sr, sr^2, sr^3\}$." har ersatts med: Mängden $M_{symmetrier}$ innehåller alltså precis de element som den dihedrala gruppen D_8 gör. Följer gör ett bevis för att $M_{symmetrier}$ är en grupp, specifikt gruppen D_8.

Avsnitt 4.1.1: Rubriken "Bevis av $M_{symmetrier} = D_8$" har ersatts med: Bevis att $M_{symmetrier}$ är en grupp

Avsnitt 4.2 Permutationer och gruppverkan Sista delen av sista meningen har orden "upp till" lagts till så att meningen blir: "alltså kan upp till sju ytterligare lösningar fås från en giltig lösning."

Avsnitt 7 Diskussion
Följande stycke har lagts till: Min teori kring vad den plötsliga minskningen för $n = 5$ och $n = 7$ bygger på att det har med att värdena är udda att göra på något sätt. Under mitt arbete med det här problemet har jag för det mesta inte sett några jämna värden med giltiga damplaceringar i hörnen men för $n = 5$ och $n = 7$ finns flertalet sådana. När vi pratade om de tre fallen som användes för att hitta en explicit lösning såg vi också att för udda n, placerades den sista damen i just ett hörn. Kanske beror minskningen i max backtracksteg och snittet på att fler hörnplaceringar är giltiga och därmed har större andel av permutationerna få backtracksteg än för de andra värdena på n. Å andra sidan syns inte samma plötsliga minskning i värde för $n = 9$ som också är ett udda tal, alltså kan det faktum att $n = 5, 7$ är udda inte vara hela förklaringen.

Avsnitt 6.3 Maximalt antal backtracksteg: bråkform har lagts till i tabellen i Figur 17.

Appendix Figur 20: Funktionalitet för att skriva ut snitten i bråkform har lagts till.

Referenser Har snyggat till referenserna lite.