Veri Yapıları Proje Ödevi

23120205052 – Hüseyin Melih ÖZBEK

23120205015 – Nezih Ahmet SUSMAZ

23120205020 – Sevban BOZASLAN

Algoritmaların ve Veri Yapılarının Seçimi

Bu kodda kullanılan algoritmalar ve veri yapıları, belirli ihtiyaçlara uygun şekilde seçilmiştir:

Veri Yapıları

1. **Bağlı Liste (Linked List):**
   Müşteri ve gönderi bilgilerini dinamik olarak saklamak için kullanılmıştır.
   a. **Neden Seçildi?**
      Dinamik bir yapıya sahip olması, yeni müşterilerin ve gönderilerin kolayca eklenmesini sağlar. Hafıza kullanımında esneklik sunar.
2. **Yığın (Stack):**
   Her müşteri için son 5 gönderinin takibini yapmak üzere kullanılmıştır.
   a. **Neden Seçildi?**
      Yığın yapısı LIFO (Last In, First Out) prensibine göre çalışır. Bu, son gönderilerin hızlı bir şekilde erişilmesi gereken durumlar için uygundur.
3. **Öncelikli Kuyruk (Priority Queue):**
   Gönderilerin teslimat sürelerine göre önceliklendirilmesi için kullanılmıştır.
   a. **Neden Seçildi?**
      Teslimat süresine göre sıralama ve işlemin önceliğini belirlemek için gereklidir.
4. **İkili Ağaç (Binary Tree):**
   Teslimat rotalarının modellenmesi için kullanılmıştır.
   a. **Neden Seçildi?**
      İkili ağaç yapısı, hiyerarşik veri temsilini kolaylaştırır ve rotaların dallanmasını etkin bir şekilde organize eder.

Algoritmalar

1. **Sıralama (Merge Sort):**
   Gönderilerin sıralanmasında kullanılmıştır.
   a. **Neden Seçildi?**
      Kararlı (stable) bir sıralama algoritmasıdır ve büyük veri setleri için uygundur.
2. **Binary Search (İkili Arama):**
   Teslim edilen gönderiler arasında belirli bir gönderiyi aramak için kullanılmıştır.
   a. **Neden Seçildi?**
      Önceden sıralanmış veri üzerinde hızlı arama yapar ve zaman karmaşıklığını O(log n) seviyesine indirir.
3. **Ağaç Derinliği Hesaplama:**
   Teslimat ağacında en kısa rotayı bulmak için kullanılmıştır.
   a. **Neden Seçildi?**
      Hiyerarşik veri yapısında derinlik hesaplaması, rota optimizasyonu için gereklidir.

Performans Analizi ve Sonuçları

Kodda Kullanılan Algoritmaların Zaman Karmaşıklığı

1. **Bağlı Liste Üzerinde Arama:**
   a. Ortalama Zaman Karmaşıklığı: O(n)
      Müşteri ve gönderi bilgileri arasında arama yapmak için kullanılır.
2. **Merge Sort:**
   a. Zaman Karmaşıklığı: O(n log n)
      Teslim edilmeyen gönderileri sıralamak için uygundur.
3. **Binary Search:**
   a. Zaman Karmaşıklığı: O(logn)
      Teslim edilen gönderiler arasında hızlı arama yapılmasını sağlar.
4. **Öncelikli Kuyruk Ekleme (Enqueue):**
   a. Zaman Karmaşıklığı: O(n)
      Yeni bir gönderi eklenirken teslimat sürelerine göre sıralama yapılır.

Bellek Kullanımı

I. **Bağlı Liste:** Dinamik olarak bellek tahsis ettiği için yalnızca ihtiyaç duyulan kadar alan kullanılır.

II. **Stack:** Her müşteri için maksimum 5 gönderi saklanır, bu nedenle bellek tüketimi sabittir. (O(1))

III. **Priority Queue ve Binary Tree:** Dinamik olarak büyüyen yapılar olduğundan, büyüklüklerine göre bellek tüketimi değişir.

Kullanıcı Rehberi

Programın Amaçları

Bu program, bir kargo yönetim sistemi olarak tasarlanmıştır. Müşteri bilgilerini, gönderim geçmişini ve teslimat rotalarını yönetir. Aynı zamanda gönderi sıralaması ve araması gibi işlemleri kolayca gerçekleştirir.

Menü Açıklamaları

1. **Yeni Müşteri Ekle:**
    a. Müşteri ID ve isim bilgisi girerek yeni müşteri ekleyin.
2. **Kargo Gönderimi Ekle:**
    a. Belirli bir müşteriye, gönderi ID, tarih, durum (teslim edildi veya edilmedi), ve teslimat süresi bilgisiyle yeni gönderi ekleyin.
3. **Teslim Edilen Kargoların Durumunu Sorgula:**
    a. Bir müşteri için belirli bir gönderiyi arayın. Eğer gönderi teslim edilmişse detaylarını görüntüleyin.
4. **Gönderim Geçmişini Görüntüle:**
    a. Bir müşterinin son 5 gönderisini görüntüleyin.
5. **Teslim Edilmeyen Kargoları Listele (Sıralı):**
    a. Teslim edilmeyen gönderileri teslimat sürelerine göre sıralı şekilde listeleyin.
6. **Teslimat Rotalarını Göster:**
    a. Teslimat rotalarını ağaç yapısı üzerinde görüntüleyin ve ağacın derinliğini hesaplayın.

Programı Kullanırken Dikkat Edilmesi Gerekenler

I. Müşteri eklenmeden önce gönderi eklenemez. Bu nedenle önce müşteri eklenmelidir.

II. Gönderilerin durumları doğru şekilde (Delivered/Not Delivered) girilmelidir.

III. Ağaç yapısı manuel olarak oluşturulmuştur. Teslimat rotası düzenlenirken bu yapıya dikkat edilmelidir.

Kullanıcı Rehberi

Yeni müşteri ekleme:

```
Müşteri ID: 3
Müşteri Adı: Serpil
```

Yeni bir gönderi ekleme (Teslim edilmiş):

```
Müşteri ID: 3
Kargo ID: 103
Tarih: 2024-12-14
Durum: Delivered
Teslimat Süresi: 2
```

Teslim edilmeyen kargoları listeleme:

```
Müşteri ID: 3
```

Yeni bir gönderi ekleme (Teslim edilmemiş):

```
Müşteri ID: 3
Kargo ID: 104
Tarih: 2024-12-15
Durum: Not Delivered
Teslimat Süresi: 7
```

Teslim edilmiş bir gönderiyi arama:

```
Müşteri ID: 3
Kargo ID: 103
```

Teslimat rotalarını görüntüleme:

```
Şehirler: İstanbul, Kocaeli, Bursa, Balıkesir, Manisa
```

Teslim edilmeyen kargoları sıralı listeleme:

Müşteri ID: 1

Belirli bir kargonun geçmişini görüntüleme:

```
Müşteri ID: 2
Kargo ID: 102
```

Son 5 gönderiyi görüntüleme (yeni müşteri için):

```
Müşteri ID: 3
```

Yeni müşteri ekleyip gönderi ekleme:

```
Müşteri ID: 4
Müşteri Adı: Ahmet
Kargo ID: 105
Tarih: 2024-12-17
Durum: Not Delivered
Teslimat Süresi: 6
```

Test Senaryosu

```
PRINTING Customers:
Customer ID: 2, Name: Serpil Yazicioglu
  Shipment ID: 204, Date: 2024-12-01, Status: Delivered, Delivery Time: 4
  Shipment ID: 201, Date: 2024-12-11, Status: Not Delivered, Delivery Time: 2
Customer ID: 1, Name: Ahmet Yilmaz
  Shipment ID: 106, Date: 2024-12-09, Status: Not Delivered, Delivery Time: 2
  Shipment ID: 105, Date: 2024-12-08, Status: Not Delivered, Delivery Time: 1
  Shipment ID: 104, Date: 2024-12-07, Status: Delivered, Delivery Time: 4
  Shipment ID: 102, Date: 2024-12-05, Status: Not Delivered, Delivery Time: 5
  Shipment ID: 101, Date: 2024-12-01, Status: Delivered, Delivery Time: 3

Sorting Undelivered Shipments using Merge Sort:
Sorted Not Delivered Shipments by Delivery Time:
Shipment ID: 105, Delivery Time: 1, Status: Not Delivered
Shipment ID: 106, Delivery Time: 2, Status: Not Delivered
Shipment ID: 102, Delivery Time: 5, Status: Not Delivered

Searching with Binary Search for Delivered Shipment:
Sorted Delivered Shipments:
Shipment ID: 101
Shipment ID: 104
Searching for Delivered Shipment ID: 104
Found Shipment: ID: 104, Date: 2024-12-07, Delivery Time: 4

PRINTING LAST 5 SHIPMENTS using Stack.
Last 5 shipments for Customer ID: 1:
  Shipment ID: 106, Date: 2024-12-09, Status: Not Delivered, Delivery Time: 2
  Shipment ID: 105, Date: 2024-12-08, Status: Not Delivered, Delivery Time: 1
  Shipment ID: 104, Date: 2024-12-07, Status: Delivered, Delivery Time: 4
  Shipment ID: 102, Date: 2024-12-05, Status: Not Delivered, Delivery Time: 5
  Shipment ID: 101, Date: 2024-12-01, Status: Delivered, Delivery Time: 3

PRINTING Tree:
Istanbul
|--Kocaeli
  |--Balikesir
  |--Manisa
|--Bursa
Shortest Delivery Time (Tree Depth): 2
PS C:\Users\msi-nb\Desktop\vscode>
```

Test Kodu

```c
int main() {
    addCustomer(1, "Ahmet Yilmaz");
    addShipment(1, 101, "2024-12-01", "Delivered", 3);
    addShipment(1, 102, "2024-12-05", "Not Delivered", 5);
    addShipment(1, 104, "2024-12-07", "Delivered", 4);
    addShipment(1, 105, "2024-12-08", "Not Delivered", 1);
    addShipment(1, 106, "2024-12-09", "Not Delivered", 2);
    addCustomer(2, "Serpil Yazicioglu");
    addShipment(2, 201, "2024-12-11", "Not Delivered", 2);
    addShipment(2, 204, "2024-12-01", "Delivered", 4);
    displayCustomers();

    printf("\nSorting Undelivered Shipments using Merge Sort:\n");
    sortUnDeliveredShipments(1);

    printf("\nSearching with Binary Search for Delivered Shipment:\n");
    searchDeliveredShipment(1, 104);


    enqueue(201, 2, "Processing");
    enqueue(202, 1, "In Transit");
    displayLast5Shipments(1);
```

```
    TreeNode *root = createCity(1, "Istanbul");
    root->left = createCity(2, "Kocaeli");
    root->right = createCity(3, "Bursa");
    root->left->left = createCity(4, "Balikesir");
    root->left->right = createCity(5, "Manisa");
    printf("\nPRINTING Tree:\n");
    printTree(root, 0);

    int shortestTime = calculateShortestDeliveryTime(root);
    printf("Shortest Delivery Time (Tree Depth): %d\n", shortestTime);

    return 0;
}
```