Kocaeli Üniversitesi

Bilgisayar Mühendisliği Bölümü

Yazılım Laboratuvarı I Proje 2

Tunay Baştürk – Umut Kılıç

190201032@kocaeli.edu.tr-190201028@kocaeli.edu.tr

Projenin Özeti:

Bizden herhangi bir yazılım dilini kullanarak samuray sudokuyu threadlerle çözdürüp grafiğini çizdirmemiz ve çözüm adımlarını veritabanına veya txt'ye yazdırmamız istendi . Biz ise projeyi tamamlamak için pyhton dilini kullandık. Çözüm adımlarını yazdırmak için ise txt kullandık.

GİRİŞ:

Biz proje için python dilini kullandık.

Python programlama dili; 1991–1996 yılları arasında başında Amsterdam'da Guido Van rossum tarafından geliştirilmiş bir programlama dilidir. Az bilinen bir gerçek Python ismini Piton yılanından değil, geliştirici Rossum'un çok sevdiği MonthyPython adlı komedi grubunun sergilediği gösteriden almıştır.

Python, birçok dilin aksine derlemeye gerek kalmadan çalıştırılabilir. Object Oriented Programming(OOP)'i destekler ama Class açma zorunluluğu yoktur. Kolay öğrenilebilir, kolayca okunabilir bir programlama dilidir. Çapraz platform desteğine sahiptir, birçok farklı platformda kendisine yer bulmaktadır

YÖNTEM ve YAZILIM MİMARİSİ:

Her sudokuya ait matrisi oluşturduk. a sol üst matris , b sağ üst matris , c sol alt matris , d sağ alt matris , + orta matris olacak şekilde id leri atadık. Her matrisin karelerini Isimlendir fonksiyonunu çağırarak hepsine özgü kare adlandırılmaları yapıldı. Örneğin sol üst matrisin ilk karesi A1a şeklinde isimlendirildi. KesimleriAta() fonksiyonu ile beş matrisin ortak karelerinin adlandırılmalarını orta matrise göre değil de orta matrisin kesiştiği matrisin karelerine göre isimlendirilme yapıldı. Örneğin sol üst matrisle orta matrisle ilk kesişim karesine G7a adlandırılması yaparak kesişen karelere bu fonksiyonda atamalar yapıldı. Matris grupları ise o karenin bulunduğu sütunları satırları ve üçe üçlük hücredeki elemanları liste olarak tutar. TumKareler listemizde daha önce oluşturduğumuz matris karelerini toplayıp bu listemize aktardık. TumGruplar listemizde daha önce oluşturduğumuz matris gruplarını toplayıp bu listemize aktardık. Bulundugu Noktalar adında tüm o andaki karenin bulunduğu sütun satır ve üçe üçlük hücreleri tüm gruplar ve tüm karelerde gezerek bu oluşturduğumuz sözlüğe atadık. Komsular adında o andaki karenin komşularını tutan sözlük oluşturuldu.

```python
def Sudoku_ayristirma(grid, ekle):

    Ihtimaller = dict((s, Rakamlar) for s in TumKareler)
    for s, d in Sudoku_ihtimalleri(grid).items():
        if d in Rakamlar and not Ata(Ihtimaller, s, d):
            return False
    return Ihtimaller
```

Sudoku_ayristirma fonksiyonu bütün karelerin ihtimallerini 1,2,3,4,5,6,7,8,9 değerini atar.

```python
def SudokuParcalama(grid):
    global sol_ust, sag_ust, sol_alt, sag_alt, orta
    a = Sudoku([x[:9] for x in grid[:9]])
    b = Sudoku([x[9:18] for x in grid[:6]] + [x[12:21] for x in grid[6:9]])
    c = Sudoku([x[:9] for x in grid[12:]])
    d = Sudoku([x[12:21] for x in grid[12:15]] + [x[9:18] for x in grid[15:]])
    mid = Sudoku([x[6:15] for x in grid[6:9]] + [x[0:9] for x in grid[9:12]] + [x[6:15] for x in grid[12:15]])
    sol_ust = dict(zip(AmatrisKareleri, a))
    sag_ust = dict(zip(BmatrisKareleri, b))
    sol_alt = dict(zip(CmatrisKareleri, c))
    sag_alt = dict(zip(DmatrisKareleri, d))
    orta = dict(zip(OrtamatrisKareleri, mid))
```

SudokuParcalama fonksiyonumuzda sudokumuzu sol üst , sağ üst , sol alt , sağ alt , orta adında beş farklı sözlüğe ayrırız.

```python
def Sudoku_ihtimalleri(grid):

    a = Sudoku([x[:9] for x in grid[:9]])
    b = Sudoku([x[9:18] for x in grid[:6]] + [x[12:21] for x in grid[6:9]])
    c = Sudoku([x[:9] for x in grid[12:]])
    d = Sudoku([x[12:21] for x in grid[12:15]] + [x[9:18] for x in grid[15:]])
    mid = Sudoku([x[6:15] for x in grid[6:9]] + [x[0:9] for x in grid[9:12]] + [x[6:15] for x in grid[12:15]])
    chars = a + b + c + d + mid

    sqrs = AmatrisKareleri + BmatrisKareleri + CmatrisKareleri + DmatrisKareleri + OrtamatrisKareleri
    sol_ust = dict(zip(AmatrisKareleri, a))
    sag_ust = dict(zip(BmatrisKareleri, b))
    sol_alt = dict(zip(CmatrisKareleri, c))
    sag_alt = dict(zip(DmatrisKareleri, d))
    orta = dict(zip(OrtamatrisKareleri, mid))
    assert len(chars) == 405
    return dict(zip(sqrs, chars))
```

Sudoku_ihtimalleri fonksiyonu txt'den okuttuğumuz değerleri a,b,c,d,mid adında beş farklı listeye aktarılır. Bu beş farklı listeyi chars adındaki listeye ekleriz . Sqrs adında listemize daha önce bulduğumuz a matris kareleri , b matris kareleri , c matris kareleri , d matris karelerini , mid matris karelerini atarız ve bu iki listeyi sözlük şeklinde döndürürüz.

```python
def Ata(Ihtimaller, s, d):

    Diger_ihtimaller = Ihtimaller[s].replace(d, '')

    if all(Ihtimal_eleme(Ihtimaller, s, d2) for d2 in Diger_ihtimaller):

        return Ihtimaller
    else:

        return False
```

Ata fonksiyonu içine gönderilen ihtimalleri ihtimal ele fonksiyonu ile uygun olup olmadığını kontrol eder. Uygun ise bu ihtimali döner uygun değilse false döner .

```python
def Ihtimal_eleme(Ihtimaller, s, d):
    if d not in Ihtimaller[s]:
        return Ihtimaller
    Ihtimaller[s] = Ihtimaller[s].replace(d, '')

    if len(Ihtimaller[s]) == 0:
        return False
    elif len(Ihtimaller[s]) == 1:
        d2 = Ihtimaller[s]
        if not all(Ihtimal_eleme(Ihtimaller, s2, d2) for s2 in Komsular[s]):
            return False

    for u in BulunduguNoktalar[s]:
        dplaces = [s for s in u if d in Ihtimaller[s]]
        if len(dplaces) == 0:
            return False
        elif len(dplaces) == 1:
            if not Ata(Ihtimaller, dplaces[0], d):
                return False
    return Ihtimaller
```

Ihtimal_eleme fonksiyonu içine gönderilen ihtimaller dizisinin uzunluğuna göre kontrol yapar . Örneğin uzunluğu sıfır ise false , uzunluğu bir ise o andaki ihtimali yeni bir değişkinde tutar. Bu değere uygun yer olup olmadığını kontrol eder, eğer varsa ata fonksiyonumuz ile bu noktaya değer atanıp atanmadığını kontrol eder. Eğer atanmışsa false döner.Son olarak içine gönderdiğimiz ihtimalleri değiştirerek bu ihtimalleri döner.

```python
def Matris_kontrol(ihtimaller):

    if not ihtimaller:
        print("Çözum bulunamadı, çözümü bir daha kontrol edin.")
        return
    Matrise_DegerAtama(ihtimaller, AmatrisKareleri, matris1)
    Matrise_DegerAtama(ihtimaller, BmatrisKareleri, matris2)
    Matrise_DegerAtama(ihtimaller, CmatrisKareleri, matris4)
    Matrise_DegerAtama(ihtimaller, DmatrisKareleri, matris3)
    Matrise_DegerAtama(ihtimaller, OrtamatrisKareleri, matris5)
```

Matris_kontrol fonksiyonunda ilk başta ihtimalimizin olup olmadığını kontrol eder eğer ihtimalimiz varsa matrise_degeratama fonksiyonumuz çağırır.

```python
def Matrise_DegerAtama(Ihtimaller, sqr, matriss):
    for r in Satirlar:
        matriss.append(''.join(Ihtimaller[sqr[(ord(r) - 65) * 9 + int(c) - 1]] for c in Sutunlar))
```

Matrise_DegerAtama fonksiyonunda gönderilen matrise uygun değerler atanır.

```python
def Arama(Ihtimaller):
    if Ihtimaller is False:
        return False
    global zaman_ekleyici
    global zaman
    global kare_ekleyici
    global kare_listesi
    f = open("sonuç.txt", 'w')
    for s in TumKareler:
        if len(Ihtimaller[s]) == 1:
            if (s[2] == 'a'):
                f.write("Sol ust matriste cozulen degerin yeri : " + str(int(ord(s[0]) - 65)) + " X " + str(
                    int(s[1]) - 1) + " degeri = " + Ihtimaller[s] + "\n")
            elif (s[2] == 'b'):
                f.write("Sag ust matriste cozulen degerin yeri : " + str(int(ord(s[0]) - 65)) + " X " + str(
                    int(s[1]) - 1) + " degeri = " + Ihtimaller[s] + "\n")
            elif (s[2] == 'c'):
                f.write("Sol alt matriste cozulen degerin yeri : " + str(int(ord(s[0]) - 65)) + " X " + str(
                    int(s[1]) - 1) + " degeri = " + Ihtimaller[s] + "\n")
            elif (s[2] == 'd'):
                f.write("Sag alt matriste cozulen degerin yeri : " + str(int(ord(s[0]) - 65)) + " X " + str(
                    int(s[1]) - 1) + " degeri = " + Ihtimaller[s] + "\n")
            elif (s[2] == 'e'):
                f.write("Orta  matriste cozulen degerin yeri   : " + str(int(ord(s[0]) - 65)) + " X " + str(
                    int(s[1]) - 1) + " degeri = " + Ihtimaller[s] + "\n")

        zaman = time.process_time() - start1
        zaman_ekleyici += zaman
        zaman_listesi.append(zaman_ekleyici / 1000)
    f.close()
    if all(len(Ihtimaller[s]) == 1 for s in TumKareler):
        return Ihtimaller
    n, s = min((len(Ihtimaller[s]), s) for s in TumKareler if len(Ihtimaller[s]) > 1)
    return some(Arama(Ata(Ihtimaller, s, d))
                for d in Ihtimaller[s])
```

Arama fonksiyonunda o matrise gelebilecek değerleri dener ve o andaki matrisimize gelen değerler uzunluğu bire eşit ise matrisin kimliğine göre arama yaparak yeni oluşturduğumuz sonuç txt'sine o ihtimali atar ve bu atama süresince geçen zamanı zaman adındaki değişkenimize atarız ve bu her satırda bir değer bulduğu zaman yeni bir zaman değişkeni oluşturduğumuzdan bunların hepsini zaman listesi adlı listede tutarız.

```python
def Coz(grid):
    Matris_kontrol(Arama(Sudoku_ayristirma(grid, 1)))
    return Arama(Sudoku_ayristirma(grid, 1))
```

Coz fonksiyonunda threadler ile içine gönderdiğimiz matrisin çözümünü daha önce yukarıda anlattığımız arama , sudoku ayrıştırma fonksiyonlarımızı başlatırız.

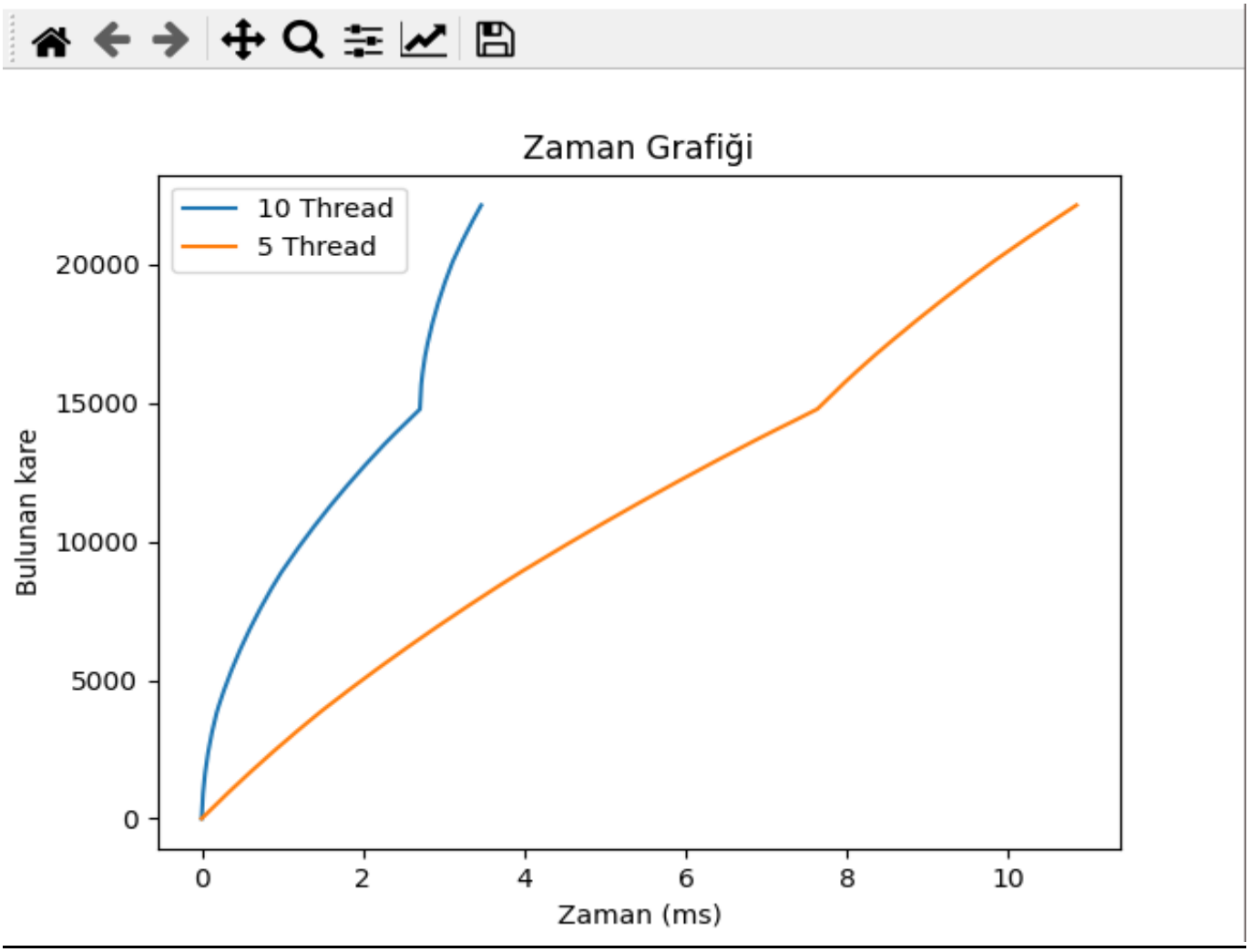
GUI için tkinter kütaphanesini kullanarak uygun noktaları verip label şeklinde ekranımıza kareler şeklinde samurat sudokuyu çizdirdik.

Thread sayacı adında global değişken tanımlandı. Bu değişkenimiz sıfıra eşit iken onlu threadi çöz fonksiyonuna gönderilerek çözüme başlatıldı. Her bir threadde geçen zaman hesaplanılıp toplam süreye eklenildi. Onlu thread çözümü bitirdikten sonra thread sayacımızın değeri bir arttırılıp beşli thread başlatıldı.Herbir thread çöz fonksiyonuna gönderildi. Herbir threadde geçen zaman hesaplanılıp toplam süreye eklenildi. En son olarak çözümler bitip süreler hesaplatıldıktan sonra grafikler bu zamanlara göre matplotlib.pyplot kütüphanesi kullanılarak çizdirildi.

DENEYSEL SONUÇLAR:

Samuray Sudoku

1	6	5	7	9	8	4	2	3				7	3	9	6	4	8	1	2	5
4	9	2	5	6	3	8	1	7				1	4	8	7	5	2	6	3	9
3	8	7	2	1	4	9	5	6				5	6	2	3	9	1	7	4	8
9	4	6	3	5	2	1	7	8				9	7	3	5	8	6	4	1	2
8	7	3	6	4	1	5	9	2				2	5	1	4	3	9	8	6	7
2	5	1	8	7	9	3	6	4				4	8	6	1	2	7	5	9	3
5	3	8	9	2	6	7	4	1	8	2	3	6	9	5	8	1	3	2	7	4
6	1	9	4	3	7	2	8	5	9	6	7	3	1	4	2	7	5	9	8	6
7	2	4	1	8	5	6	3	9	5	1	4	8	2	7	9	6	4	3	5	1
						3	9	2	4	7	6	5	8	1						
						8	7	6	3	5	1	2	4	9						
						5	1	4	2	9	8	7	3	6						
9	1	8	5	3	6	4	2	7	6	3	9	1	5	8	9	4	3	7	6	2
6	2	3	7	4	9	1	5	8	7	4	2	9	6	3	1	2	7	8	5	4
5	4	7	1	2	8	9	6	3	1	8	5	4	7	2	6	5	8	1	3	9
3	9	6	4	5	1	7	8	2				3	4	1	5	7	2	9	8	6
2	5	1	3	8	7	6	4	9				6	8	5	4	1	9	3	2	7
8	7	4	6	9	2	3	1	5				7	2	9	8	3	6	5	4	1
1	8	5	9	7	4	2	3	6				2	1	6	7	8	5	4	9	3
4	3	9	2	6	5	8	7	1				5	3	4	2	9	1	6	7	8
7	6	2	8	1	3	5	9	4				8	9	7	3	6	4	2	1	5




```
Sol alt matriste cozulen degerin yeri : 6 X 8 degeri = 6
Sag ust matriste cozulen degerin yeri : 4 X 7 degeri = 6
Sol ust matriste cozulen degerin yeri : 4 X 4 degeri = 4
Sag alt matriste cozulen degerin yeri : 7 X 2 degeri = 4
Sol alt matriste cozulen degerin yeri : 7 X 8 degeri = 1
Orta  matriste cozulen degerin yeri    : 3 X 1 degeri = 9
Sol ust matriste cozulen degerin yeri : 8 X 2 degeri = 4
Sag ust matriste cozulen degerin yeri : 1 X 2 degeri = 8
Sol ust matriste cozulen degerin yeri : 3 X 5 degeri = 2
Sag ust matriste cozulen degerin yeri : 6 X 8 degeri = 4
Sol ust matriste cozulen degerin yeri : 6 X 2 degeri = 8
Sag alt matriste cozulen degerin yeri : 4 X 4 degeri = 1
Sol ust matriste cozulen degerin yeri : 7 X 2 degeri = 9
Sag alt matriste cozulen degerin yeri : 7 X 8 degeri = 8
Sag ust matriste cozulen degerin yeri : 5 X 7 degeri = 9
Sag alt matriste cozulen degerin yeri : 8 X 0 degeri = 8
Sol ust matriste cozulen degerin yeri : 7 X 4 degeri = 3
Sag alt matriste cozulen degerin yeri : 3 X 6 degeri = 9
Sol alt matriste cozulen degerin yeri : 4 X 3 degeri = 3
Sag ust matriste cozulen degerin yeri : 5 X 6 degeri = 5
Sag alt matriste cozulen degerin yeri : 5 X 0 degeri = 7
Sol ust matriste cozulen degerin yeri : 8 X 5 degeri = 5
Sol alt matriste cozulen degerin yeri : 8 X 0 degeri = 7
Sol alt matriste cozulen degerin yeri : 0 X 2 degeri = 8
Sag ust matriste cozulen degerin yeri : 8 X 0 degeri = 8
Sag ust matriste cozulen degerin yeri : 0 X 6 degeri = 1
Sag ust matriste cozulen degerin yeri : 3 X 7 degeri = 1
Sol alt matriste cozulen degerin yeri : 6 X 0 degeri = 1
Sag alt matriste cozulen degerin yeri : 2 X 5 degeri = 8
Sag alt matriste cozulen degerin yeri : 4 X 6 degeri = 3
Sag alt matriste cozulen degerin yeri : 1 X 1 degeri = 6
Sol ust matriste cozulen degerin yeri : 1 X 1 degeri = 9
Sag ust matriste cozulen degerin yeri : 3 X 5 degeri = 6
Sol ust matriste cozulen degerin yeri : 5 X 6 degeri = 3
Sol ust matriste cozulen degerin yeri : 6 X 4 degeri = 2
```

AKIŞ ŞEMASI:

Akış şeması uzun olduğundan ektedir -> Akışşeması.html içerisindedir.

SONUÇ:

Bu proje sonucunda python dilini , algoritma yazmayı ,pythondaki gui kütüphanelerini(tkinter, pyqt5 , pygame) kullanarak tasarım yapmayı, threadleri kullanma mantığını, takım halinde görev dağılımı yapmayı , pes etmemeyi , bolca araştırma yapmayı ve en önemlisi sabırlı olmayı öğrendik .

KAYNAKÇA:

1- https://www.youtube.com/playlist?list=PLSmHiN0iazy_qX_6Tmecj4tTOefqh2-m2
2- https://www.youtube.com/watch?v=1aVi-Lp10ns&list=PLK6Whnd55IH7Q_6p_BVAsI108kcd9re-B&ab_channel=MertMekatronik
3- https://www.youtube.com/playlist?list=PLOl6SW8nLgJx9guRvfylVwrMXIginZhin
4- https://www.youtube.com/watch?v=G_UYXzGuqvM&t=522s&ab_channel=Computerphile
5- https://www.youtube.com/watch?v=IEEhzQoKtQU&t=1723s&ab_channel=CoreySchafer
6- https://www.youtube.com/watch?v=iW6_F77ut0w&ab_channel=Yaz%C4%B1l%C4%B1mBilimi
7- https://www.youtube.com/watch?v=EzHgbO1Cee4&list=PLWctyKyPphPiul3WbHkniANLqSheBVP3O
8- https://www.samurai-sudoku.com/
9- https://www.youtube.com/watch?v=tvvEqvyh_Vw&ab_channel=Yaz%C4%B1l%C4%B1mBilimi
10- https://www.youtube.com/watch?v=pPHHmkO9r3o&ab_channel=CodeCube