T.C

KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ PROGRAMI

ÖDEV KONUSU

Limandaki yük indirme-yükleme otomasyon sisteminin simülasyonunun Python kullanılarak yapılması

Hazırlayanlar

ZEYNEP GÜNEŞ

220501018

ESMANUR AYDIN

220501005

DERS SORUMLULARI

Öğr. Gör.NURBANU ALBAYRAK

Öğr. Gör.HÜSEYİN TARIK DURU

Arş.Gör. ERAY DURSUN

Arş.Gör.ŞEVVAL ŞOLPAN

13/12/2023

proje1/proje2 at main · Zeynepgunss/proje1 (github.com)

Yeniköy Mahallesi Ilıca Caddesi No:29, Başiskele,Kocaeli
info@kocaelisaglik.edu.tr kocaelisaglik.edu.tr

İÇİNDEKİLER

1. ÖZET

Bu ödevde, limandaki yük indirme-yükleme otomasyon sisteminin Python dilinde simülasyonu gerçekleştirilmiştir. Senaryonun temel işleyişinde, her bir zaman diliminde TIR'ların yüklerini indirme işlemi gerçekleştirilerek ardından gemilere yükleme yapılmaktadır. Bu süreçte, plaka numaralarına göre sıralama ve gemi numaralarına göre önceliklendirme gibi kurallar belirlenmiştir. Limandaki TIR ve gemi etkileşimlerini modellemek amacıyla 'Gemi' ve 'Tır' sınıfları oluşturulmuş, ayrıca yüklerin yönetimi için 'Stack' veri yapısı kullanılmıştır. Olayları ve gemi bilgilerini içeren CSV dosyalarından veri okuma işlemi gerçekleştirilmiş, ardından belirli kriterlere göre TIR ve gemi etkileşimleri simüle edilmiştir.

2. GİRİŞ

Projemizin ana hedefleri şunlardır:

TIR'lar ve gemiler arasındaki yük indirme-yükleme süreçlerini simüle etmek.

Senaryoya özgü sıralamalar ve kısıtlamalar altında liman operasyonlarını modellemek.

Limanın kapasite kontrolünü sağlamak ve belirli koşullar altında beklemeye geçmek.

Geliştirilen simülasyonun, gerçek dünya liman işlemlerini yansıtabilecek şekilde doğru ve güvenilir sonuçlar üretmesini sağlamak.

Bu proje, liman işlemlerinin otomasyonu ve verimliliği konularında teorik bilgilerin uygulamaya dönüştürülmesini hedeflemektedir. Ayrıca, Python programlama dilinin güçlü özelliklerini kullanarak modüler, okunabilir ve genişletilebilir bir simülasyon sistemi oluşturmak da projenin temel amaçları arasındadır._

3. YÖNTEM

Simülasyon, Python programlama dili kullanılarak obje tabanlı programlama prensipleri doğrultusunda tasarlanmıştır. Temel olarak, TIR ve gemi nesneleri oluşturulmuş ve bu nesneler arasında belirli kısıtlamalara ve sıralamalara uygun işlemler gerçekleştirilmiştir. Senaryoya özgü sıralama ve kısıtlamalar, program içinde kontrol yapıları ve algoritmalar aracılığıyla uygulanmıştır.

Vinç işlemleri, istif alanları, kapasite kontrolü ve liman operasyonlarının diğer yönleri, modüler fonksiyonlar ve sınıflar aracılığıyla yönetilmiştir.

Simülasyon, her bir zaman adımında gerçekleşen olayları takip eder ve belirli koşullar altında otomatik olarak ilerler. Limanın kapasite kontrolü, gemi sıralaması ve TIR plaka numarası sıralaması gibi temel algoritmalar, belirli senaryo gereksinimlerini karşılamak üzere özel olarak tasarlanmıştır.

KOD AÇIKLAMASI:

import csv:

Bu satır, csv kütüphanesini projeye dahil eder. Bu kütüphane CSV dosyalarını okumak ve yazmak için kullanılır.

class Gemi:

```
def __init__(self, geliş_zamanı, gemi_adı, kapasite, gidecek_ülke):
    self.geliş_zamanı = geliş_zamanı
    self.gemi_adı = gemi_adı
    self.kapasite = kapasite
    self.gidecek_ülke = gidecek_ülke
```

Bu satırlarda Gemi adında bir sınıf tanımlar. Gemi nesnelerinin özelliklerini içerir: geliş zamanı, gemi adı, kapasite ve gidecek ülke.

class Tır:

```
def __init__(self, geliş_zamanı, plaka, ülke, ton_20_adet, ton_30_adet, yuk_miktari, maliyet):
    self.geliş_zamanı = geliş_zamanı
    self.plaka = plaka
    self.ülke = ülke
    self.ton_20_adet = ton_20_adet
    self.ton_30_adet = ton_30_adet
    self.yuk_miktari = yuk_miktari
    self.maliyet = maliyet
```

Burda Tır adında bir sınıf tanımlar. Tır nesnelerinin özelliklerini içerir: geliş zamanı, plaka, ülke, tonaj bilgileri, yük miktarı ve maliyet.

class Stack:

```
def __init__(self):
    self.items = []
```

```
    def is_empty(self):
        return len(self.items) == 0

    def push(self, item):
        self.items.append(item)

    def pop(self):
        if not self.is_empty():
            return self.items.pop()

    def peek(self):
        if not self.is_empty():
            return self.items[-1]

    def size(self):
        return len(self.items)
```

Bu satırlar Stack adında bir sınıf tanımlar. Bu sınıf, bir yığın (stack) temel işlevlerini gerçekleştirir: boş olup olmadığını kontrol etme, eleman ekleyebilme (push), en üstteki elemanı çıkarabilme (pop), en üstteki elemana bakabilme (peek) ve yığının boyutunu öğrenebilme (size).

```
olaylar = []
with open('olaylar.csv', newline='') as file:
    okuyucu1 = csv.reader(file)
    for index, row in enumerate(okuyucu1):
        if index == 0:
            continue
        olaylar.append(row)
```

Bu satırlar 'olaylar.csv' dosyasını okur ve her bir satırı olaylar listesine ekler. İlk satır (başlık satırı) atlanır.

```
gemiler = []
with open('gemiler.csv', newline='') as file:
    okuyucu2 = csv.reader(file)
    for index, row in enumerate(okuyucu2):
        if index == 0:
            continue
        gemiler.append(row)
```

Bu satırlar 'gemiler.csv' dosyasını okur ve her bir satırı gemiler listesine ekler. İlk satır (başlık satırı) atlanır.

```
tir_zaman = {}
for veri in olaylar:
    zaman = int(veri[0])
    plaka = veri[1]
    ulke = veri[2]
    ton_20 = veri[3]
    ton_30 = veri[4]
    yuk_miktari = veri[5]
    maliyet = veri[6]
    if zaman in tir_zaman:
        tir_zaman[zaman].append(Tır(zaman, plaka, ulke, ton_20, ton_30, yuk_miktari, maliyet))
    else:
        tir_zaman[zaman] = [Tır(zaman, plaka, ulke, ton_20, ton_30, yuk_miktari, maliyet)]
```

Bu satırlarda, tırları ve geliş zamanlarını içeren bir sözlük oluşturur. Her bir zaman diliminde gelen tırları bu sözlüğün içinde tutar.

```
gemi_verisi = gemiler[0]
gemi = Gemi(int(gemi_verisi[0]), gemi_verisi[1], int(gemi_verisi[2]), gemi_verisi[3])
```

Burda gemi nesnesi oluşturur. Gemi bilgileri 'gemiler.csv' dosyasının ilk satırından alınır.

for veri in gemiler:

gemi_zaman = int(veri[0])

adı = veri[1]

kapasite = veri[2]

gidecek_ülke = veri[3]

print(f"\nGemi Bilgileri: Zaman={gemi_zaman}, Adı={adı}, Kapasite={kapasite}, Gidecek Ülke={gidecek_ülke}")

Burda "gemiler "listesindeki her bir gemi verisi için işlemleri gerçekleştirir. Gemi bilgilerini ekrana yazdırır.

 # Geminin doluluk durumu

gemi_dolu = False

 # Gemiye yüklenen toplam tonaj

gemi_toplam_tonaj = 0

 # Vinç sayısını kontrol etmek için değişken

vinc_islem_sayisi = 0

 # Yükleri gemiye yüklenme işlemi

istif_alani = Stack()

Bu satırlar, her bir gemi için bazı değişkenleri başlatır. Geminin doluluk durumunu kontrol etmek için gemi_dolu, toplam tonajı saklamak için gemi_toplam_tonaj, vinç işlem sayısını takip etmek için vinc_islem_sayisi, ve yükleri gemiye yüklemek için istif_alani adında bir Stack oluşturulur.

for zaman in sorted(tir_zaman.keys()):

tirlar_zamana_gore_sirali = sorted(tir_zaman[zaman], key=lambda x: x.plaka)

. Bu satırlar, tırları zamanlarına göre sıralar. tir_zaman sözlüğündeki tırları belirli bir zamana göre sıralar

for tır in tirlar_zamana_gore_sirali:

 # Gemi kapasitesini kontrol et

if int(gemi_toplam_tonaj) + int(tır.yuk_miktari) > int(kapasite):

print(f"Gemi kapasitesi aşıldı, Gemi bekletiliyor, Zaman: {zaman}")

gemi_dolu = True

break

Bu satırlar, gemi kapasitesini kontrol eder. Eğer gemiye yüklenecek olan tonaj, geminin kapasitesini aşıyorsa gemi bekletilir.

 # Yükleri istif alanına yükle

istif_alani.push(tır)

print(f"Yük istif alanına yüklendi, Zaman: {zaman}, Tır: {tır.plaka}, Yük Miktarı: {tır.yuk_miktari}")

gemi_toplam_tonaj += int(tır.yuk_miktari)

vinc_islem_sayisi += 1

 # İstif alanı doluysa veya tüm tırlar yüklenmişse döngüyü sonlandır

if gemi_dolu or vinc_islem_sayisi >= 20:

break

Bu satırlar, istif alanının dolu olması veya vinç işlem sayısının 20'yi aşması durumunda döngüyü sonlandırır.

while not istif_alani.is_empty():

yuk = istif_alani.pop()

print(f"Yük gemiye yüklendi, Zaman: {zaman}, Tır: {yuk.plaka}, Yük Miktarı: {yuk.yuk_miktari}")

Bu satırlar, istif alanındaki yükleri gemiye yükler.

print(f"Gemi İstatistikleri: Toplam Tonaj={gemi_toplam_tonaj}, Vinc İşlem Sayısı={vinc_islem_sayisi}")

Bu satır, ilgili gemi için istatistikleri ekrana yazdırır. Toplam tonaj ve vinç işlem sayısı görüntülenir.

4. SONUÇ VE ÖĞRENİLEN DERSLER

SONUÇ:

Geliştirilen liman simülasyon sistemi, belirli senaryo gereksinimlerini başarıyla yerine getirerek liman işlemlerinin otomasyonunu modellemektedir. TIR'lar ve gemiler arasındaki etkileşim, sıralamalar ve kapasite kontrolü başarılı bir şekilde yönetilmektedir.

ÖĞRENİLEN DERSLER:

Bu projenin geliştirilmesi sürecinde çeşitli öğrenim fırsatları ortaya çıkmıştır. İlk olarak, liman operasyonlarının karmaşıklığı ve optimize edilmesi gerekliliği konusundaki anlayış artmıştır. Nesne tabanlı programlama prensipleri ve Python'un bu tür projelerdeki etkili kullanımı konusundaki yetenekler geliştirilmiştir.

Ayrıca, senaryo gereksinimlerini anlamak ve bunları programlamada etkili bir şekilde uygulamak için algoritmik düşünce yetenekleri geliştirilmiştir. İstif alanları, gemi sıralamaları ve kapasite kontrolü gibi konularda karşılaşılan zorluklar, algoritmaların optimize edilmesi ve tasarımın geliştirilmesi için değerli deneyimler sağlamıştır.

Sonuç olarak, bu proje liman operasyonları simülasyonu alanında teorik bilgilerin pratik uygulamaya dönüştürülmesine yönelik değerli bir deneyim sunmuştur.

5. KAYNAKÇA

- https://chat.openai.com
- https://docs.python.org/tr/3/tutorial/datastructures.html
- https://www.obenseven.com.tr/yazilim/python/pythonda-dosyalar-ile-

calismak/csv-dosyalari-ile-calismak/

- https://python-istihza.yazbel.com/nesne_tabanli_programlama1.html
- https://python.sitesi.web.tr/python-class.html
- https://python-istihza.yazbel.com/sozluklerin_metotlari.html
- https://www.mobilhanem.com/python-dictionarysozluk-ve-metotlari/

GİTHUB: https://github.com/esm4ydn/python2/new/main?filename=.gitignore