Raporlama ve Performans Analizi

**Projede Yaptıklarım**

1. **Veri Yapılarının Seçimi ve Uygulanması**
   * **Bağlı Liste (Linked List):**
     + Müşteri bilgilerini ve kargo verilerini tutmak için bağlı liste yapısını kullandım. Bu veri yapısını seçmemin nedeni, ekleme ve silme işlemlerinde düşük maliyetli (O(1)) işlemler yapmamı sağlamasıdır.

# Yığın (Stack):

* + - Her müşteri için gönderim geçmişini saklamak amacıyla yığın veri yapısını kullandım. Yığın, son eklenen gönderiye hızlı bir şekilde erişmemi sağladı.

# Öncelik Kuyruğu (Priority Queue):

* + - Teslim edilmemiş kargoları teslim sürelerine göre sıralamak için öncelik kuyruğu kullandım. Bu yapı sayesinde teslim edilmemiş kargoları öncelikli olarak işleme almayı başardım.

# Ağaç Yapısı (Tree):

* + - Şehirler arasındaki bağlantıları modellemek için ağaç veri yapısını tasarladım ve uyguladım. Bu yapı, alt şehirlerin kolayca eklenmesini ve şehirler arası derinlik hesaplamalarını mümkün kıldı.

# Algoritmaların Tasarımı ve Kullanımı

* + **Teslim Edilmiş Kargoların Sıralanması:**
    - Teslim edilmiş kargoları sıralamak için Bubble Sort algoritmasını kullandım. Zaman karmaşıklığı O(n^2) olmasına rağmen, az sayıda kargo verisiyle etkili çalıştı.

# Binary Search:

* + - Teslim edilmiş kargolar arasında hızlı bir şekilde arama yapmak için Binary Search algoritmasını entegre ettim. Bu algoritma, sıralı veriler üzerinde O(log n) zaman karmaşıklığıyla arama yapmamı sağladı.

# Merge Sort:

* + - Teslim edilmemiş kargoları teslim sürelerine göre sıralamak için Merge Sort algoritmasını kullandım. Bu algoritmanın zaman karmaşıklığı O(n log n) olduğu için büyük veri setlerinde bile verimli çalıştı.

# Derinlik Hesaplama (Depth Calculation):

* + - Ağaç yapısının derinliğini hesaplamak için bir rekürsif algoritma tasarladım. Bu algoritma, O(n) zaman karmaşıklığı ile çalışarak tüm alt şehirlerin derinliğini doğru bir şekilde belirledi.

# Performans Analizi

* **Zaman Karmaşıklığı:**
  + Bubble Sort yerine Merge Sort kullansaydım, teslim edilmiş kargoların sıralanması daha verimli olabilirdi.
  + Binary Search kullanımı, arama işlemlerinde zaman tasarrufu sağladı ve doğru bir seçim oldu.

# Bellek Kullanımı:

* + Dinamik veri yapıları (bağlı liste, yığın, ağaç) sayesinde bellek yönetimini verimli bir şekilde gerçekleştirdim.
  + Merge Sort algoritmasının ek bellek ihtiyacı olmasına rağmen, teslim edilmemiş kargoların sıralanmasında bu maliyeti tolere edebildim.

# Geliştirme Önerileri

1. Teslim edilmiş kargoların sıralanması için Bubble Sort yerine Merge Sort veya Quick Sort kullanılabilir.
2. Öncelik kuyruğu implementasyonunda Binary Heap yapısı tercih edilerek işlem maliyeti O(log n) seviyesine düşürülebilir.
3. Ağaç yapısında derinlik hesaplama için iteratif algoritmalar değerlendirilebilir.

# Sonuç

Bu projede kullandığım veri yapıları ve algoritmalar, kargo işlemlerinin düzenli ve hızlı bir şekilde gerçekleştirilmesini sağladı. Proje boyunca hem zaman hem de bellek kullanımını optimize etmeye odaklandım. Performans açısından geliştirilmesi gereken noktalar olsa da, mevcut çözüm proje gereksinimlerini başarıyla karşılamaktadır.

Zaman zaman ChatGPT'den de yararlandım, çünkü 3 kişilik bir projeyi 300'den fazla kod satırıyla kendi başıma yapmaya çalıştım. Elimden gelenin en iyisini yapmaya çalıştım. Gerisi takdirinize kalmış, Serpil Hocam.

Saygılar, [Sefa Taşdemir 23120205078]