Müşteri Verilerinin Yönetiminde Linked List Kullanımı

- Müşteri verilerini yönetmek için Linked List kullanımı, dinamik veri yapıları gereksiniminde, bellek verimliliği sağlamakta ve hızlı ekleme/silme işlemleri gerektiren senaryolarda önemli avantajlar sunar.
- Ancak, sabit erişim ve sıralama gereksinimleri göz önüne alındığında bazı durumlarda alternatif veri yapıları (örneğin, ArrayList veya HashMap) da değerlendirilebilir.

Kargo Önceliklendirme için Priority Queue Kullanımı

- Kargo önceliklendirme için Priority Queue kullanımı, kargoların önceliklerine göre sıralanmasını sağlar ve işlemlerin daha hızlı ve verimli bir şekilde yapılmasına olanak tanır.
- Kargo yönetim sistemlerinde sıklıkla karşılaşılan sıralama ve önceliklendirme ihtiyaçlarını karşılamak için ideal bir veri yapısıdır. Bu, özellikle acil teslimatlar veya öncelikli kargolar için etkili bir çözüm sunar.
- Alternatif olarak kargo önceliklendirme işlemi için Heap veri yapısı da kullanılabilir, çünkü heap, özellikle min-heap kullanılarak kargoların öncelik değerlerine göre hızlı bir şekilde sıralanmasını sağlar.

Kargo Rotalamada Tree Data Structure Kullanımı

- Kargo rotalarını yönetmek için ağaç veri yapısının kullanılması, teslimat noktalarını hiyerarşik bir şekilde temsil etmeyi sağlar. Ağaç yapısının kök düğümü, merkezi depoyu, alt düğümleri ise teslimat yapılacak şehirleri temsil eder.
- Bu yapı, teslimat rotalarını hızlı bir şekilde takip etmeye ve şehirler arası bağlantıları dinamik bir şekilde yönetmeye olanak tanır. Ayrıca, derinlik ve sıralama kriterlerine göre en kısa teslimat sürelerini hesaplama işlemleri de daha verimli hale gelir.

Gönderim Geçmişi Sorgulamada Stack Kullanımı

- Stack (yığın) veri yapısı, son giren ilk çıkar (LIFO) prensibine göre çalışır ve son gönderilen kargoları sorgulamak için idealdir.
- Bu yapı, kargoları ters sırayla ekleyip çıkarma işlemi yaparak, en son eklenen kargonun ilk sorgulanan olmasını sağlar. Her yeni gönderim yapıldığında, kargo yığına eklenir ve en son eklenen kargo, geçmiş sorgular sırasında ilk olarak gösterilir.
- Alternatif olarak, LinkedList gibi bir veri yapısı kullanarak, son 5 kargoyu saklamak ve listeyi sürekli güncellemek de kullanılabilir; böylece yığın yapısının avantajlarına sahip olunur.

Kargo Durum Sorgulamada Sort İşlemleri için Default Sort Methodu(timsort) ve Merge Sort Kullanımı

- Kargo durumlarının sıralanmasında, verilerin hızla işlenmesi için Java'nın varsayılan sıralama yöntemi olan Timsort kullanılmıştır. Timsort, sıralı verilerde daha hızlı çalışarak verimli bir sıralama sağlar.
- Bununla birlikte, Merge Sort da büyük ve karmaşık veri kümelerinde tutarlı performans sunarak, her durumda O(n log n) zaman karmaşıklığı ile sıralama yapar.

- Bu iki yöntem, farklı veri yapılarına göre sıralama ihtiyaçlarına göre tercih edilmiştir. Timsort genellikle daha hızlı ve verimli sonuç verirken, Merge Sort büyük veri setlerinde istikrarlı sonuçlar sağlar.
- Timsort ve Merge Sort algoritmalarının her ikisinin de space complexity değeri O(n)'dir. Yani her iki algoritma da sıralama işlemi sırasında ek bellek kullanırlar.

Ağaç Yapısı ile Teslimat Süresi Tespiti

- Derinlik kullanılarak kargoların teslimat sürelerinin lokasyonla ilişkili şekilde tespit edilmesi sağlanmıştır.
- Root node'u 1 gün baz alınıp, her akrabalık derecesinde 1 gün arttırma mantığı ile deliveryTime belirlenmiştir.
- Yani bu şekil bir algoritmada her kol için birer kargo aracının bulunduğu varsayılmıştır.

Kullanıcı ID Benzersiz Kılmak için HashMap Kullanımı

- HashMap, her kullanıcıyı benzersiz bir anahtar (ID) ile eşleştirir ve her anahtarın karşısına kullanıcı verilerini (değer) yerleştirir.
- Anahtarlar üzerinde hızlı arama, ekleme ve silme işlemleri sağlar, böylece kullanıcı ID'lerine hızlı erişim sağlanır.
- Aynı ID'yi birden fazla kullanıcıya atamak mümkün olmadığından, her kullanıcı ID'si HashMap içinde benzersiz olur.