Son Teslim tarihi: 10.01.2025

Ödev #4

OĞUZHAN TOPALOĞLU
Ç19052025 – Grup 1

Bilgisayar Mühendisliği Bölümü,
Elektrik-Elektronik Fakültesi,
Yıldız Teknik Üniversitesi

Istanbul, 2025

Video Linki: https://youtu.be/DKRlGfG6_t8

İÇERİK

1. Problemin Çözümü

Ödevde N-Queens problemini çözmek için dört farklı algoritma implementasyonu kodlanmıştır: Brute Force, Optimized_1, Optimized_2, ve Backtracking. Amaç, her algoritmanın çözüm üretme süresi ve verimliliğini karşılaştırmaktır. N-Queens problemi, N x N boyutundaki bir tahtada N adet vezirin yerleştirilmesi gereken bir problemdir. Her vezir bir satıra ve sütuna yerleştirilmelidir, aynı zamanda hiçbir vezir diğerini tehdit etmemelidir.

Algoritmaların Açıklamaları:

1- Brute Force

Tüm olası yerleştirme kombinasyonları (permütasyonlar) hesaplanır ve her bir kombinasyonun geçerliliği kontrol edilir. Bu basit ve doğrudan bir yaklaşımdır ancak çözüm bulma süresi çok uzun olabilir. Özellikle N büyüdükçe hesaplar dakikaları ve saatleri aşmaktadır. Bunun nedeni permütasyon sayısının faktöriyel bir şekilde büyümesidir.

2- Optimized_1

Bu yöntemde satır satır ilerlenerek her bir satıra bir vezir yerleştirilir. Her vezir yerleştirildikten sonra, önceki satırlarda yerleştirilen vezirlerle çatışma olup olmadığı kontrol edilir. Bu yöntemin geçerlilik kontrolü, yalnızca önceki satırlardaki yerleşimlere dayalıdır, böylece hesaplama maliyeti düşer.

3- Optimized_2

Bu yöntem Optimized_1'e benzer ancak bir ek optimizasyon içerir. Bu ek optimizasyon da satırların dışında hangi satırların dolu olduğunun bir dizide tutulmasıdır. Bu algoritmada önceden işaretlenmiş sütun bilgileri sayesinde gereksiz hesaplamalar engellenir. Optimized_1'e göre daha fazla bellek kullanılır ancak hızda gelişmeler elde edilir.

4- Backtracking

Vezir yerleştirilirken her adımda, geçerli yerleştirmenin geçerliliği kontrol edilir. Geçerli değilse, algoritma önceki adımlara geri dönerek yeni bir yerleşim dener. Bu çözümün avantajı çözüm bulunduğunda algoritma hemen sonlanmasıdır. Bunun sayesinde bir sürü gereksiz adımlar atlanır. Diğer yöntemlere göre daha karmaşıktır ancak çok daha verimli çalışır.

2. Karşılaşılan Sorunlar

Ödev sırasında en çok optimized_1 ve optimized_2 implementasyonunda sorun yaşadım. Bunun nedeni backtracking ve brute force için internette bir sürü kaynak olmasına rağmen optimized_1 ve optimized_2 için olmamasıydı.

Optimized_2'yi tamamladıktan sonra for döngülerinin sırasını karıştırdığım için sütunları kullanarak ek inceleme yapmamayı efektif bir şekilde kullanamıyordum. Bu yüzden de optimized_2'de optimized_1'den daha yavaş bir sonuç elde ediyordum. Bu hatamı düzeltmem bayağı zaman aldı.

Ayrıca print etme sürecinde de sıkıntılar yaşadım. Bir sürü function signature'ı değiştiğinden refactoring zaman aldı. Sonunda helper fonksiyonlar tanımlayarak refactoring'i bitirmeyi başardım.

3. Karmaşıklık Analizi

1- Brute Force Algoritması

Brute Force algoritması, problemi çözmek için tüm olası yerleşim permütasyonlarını hesaplar. Bu da tahta üzerindeki N * N hücrede her veziri yerleştirmeyi dener. Permütasyonların sayısı N! olduğundan, her hücre için N farklı yerleşim denemek gerekmektedir. Bu işlemin her biri N! kadar tekrarlandığı için zaman karmaşıklığı O(N!)'dir.

2- Optimized_1 Algoritması

Optimized_1 algoritması, her bir satırda yerleştirilecek veziri, daha önce yerleştirilen vezirlerin sütunlarına ve çaprazlarına bakarak geçerli yerleşimlerden seçer. Bu, yerleşimlerin sayısını azaltır ve Brute Force'tan daha verimli bir çözüm sunar. Ancak yine de her satırda geçerli yerleşimleri kontrol etmek için N * N kadar işlem yapılması gerektiği için zaman karmaşıklığı O(N^2)'dir.

3- Optimized_2 Algoritması

Optimized_2, Optimized_1 ile benzer şekilde çalışır ancak sütun kontrolü de işin içine eklenir. Bu ek kontrol sayesinde, average case'de Optimized_1'den daha hızlı çalışır. Ancak genel karmaşıklık hala O(N^2)'dir, çünkü her satırda yine N farklı kontrol yapılır ve bu işlem her satır için tekrarlanır.

4- Backtracking Algoritması

Backtracking algoritması, her bir satırda yerleştirilen vezirlerin sütunlarını ve çaprazlarını işaretler ve geçersiz yerleşimleri hızlıca dışlar. Bu sayede sadece geçerli çözümler üzerinde işlem yapılır. Her satırda geçerli yerleşimlerin kontrol edilmesi için O(N) işlem yapılır. Ancak her satır için bu işlem tekrarlanır, bu da teorik olarak O(N^2) karmaşıklığa yol açar. Bununla birlikte, geçersiz yerleşimler anında dışlandığı için büyük N'lerde pratikte daha hızlı çalışabilir ve karmaşıklık daha düşük seviyelere inebilir.

4. Ekran Görüntüleri

```
C:\Users\oguzh\Desktop\Kodlama\zSpecial\university-stuff\algoritma analizi\hw4>C19052025.exe

--------- Solving options ---------
0- Solve using BRUTE FORCE
1- Solve using OPTIMIZED_1
2- Solve using OPTIMIZED_2
3- Solve using BACKTRACKING
4- Solve using ALL ALGORITHMS
5- Solve using ALL ALGORITHMS excluding Brute Force
6- Terminate the program
Choose an option [0, 6]: 0
Enter N value: 4

Brute Force Solution #1:
. Q . .
. . . Q
Q . . .
. . Q .

Brute Force Solution #2:
. . Q .
Q . . .
. . . Q
. Q . .

BRUTE FORCE   => N: 4, solCount: 2, timeInSecs: 0.003
```

```
--------- Solving options ---------
0- Solve using BRUTE FORCE
1- Solve using OPTIMIZED_1
2- Solve using OPTIMIZED_2
3- Solve using BACKTRACKING
4- Solve using ALL ALGORITHMS
5- Solve using ALL ALGORITHMS excluding Brute Force
6- Terminate the program
Choose an option [0, 6]: 1
Enter N value: 5

Optimized_1 Solution #1:
Q . . . .
. . Q . .
. . . . Q
. Q . . .
. . . Q .

Optimized_1 Solution #2:
Q . . . .
. . . Q .
. Q . . .
. . . . Q
. . Q . .
```

```
Optimized_1 Solution #3:
. Q . . .
. . . Q .
Q . . . .
. . Q . .
. . . . Q

Optimized_1 Solution #4:
. Q . . .
. . . . Q
. . Q . .
Q . . . .
. . . Q .

Optimized_1 Solution #5:
. . Q . .
Q . . . .
. . . Q .
. Q . . .
. . . . Q

Optimized_1 Solution #6:
. . Q . .
. . . . Q
. Q . . .
. . . Q .
Q . . . .
```

```
Optimized_1 Solution #9:
. . . . Q
. Q . . .
. . . Q .
Q . . . .
. . Q . .

Optimized_1 Solution #10:
. . . . Q
. . Q . .
Q . . . .
. . . Q .
. Q . . .

OPTIMIZED_1   => N: 5, solCount: 10, timeInSecs: 0.014
```

```
// Aşağıdaki makro ile sonuçların yazılıp yazılmamasını belirtebilirsiniz
//   hiçbir sonuç yazılmasın istiyorsanız makroyu yoruma alın
// #define PRINT_BOARDS
```

```
--------- Solving options ---------
0- Solve using BRUTE FORCE
1- Solve using OPTIMIZED_1
2- Solve using OPTIMIZED_2
3- Solve using BACKTRACKING
4- Solve using ALL ALGORITHMS
5- Solve using ALL ALGORITHMS excluding Brute Force
6- Terminate the program
Choose an option [0, 6]: 4
Enter N value: 10
BRUTE FORCE   => N: 10, solCount: 724, timeInSecs: 0.103
OPTIMIZED_1   => N: 10, solCount: 724, timeInSecs: 0.006
OPTIMIZED_2   => N: 10, solCount: 724, timeInSecs: 0.004
BACKTRACKING  => N: 10, solCount: 724, timeInSecs: 0.003
```

```
--------- Solving options ---------
0- Solve using BRUTE FORCE
1- Solve using OPTIMIZED_1
2- Solve using OPTIMIZED_2
3- Solve using BACKTRACKING
4- Solve using ALL ALGORITHMS
5- Solve using ALL ALGORITHMS excluding Brute Force
6- Terminate the program
Choose an option [0, 6]: 5
Enter N value: 13
OPTIMIZED_1   => N: 13, solCount: 73712, timeInSecs: 1.064
OPTIMIZED_2   => N: 13, solCount: 73712, timeInSecs: 0.658
BACKTRACKING  => N: 13, solCount: 73712, timeInSecs: 0.398
```

```
--------- Solving options ---------
0- Solve using BRUTE FORCE
1- Solve using OPTIMIZED_1
2- Solve using OPTIMIZED_2
3- Solve using BACKTRACKING
4- Solve using ALL ALGORITHMS
5- Solve using ALL ALGORITHMS excluding Brute Force
6- Terminate the program
Choose an option [0, 6]: 5
Enter N value: 14
OPTIMIZED_1   => N: 14, solCount: 365596, timeInSecs: 6.840
OPTIMIZED_2   => N: 14, solCount: 365596, timeInSecs: 4.101
BACKTRACKING  => N: 14, solCount: 365596, timeInSecs: 2.390
```

```
--------- Solving options ---------
0- Solve using BRUTE FORCE
1- Solve using OPTIMIZED_1
2- Solve using OPTIMIZED_2
3- Solve using BACKTRACKING
4- Solve using ALL ALGORITHMS
5- Solve using ALL ALGORITHMS excluding Brute Force
6- Terminate the program
Choose an option [0, 6]: 5
Enter N value: 15
OPTIMIZED_1   => N: 15, solCount: 2279184, timeInSecs: 46.739
OPTIMIZED_2   => N: 15, solCount: 2279184, timeInSecs: 27.187
BACKTRACKING  => N: 15, solCount: 2279184, timeInSecs: 15.725
```

```
--------- Solving options ---------
0- Solve using BRUTE FORCE
1- Solve using OPTIMIZED_1
2- Solve using OPTIMIZED_2
3- Solve using BACKTRACKING
4- Solve using ALL ALGORITHMS
5- Solve using ALL ALGORITHMS excluding Brute Force
6- Terminate the program
Choose an option [0, 6]: 1
Enter N value: 4
OPTIMIZED_1   => N: 4, solCount: 2, timeInSecs: 0.000
```

```
--------- Solving options ---------
0- Solve using BRUTE FORCE
1- Solve using OPTIMIZED_1
2- Solve using OPTIMIZED_2
3- Solve using BACKTRACKING
4- Solve using ALL ALGORITHMS
5- Solve using ALL ALGORITHMS excluding Brute Force
6- Terminate the program
Choose an option [0, 6]: 5
Enter N value: 2
Invalid N value, enter 4 or above.
Enter N value: 4
OPTIMIZED_1   => N: 4, solCount: 2, timeInSecs: 0.000
OPTIMIZED_2   => N: 4, solCount: 2, timeInSecs: 0.000
BACKTRACKING  => N: 4, solCount: 2, timeInSecs: 0.000
```