BACKTRACKING

Öğrenci Adı: Melih Yelman
Öğrenci Numarası: 21011702
Ders Eğitmeni: Doç. Dr. Mehmet Amaç Güvensan
Video Linki: https://youtu.be/cmnDlb4hdvg

1- Problemin Çözümü:

Bu çalışmada, n-Queen problemi kapsamında verilen N sayısına göre, NxN'lik bir matriste birbirlerini alamayacak şekilde vezirlerin konumlandırılması sağlanmıştır. Problemin çözümünde Backtracking yaklaşımı temel alınarak dört farklı algoritmik mod geliştirilmiştir. İlk olarak, problemin kapsamı belirlenmiş ve vezirlerin birbirlerini satır, sütun ve çaprazlardan alamayacak şekilde yerleştirilmeleri gereksinimi netleştirilmiştir.

Çözüm sürecinde, programın verimliliğini artırmak amacıyla farklı optimizasyon stratejileri uygulanmıştır. İlk adım olarak, Brute Force yöntemi ile tüm olası kombinasyonlar sistematik bir şekilde denenmiş ve geçerli çözümler belirlenmiştir. İkinci adım olarak, Optimized_1 yaklaşımı benimsenerek her satırda yalnızca bir vezir bulunması kısıtlaması getirilmiştir. Bu sayede arama alanı daraltılarak gereksiz kombinasyonların değerlendirilmesi önlenmiş ve çözüm süresi önemli ölçüde azaltılmıştır. Üçüncü adım olarak, Optimized_2 yöntemi uygulanarak hem satır hem de sütun kısıtlamaları eklenmiştir. Son olarak, Backtracikng yöntemi kullanılarak vezirlerin yerleştirilmesi sırasında anlık olarak sütun ve çapraz doğrulamalar yapılmış, çakışma durumunda geri dönülerek alternatif çözümler denenmiştir.

2- Karşılaşılan Sorunlar:

Problemi çözmek için geliştirilen dört farklı algoritmanın programlanması sürecinde çeşitli zorluklarla karşılaştırılmıştır:

1. **Brute Force Yöntemi:**
    - **Rekürsif Yaklaşım:** Tüm olası hücre kombinasyonlarını denemek için kullanılan rekürsif yapının doğru şekilde yönetilmesi zor olmuştur.
    - **Çözüm Doğrulama:** Her kombinasyonun geçerli olup olmadığını kontrol etmek için gerekli kontrollerinin etkin ve hatasız bir şekilde kodlanması karmaşık olmuştur. Çapraz doğrulamaların doğru uygulanmaması, hatalı çözümlerin ortaya çıkmasına neden olmuştur.

2. **Optimized 1 Yöntemi:**
    - **Satır Kısıtlamasının Entegrasyonu:** Her satırda sadece bir vezir yerleştirme kısıtlamasını doğru bir şekilde uygulamak, algoritmanın mantığını kodlamak açısından zorluk yaratmıştır. Yanlış kodlama, bazı geçerli çözümlerin göz ardı edilmesine yol açmıştır.

3. **Optimized 2 Yöntemi:**
    - **Satır ve Sütun Kısıtlamasının Entegrasyonu:** Her satırda ve sütunda vezir olmasını sağlamak için permütasyon yaklaşımlı bir algoritma kurgulayabilmek için dizilerin doğru şekilde değiştirilmesi ve geri alınması sürecinde zorluk yaşanmıştır.

4. **Backtracking Yöntemi:**
    - **Çakışma Kontrolü Entegrasyonu:** Vezirlerin yerleştirilmesi sırasında sütun ve çapraz doğrulamalarını o an ki duruma göre gerçekleştirmek, algoritmanın genel performansını etkileyen bir faktör olmuştur. Bu kontrollerin optimize edilmemesi, çözüm süresinin beklenen uzun olmasına sebep olmuştur.

3- Karmaşıklık Analizi:

Problemi çözmek için geliştirilen dört farklı algoritmanın zaman karmaşıklıkları kaba hatlarıyla değerlendirilmiştir. Her bir algoritmanın temel işlem adımları ve bunların karmaşıklık dereceleri aşağıda incelenmiştir.

- **Brute Force Yöntemi**

  Brute Force Yöntemi, NxN matriste tüm olası hücre kombinasyonlarını denediği için en kötü durumda $O(2^{N^2})$ zaman karmaşıklığına sahiptir. Her hücre için vezir yerleştirip yerleştirilmeyeceği iki şekilde denendiğinden, kombinasyon sayısı üssel olarak artar.

- **Optimized 1 Yöntemi**

  Bu yöntem, her satırda sadece bir vezir yerleştirilmesi kısıtlanmasını uygulayarak arama alanını daraltır. Her satır için N sütun seçeneği olduğu için, zaman karmaşıklığı $O(N^N)$ olarak azalır.

- **Optimized 2 Yöntemi**

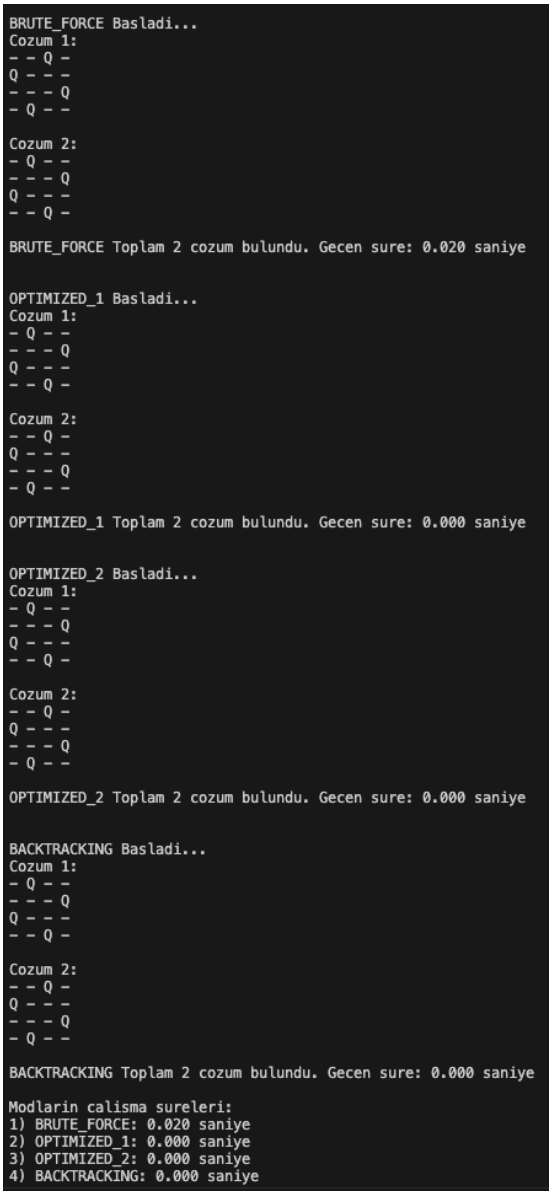
  Bu yöntem, hem satır hem de sütun kısıtlamalarını sağlayan bu yöntem, sütunları permütasyon üzerinden çözer. Permütasyon sayısı N! Olduğundan, zaman karmaşıklığı O(N!) olarak değerlendirilir.

- **Backtracking Yöntemi**

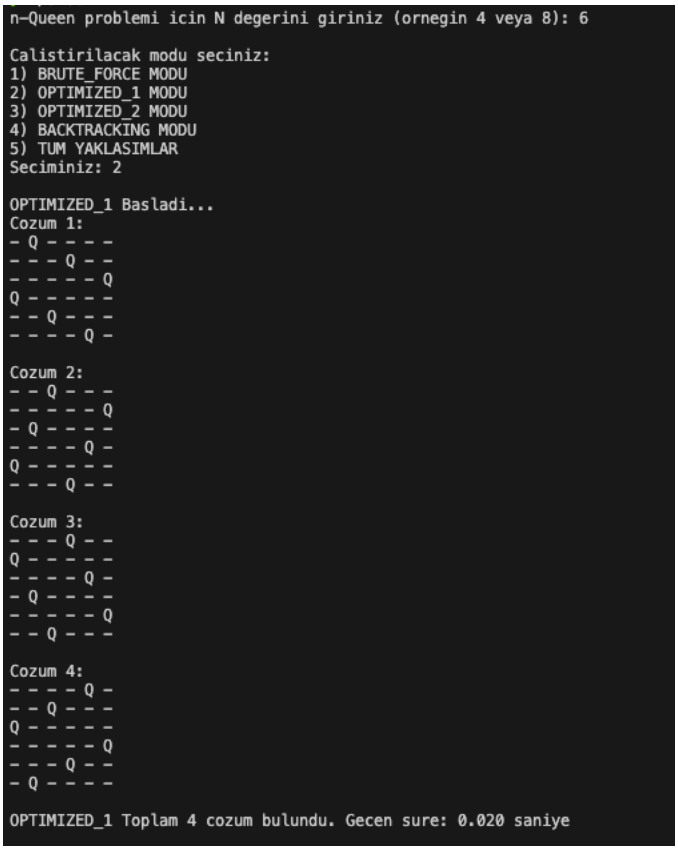
  Bu yöntem, satır satır ilerlerken, bir sonraki satıra vezir koymadan önce o sütunun veya çaprazlarının uygun olup olmadığına bakara bir eleme işlemi yapar, böylece gereksiz dallanmaları engeller. Teorik üst sınır olarak O(N!) diyebiliriz, çünkü en fazla N! Permütasyon var (Her satır için bir sütun) ancak pratikte, eleme işlemi sayesinde Optimized 2 yönteminden daha hızlı çalıştığı gözlenmiştir.

4- Ekran Görüntüleri

Ekran Görüntüsü 1:

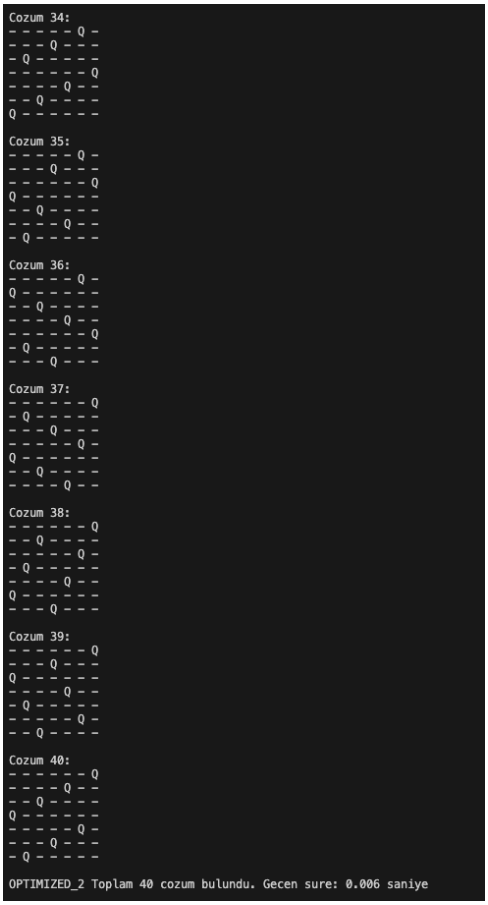


Ekran Görüntüsü 2:



```
n-Queen problemi icin N degerini giriniz (ornegin 4 veya 8): 6

Calistirilacak modu seciniz:
1) BRUTE_FORCE MODU
2) OPTIMIZED_1 MODU
3) OPTIMIZED_2 MODU
4) BACKTRACKING MODU
5) TUM YAKLASIMLAR
Seciminiz: 2

OPTIMIZED_1 Basladi...
Cozum 1:
- Q - - - -
- - - Q - -
- - - - - Q
Q - - - - -
- - Q - - -
- - - - Q -

Cozum 2:
- - Q - - -
- - - - - Q
- Q - - - -
- - - - Q -
Q - - - - -
- - - Q - -

Cozum 3:
- - - Q - -
Q - - - - -
- - - - Q -
- Q - - - -
- - - - - Q
- - Q - - -

Cozum 4:
- - - - Q -
- - Q - - -
Q - - - - -
- - - - - Q
- - - Q - -
- Q - - - -

OPTIMIZED_1 Toplam 4 cozum bulundu. Gecen sure: 0.020 saniye
```

Ekran Görüntüsü 3:



```
Cozum 34:
- - - - - Q -
- - - Q - - -
- Q - - - - -
- - - - - - Q
- - - - Q - -
- - Q - - - -
Q - - - - - -

Cozum 35:
- - - - - Q -
- - - Q - - -
- - - - - - Q
Q - - - - - -
- - Q - - - -
- - - - Q - -
- Q - - - - -

Cozum 36:
- - - - - Q -
Q - - - - - -
- - Q - - - -
- - - - Q - -
- - - - - - Q
- Q - - - - -
- - - Q - - -

Cozum 37:
- - - - - - Q
- Q - - - - -
- - - Q - - -
- - - - - Q -
Q - - - - - -
- - Q - - - -
- - - - Q - -

Cozum 38:
- - - - - - Q
- - Q - - - -
- - - - - Q -
- Q - - - - -
- - - - Q - -
Q - - - - - -
- - - Q - - -

Cozum 39:
- - - - - - Q
- - - Q - - -
Q - - - - - -
- - - - Q - -
- Q - - - - -
- - - - - Q -
- - Q - - - -

Cozum 40:
- - - - - - Q
- - - - Q - -
- - Q - - - -
Q - - - - - -
- - - - - Q -
- - - Q - - -
- Q - - - - -

OPTIMIZED_2 Toplam 40 cozum bulundu. Gecen sure: 0.006 saniye
```

Ekran Görüntüsü 4: