Samurai Sudoku

Buğrahan TOPAL 190201050, Bahadır IŞIK 190201049

1. PROBLEM TANIMI

Bizden istenen samurai sudakonun thread ile çözülmesidir. Burada ilk yaptığımız işlem samurai sudokuyu 5 ayrı sudokuya ayırmak ve bu sudokuları uygun şekilde çözmek. Buna aynı anda başlamak daha sonra ise sudokuyu 2 noktadan çözmek ve bunu 10 thread ile yapmak.

2. YAPILAN ARAŞTIRMALAR

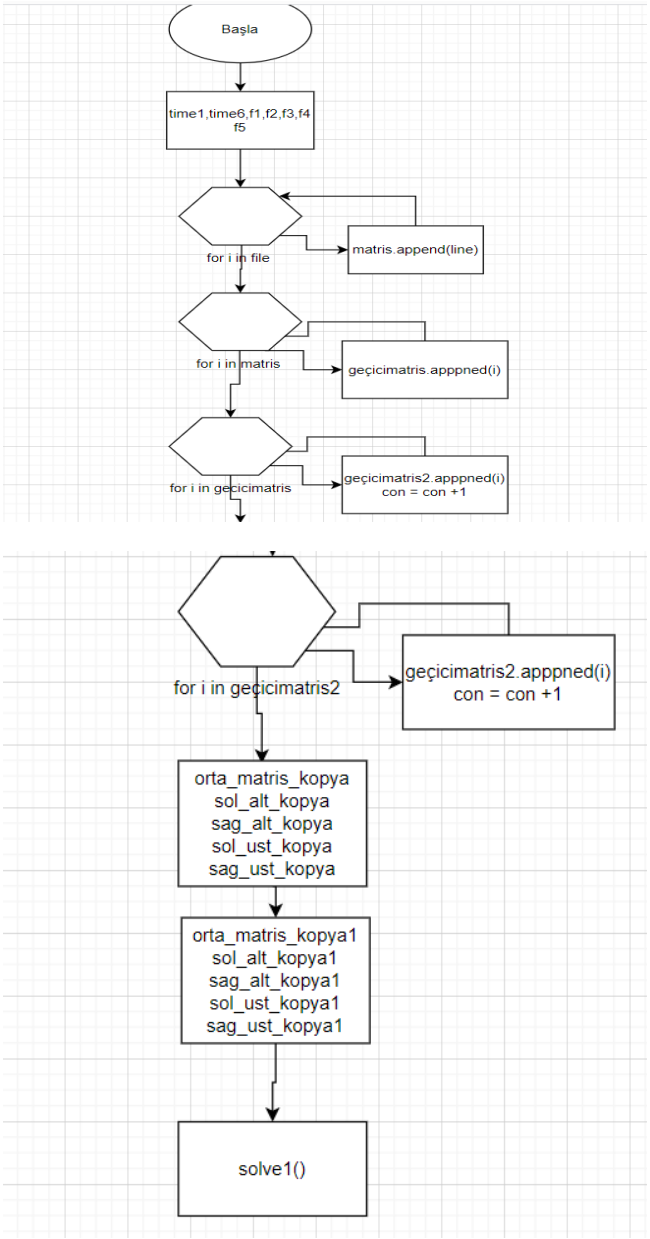
İlk sorunumuz txt ile verilen sudokuyu nasıl listeye atmamız gerektiğiydi. Bu sorunun çözümünü txt nin belli yerlerine , ekledik daha sonra bunu split fonksiyonu sayesinde istenilen biçimde ayırdık. Her bir sudoku için belirlediğimiz matrise atama yaptık.

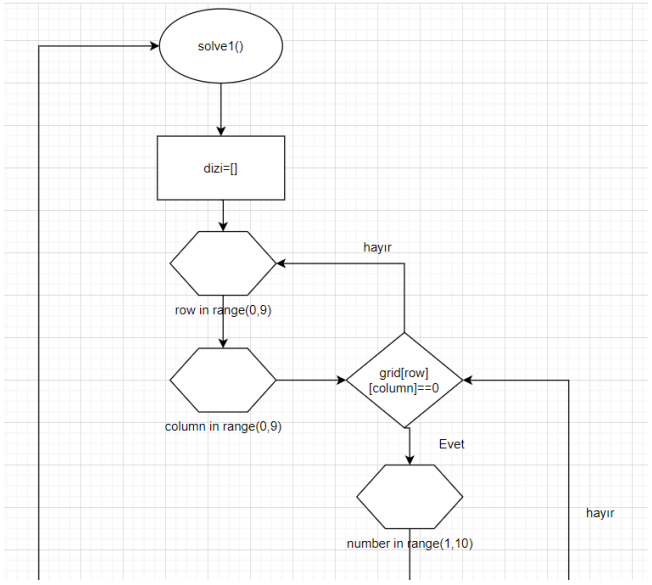
Sudokunun çözümü internetten yaptığımız araştırmalarda en çok kullanılan çözümün rekürsif olduğunu gördük ve bu durumu kendimize göre uyarladık kodumuzda bize gönderilen matrisin kaç tane çözüm yolu var ise hepsini bir liste içine attık böylece öngörülecek tüm çözümler elimizde olmuş oldu.

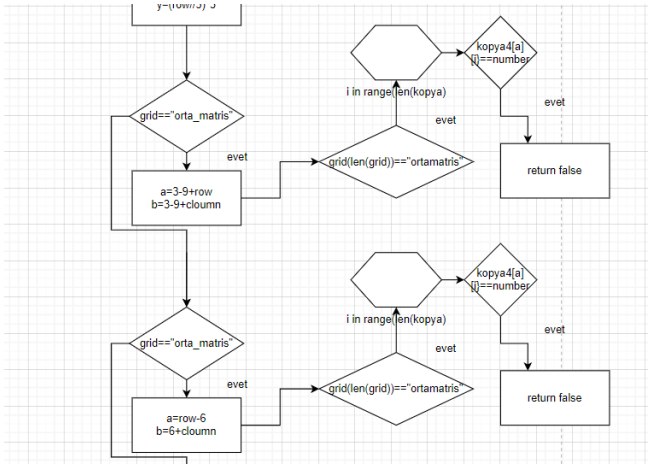
Zaman sıkıntısı sudokumuzun tüm çözümlerini aldık ama bazı durumlarda sudokunun çok fazla çözüm yolu olduğunu gördük bu çözümlerin hepsini yapması bazen 8 dakikayı bulabiliyordu bu zamanı azaltmak için olasılıkları azaltmalıydık bundan dolayı bize gönderilen sudokunun ortadaki sudoku ile kesiştiği yerlerde orta matriside kontrol ettik böylece ciddi bir zaman kazanmış olduk.

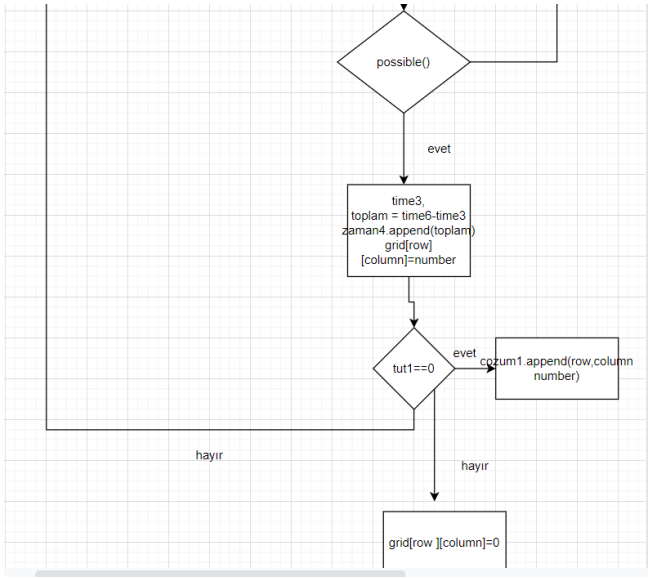
Tablo oluşturma işlemleri ile ilgili sorunumuzu time adlı kütüphane sayesinde çözdük.

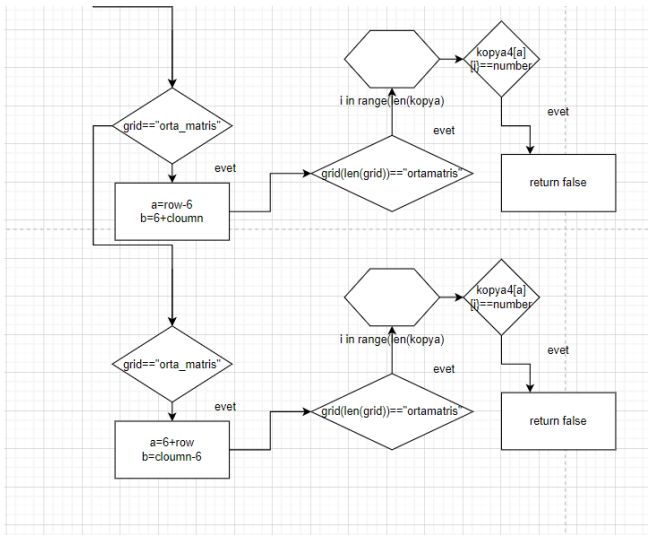
3. AKIŞ ŞEMASI

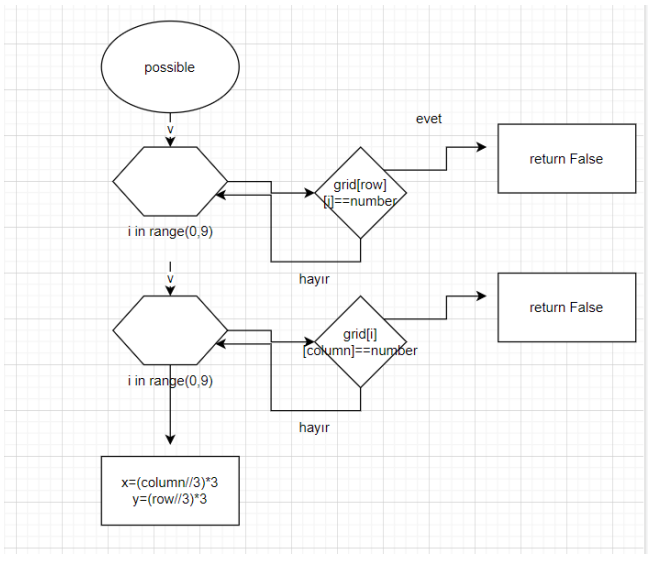


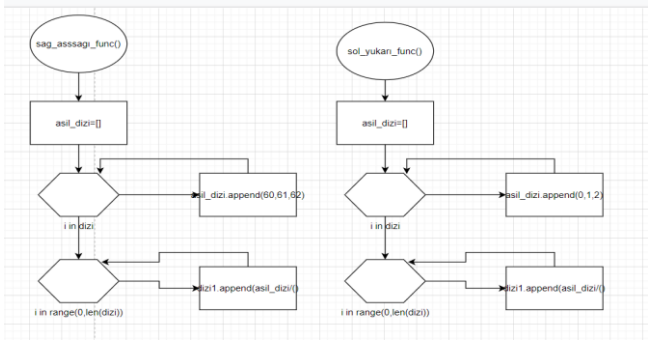

solve1()
dizi=[]
row in range(0,9)
hayır
column in range(0,9)
grid[row]
[column]==0
Evet
number in range(1,10)
hayır

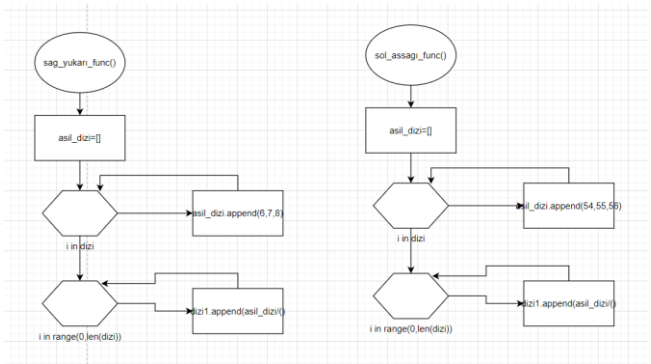

grid=="orta_matris"
evet
a=3-9+row
b=3-9+cloumn
grid(len(grid))=="ortamatris"
i in range(len(kopya)
kopya4[a]
[i]==number
return false
a=row-6
b=6+cloumn

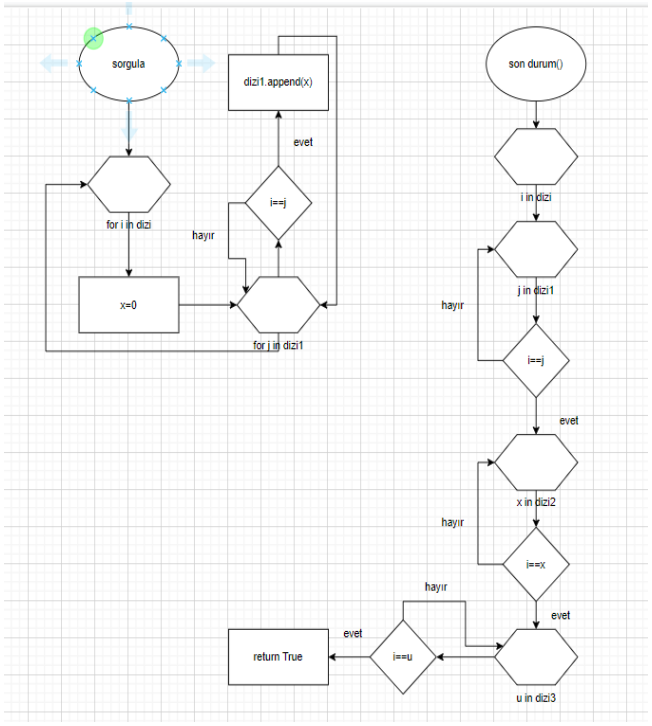

possible()
evet
time3,
toplam = time6-time3
zaman4.append(toplam)
grid[row]
[column]=number
tut1==0
cozum1.append(row,column
number)
hayır
grid[row][column]=0

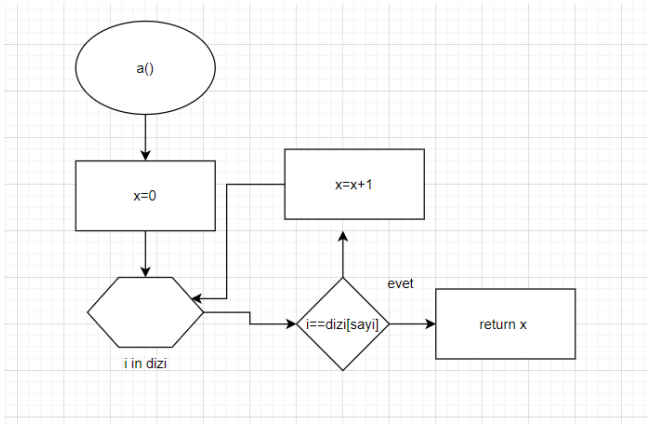

grid=="orta_matris"
evet
a=row-6
b=6+cloumn
grid(len(grid))=="ortamatris"
i in range(len(kopya)
kopya4[a]
[i]==number
return false
a=6+row
b=cloumn-6


possible
i in range(0,9)
evet
grid[row]
[j]==number
return False
hayır
i in range(0,9)
grid[i]
[column]==number
return False
hayır
x=(column//3)*3
y=(row//3)*3

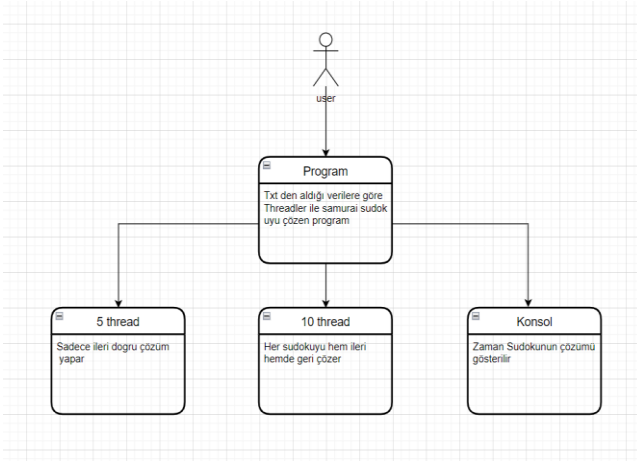

sag_asssagı_func()
asil_dizi=[]
i in dizi
asil_dizi.append(60,61,62)
dizi1.append(asil_dizi())
i in range(0,len(dizi))
sol_yukarı_func()
asil_dizi=[]
asil_dizi.append(0,1,2)


sag_yukarı_func()
asil_dizi=[]
asil_dizi.append(6,7,8)
i in dizi
dizi1.append(asil_dizi())
i in range(0,len(dizi))
sol_assagı_func()
asil_dizi=[]
asil_dizi.append(54,55,56)





4. YAZILIM MİMARİSİ



5. PROJENİN GENEL YAPISI

Projede ilk olarak txt üzerinden gelen samurai sudokuyu parçalara ayırıp her bir sudokuyu bir matris içine attık bu sayede istediğimiz zaman sudokuyu kullanabilecektik bu ayırma sırasında append ve split özelliklerini çok kullandık. Daha sonra sudoku çözücü fonksiyonu yazdık burada reqursif bir yapı izledik bu çözücü sayesinde atılan sudokunun tüm çözümlerini aldık sudokuyu çözerken hangi sayının nereye gelmesi gerektiğini anlamak için bir fonksiyon daha oluşturduk burada ise gelen satırın tüm sütün ve satırını taradık daha sonra kendi içini bunlar bize çok fazla olasılık tanıdığı için gönderilen matrisin en sonuna matrisin adını yazdık bu sayede hangi matrisin gönderildiğini anlamış olduk. Bu sayede sol ust matris geldi ise buraya uygun olan orta matriside kontrol etmiş olduk bu sayede çözüm olasıklarımız çok düşmüş oldu. Threadler ile çözümü yaptık daha sonra çözümün adımlarını txt içine yazdırdık.

Elimizde sadece tekli sudokuların çözümü olduğu için bir eleme sistemi geliştirdik bu sitemde çözüm yolları orta matrisi kestikleri yerler ile karşılaştırılıyor bu sayede samurai sudokunun nihai çözümüne ulaşıyoruz.

Sonda gösterilecek tablo için ise time modülünden yardım alıyoruz bu sayede 0.1 saniyede kaç işlem yapıldığını öğrenmiş oluyoruz.

9. REFERANSLAR

Web Sitesi

- https://medium.com/@robuno/python-split-metodu-6066ef795d0d
- https://www.w3schools.com/python/python_file_write.asp
- https://www.w3schools.com/python/python_ref_list.asp
- https://en.wikipedia.org/wiki/Sudoku_solving_algorithms
- https://dotnettutorials.net/lesson/inter-thread-communication-in-python/
- https://www.youtube.com/watch?v=GePqlA2KimU
- https://realpython.com/intro-to-python-threading/