YAZILIM LABORATUVARI I

PROJE II

SAMURAI SUDOKU ÇÖZÜMÜ

KAAN HIRÇIN, ALPER TALHA KÜÇÜK

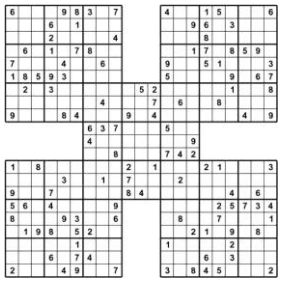
160202032 – 180202034
Bilgisayar Mühendisliği Bölümü (İÖ)
Kocaeli Üniversitesi

1. ProblemTanımı

Projede multithread yapısını kullanarak verilecek samurai sudoku üzerinden çözüm bulmamız istenmektedir, .txt uzantılı dosyada verilen başlangıç değerlerini dinamik olarak uygulamaya aktarmamız gerekmektedir. Biz bu uygulamayı Java dilinde yazdık.

Yapılan Araştırmalar

Samurai Sudoku nedir?, Samuray Sudoku (genellikle sadece Samurai veya Gattai-5 olarak anılır), beş standart Sudokunun kısmen örtüşen bir düzenlemesidir. Dört köşeli Sudoku, merkezi bir Sudoku çevresinde, dört köşeli Sudokunun her biri Merkez Sudoku ile bir blok paylaşacak şekilde düzenlenir. Bu, 41 blok üzerine dağıtılmış toplam 369 hücreli x şeklinde bir şekil oluşturur. Tekli Sudoku, özellikle de merkezi Sudoku'yu çözmek genellikle biraz daha kolaydır, çünkü iki Sudokunun paylaştığı bloklar daha kolay tamamlanabilir. Diğer varyantlar sekiz (Gattai-8), on üç (Gattai-13) veya daha fazla Sudokuyu birleştirir. Bu varyantlar aynı zamanda Canavar Samuray olarak da bilinir. Ayrıca farklı Sudoku çeşitlerini bir araya getiren Samuray Sudokuları da vardır. Bu beş Sudokunun her biri, birbirinden bağımsız olarak çözülebilir. Samuray Sudoku'ya başlamak için en iyi nokta, merkezi Sudoku'dur, çünkü merkezi Sudoku'nun dört köşe bloğu iki Sudoku'da bulunur. Bu nedenle bu blokları daha kolay tamamlayabilirsiniz. Aksi takdirde normal bir Sudoku'ya benzer şekilde çözülür.



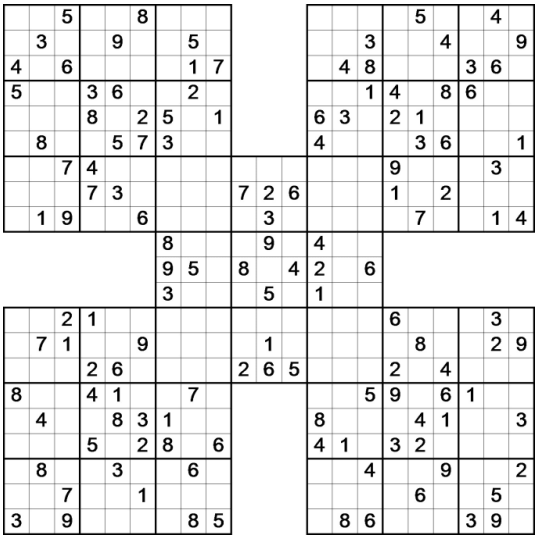
Java dilinde dosya işlemleri, Java'da disk üzerindeki dosyalar ve klasörlerle iletişim sağlamak için File sınıfından faydalanırız. File sınıfı disk üzerinde belirtilen konumdaki bir dosya ya da klasörü kod içerisinden tanımlamak için kullanılır. Aşağıdaki kod parçasına bakarsak File sınıfının yapıcısı (Constructor) içerisinde dosyanın adı **dosya.txt** olarak belirtilmiş. Bu şekilde disk üzerinde **dosya.txt** adında bir dosyaya işlem yapmak için o dosyayı bir değişkene atamış oluyoruz. **exists** metodu dosyanın disk üzerinde önceden var olduğunu kontrol etmek için kullanılıyor. Eğer dosya diskte henüz yoksa **createNewFile** ile diskte **dosya.txt** adında bir dosya oluşturuyoruz. Dosyaya metin yazma işlemini başlatmak için FileWriter sınıfından faydalanırız. FileWriter sınıfı File tipindeki bir değişkeni yazma amacıyla kullanmaya yarar. Yapıcı içerisinde yer alan boolean tipindeki değer dosyanın **append** modunda yazılmasını sağlar. **append** modu dosyanın içerisinde yer alan mevcut metinlere dokunmadan dosyanın son karakterinden itibaren yazma işlemini başlatacaktır. Yukarıdaki kod içerisinde bu değer **false** olduğundan **dosya.txt** her seferinde baştan yazılacaktır.

```
String str = "Bunu dosyaya yazdir";

File file = new File("dosya.txt");
if (!file.exists()) {
    file.createNewFile();
}

FileWriter fileWriter = new FileWriter(file, false);
BufferedWriter bWriter = new BufferedWriter(fileWriter);
bWriter.write(str);
bWriter.close();
```

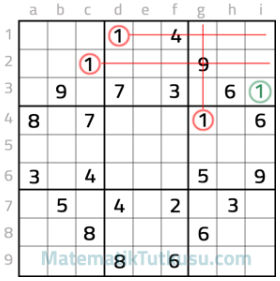
Javada Thread Kullanımı ve Runnable Arayüzü
Her bir işlemin altında çalışan alt işlem parçacıklarına thread adı verilmektedir. Threadler aynı anda birden fazla işlem yapmayı sağlayan yapılardır. Thread kullanımı için Thread sınıfını kalıtım almak(extends) veya Runnable arayüzünü uygulamak(implements) olmak üzere iki yöntem kullanılır. Thread sınıfının kullanımı için Thread sınıfı kalıtım alındıktan sonra run metodu override edilir ve gerekli komutlar yazılır. **Runnable arayüzünün** kullanımı için arayüz metotları sınıf tarafından uygulanır. Arayüz uygulandıktan sonra Thread sınıfının kurucusuna sınıf parametre

olarak geçilerek thread çalıştırılır. **Thread Yönetimi** Birden fazla thread aynı anda tek bir kaynağı kullanmaya çalıştığında belirsiz durum ortaya çıkar. Bu belirsizliğin önüne geçmek için thread'ler arasında senkronizasyon iyi bir şekilde yapılması gerekir. Senkronizasyon için threadler belirli bir süre durdurulabilir. Bu yöntem senkronizasyonu sağlarken thread yapısının sağladığı avantajı ortadan kaldırır. Diğer bir yöntem ise synchronized anahtar kelimesini kullanmaktır. Thread'lere öncelik vermek diğer çözüm yöntemlerinden birisidir. Öncelik verme yöntemi thread'lerin senkron çalışmasını garanti etmez. Java ile thread kullanımı bir sınıf kullanımından farksızdır. Ancak thread senkronizasyonunun yapılan işlem türüne göre iyi hazırlanması gerekir. Senkronizasyon iyi yapılmadığında beklenmedik sonuçlar ortaya çıkar ve yeterli verim alınmayabilir. Thread yönteminin verimli bir şekilde kullanılması için ayrıca problemin-algoritmanın parçalara ayrılabiliyor olması gerekir.

2. Tasarım Akışşeması



Yukarıdaki Samurai Sudoku örneği, 5 normal sudokuya indirgenerek çözüme başlanır.



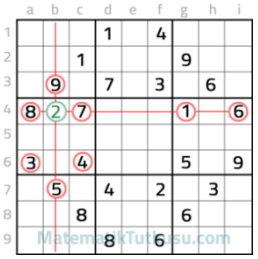
Sıkıştırma Tekniği, Her bölgede (her satırda veya her sütunda) herhangi bir rakamdan sadece bir tane bulunmalıdır. Sudoku çözülürken en çok kullanılan yöntem, bölgelerdeki (satırlardaki veya sütunlardaki) eksik olan rakamları taramaktır. Hangi bölgelerde hangi rakamların eksik olduğuna bakılır (bunu yaparken tüm sudoku içindeki en fazla adette bulunan rakamdan başlayınız). Bu eylem yapılırken bölgede eksik olan bir rakam, yan veya alt/üst bölgelerdeki aynı rakamlar ile yatay/dikey taramalar yapılarak o bölgede bulunamayacağı yerler elenir; bulunma ihtimali olan tek yer bulunmaya çalışılır.

Resimdeki örnekte, birinci ve ikinci bölgede 1 rakamı mevcuttur. Üçüncü bölgede 1 rakamı yoktur. Üçüncü bölgede 1 rakamının yerini bulmayı düşündüğümüzde, birinci ve ikinci bölgelerde 1'lerin bulunduğu satırlarda (1. ve 2. satır) bir daha 1 rakamı olamayacağı için üçüncü bölgede kırmızı ile çizilmiş satırlara 1 rakamı gelemez.

Altıncı bölgedeki 1 rakamının bulunduğu yerden dolayı üçüncü bölgedeki **g** sütununa 1 rakamı gelemez.

O zaman üçüncü bölgede 1 rakamının gelebileceği sadece bir yer kalmıştır.

Üçüncü bölgedeki 1 rakamının yeri **i3**'te olmalıdır.



Satır-sütun kesişmesinde tek ihtimal ,Sudoku içinde herhangi bir satır ile bir sütunun kesişimindeki rakam olmayan yerleri gözden geçirdiğimizde, kesişimdeki yere gelecek rakamı bulma ihtimali olabilir (Kesiştireceğiniz yer seçiminde, satır ve sütunda çok sayı bulunan yerleri seçiniz).

b sütunu ve 4. satırın kesişiminde yerde (b4) rakam yoktur. Bu rakamı bulmak için şöyle bir yol izlenir.

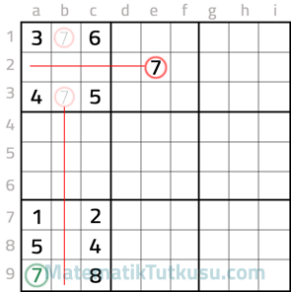
4. satırda **8**, **7**, **1** ve **6** rakamları olduğundan b4'e bu rakamlardan birisi gelemez.

b sütununda **9** ve **5** rakamları olduğundan b4'e bu rakamlardan birisi gelemez.

Dördüncü bölgede **8**, **7**, **4** ve **3** rakamları olduğundan b4'e bu rakamlardan birisi gelemez.

O zaman b4'e gelemeyecek rakamları listesi gözden geçirdiğimizde; **1, 3, 4, 5, 6, 7, 8, 9** rakamlarıdır.

O zaman **b4**'e gelebilecek tek rakam kalmıştır. O da **2** rakamıdır.



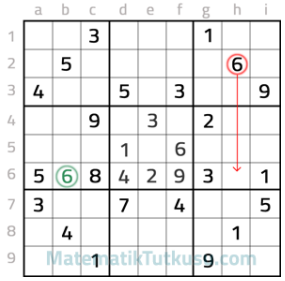
Sudoku satır-sütun etkileşimleri, Bazı sudoku sorularında satırlardaki rakamlar ile sütunlardaki rakamlar arasındaki özel durumlardan dolayı bir bölgede bir rakamın yerini tam olarak bulamasak bile o bölgede gelebileceği yerleri tahmin etmek kolaydır.

Bu teknikte, bir bölgede bir rakamın gelebileceği yerler (bir, iki veya üç yere gelebilme durumlarında) bir satır boyunca veya bir sütun boyunca olduğunda, o satırın veya sütunun diğer bölgelerinde bir daha o rakam bulunamaz.

Resimdeki kısmi sudokunun ikinci bölgesinde 7 rakamının bulunduğu satırda (2. satır) bir daha 7 rakamı bulunmayacağına göre birinci bölgede **a2**, **b2** ve **c2** karelerine 7 rakamı gelemez. Bu yüzden birinci bölgede 7 rakamının gelebileceği iki yer vardır. Bu yerler **b1** ve **b3**'tür.

b sütununu düşündüğümüzde birinci bölgede **b1** veya **b3** karelerinin birisinde mutlaka 7 rakamı bulunması gerekeceğinden (hangisinde olduğunu kesin olarak bilmesekte) yedinci bölgede b sütununa 7 rakamı gelemez. O zaman yedinci bölgede 7 rakamının gelebileceği sadece bir yer kalmaktadır.

Yedinci bölgedeki 7 rakamının yeri **a9**'da olmalıdır.



Satır veya sütundaki eksik rakamlar, Bir satırda (sütunda) bir çok rakam bulunmuş ve çok azı bulunmamışsa bu yöntem uygulanır.

Resimdeki altıncı satırda sadece iki rakam belli değildir. 1,2,3,4,5,8 ve 9 rakamları bilinmektedir. 6 ve 7 rakamları çıkmamıştır.

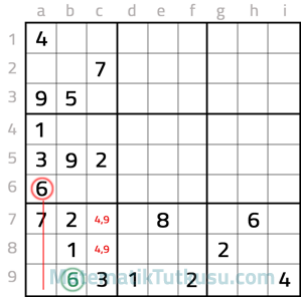
Kalan boş kareler **b6** ve **h6**'dır.

h6'nın bulunduğu h sütununda bir tane 6 rakamı bulunmaktadır (h2'de).

Bu 6 rakamı h sütununda bulunduğu için h6'ya 6 rakamı gelemeyecektir.

O zaman altıncı satırda eksik olan 6 rakamının gelebileceği tek yer vardır, o da **b6** karesidir. Ve yine bu satırda eksik kalan 7 rakamının gelebileceği bir tek yer vardır; h6'dır.

Bu yöntem "Satır-sütun kesişmesinde tek ihtimal" yöntemi ile benzerlik göstermektedir aslında.



İhtimalleri yazmak ve 'İkililer'den yararlanmak, Zor sudoku soruları klasik çözüm yöntemleri ile çözülemezler. Bu yüzden İhtimal yazma yöntemi, tıkanan bir sudokuyu açacaktır.

Bazen bir bölgede verilmemiş her boşluğa gelebilecek tüm rakam ihtimalleri, bazen bir satırda verilmemiş her boşluğa gelebilecek tüm rakam ihtimalleri, bazen de bir sütunda verilmemiş her boşluğa gelebilecek tüm rakam ihtimalleri boşluklar içine küçük rakamlar halinde yazılır. Bazen de burada olduğu gibi tüm sudokuda rakam bulunamamış her boşluk için ihtimallerin yazılması gerekebilir.

Bu ihtimallerin yazılmasından yararlanarak diğer boşluklara gelebilecek rakamlar bulunabilir. Bazen de tüm boşluklar içindeki ihtimallerin birbirleri ile etkileşimi sonucu bazı ihtimallerin elenmesi ve bir boşluğa gelebilecek tek bir rakamı bulma ihtimali olabilir.

İkililerden yararlanmak

İhtimaller yazılırken, resimdeki kısmi sudoku örneğinde olduğu gibi bazen de bir bölgedeki eksik olan bir-iki rakamın (yedinci bölgede 4 ve 9 rakamı) bölge içinde gelebilme ihtimali olan yerlere yazılması yeterli olur ki; o bölgedeki verilmeyen her rakam için ihtimalleri yazmaya gerek yoktur.

Burada "ikili" olarak kastedilen, bir bölgede (bir satır veya bir sütunda) iki rakamın gelebileceği yerler düşünülüp ihtimalleri bölgedeki boşluklara yazıldığında bu iki rakam sadece **aynı iki boşluğa gelebilme ihtimalidir** (yedinci bölgede 4 ve 9 rakamları sadece c7 ve c8'e gelebilme ihtimalleri vardır. üst ve yan bölgelerdeki 4 ve 9'ların yerlerinden dolayı).

Resimde yedinci bölgede 4 ve 9 rakamlarının **sadece aynı iki yere** gelebilme ihtimali (c7 ve c8), bu iki boşluğa başka rakam gelebilme ihtimalini ortadan kaldırmaktadır. Çünkü bu bölgedeki bu iki boşluktan başka bir yere gelemiyorsa bu iki boşluktan birinde 4, diğerinde de 9 olmalıdır. Hangisinin tam olarak c7'de mi c8'de mi olduğunu bilmesekte c7 ve c8'in 4 ve 9 rakamları ile dolu olduğunu emin olabiliriz.

Yedinci bölgede 4 ve 9 rakamlarnıın yerlerinin belli olduğunundan hareketle, o bölgede başka bir rakamın gelebileceği yeri bulmaya çalışalım. Dördüncü bölgedeki 6'nın ve dokuzuncu bölgedeki 6'nın pozisyonunda dolayı yedinci bölgede 6 rakamının gelebileceği yerler, c8 ve b9 gibi görünmektedir. Ancak biraz önce bahsettiğimiz durumdan dolayı c8'e 4 veya 9'dan başkası gelemez bu yüzden yedinci bölgede **6 rakamı** sadece **b9**'a gelebilir.

Satır veya sütun içindeki 'gizli ikililer', Sudoku çözerken ihtimallerin yazılmasından önceki yöntemde bahsetmiştik.

Resimde gördüğünüz sudokuda yedinci satırda ihtimalleri yazarak ikililer elde edebiliyor muyuz bakalım.

A7'ye gelebilecek rakamlar

Yedinci bölgede 1, 2 ve 4 olduğundan a7'ye 3'ün gelebileceğini düşünerek başlayalım. a7'nin satırında ve sütununda başka 3 rakamı olmadığından a7'ye **3** gelebilir.

Sıra ile gidersek a7'ye 5 gelebilir. a7'nin satırında ve sütununda 5 rakamı olmadığından a7'ye **5** gelebilir.

a7'ye 6 gelemez. a7'nin satırında 6 rakamı olduğundan (h7) a7'ye 6 rakamı gelemez.

a7'ye 7 gelemez. a7'nin sütununda 7 rakamı olduğundan (a4) a7'ye 7 rakamı gelemez.

a7'ye 8 gelebilir. a7'nin satırında ve sütununda 8 rakamı olmadığından a7'ye **8** gelebilir.

a7'ye 9 gelebilir. a7'nin satırında ve sütununda 9 rakamı olmadığından a7'ye **9** gelebilir.

a7 karesine 3, 5, 8, 9 rakamları gelebilir.

A7 satırındaki diğer boşluklara da bu şekilde gelebilecek rakam ihtimallerini yazalım.

Daha önceki yöntemde bahsettiğimiz "ikililer" bazen açık olarak görünmeyebilirler. Burada kendimize şu soruyu sormamız gerekiyor: "Bu satırda herhangi iki rakam sadece aynı iki karede bulunuyor mu?".

f7 ve g7 karelerine baktığımızda 1 ve 4 rakamlarının bu satır içinde sadece bu iki karede olduğunu görebiliriz. Bu şu anlama geliyor; 1 ve 4 rakamları a7 sütununda sadece f7 ve g7 karelerine gelebiliyorsa, 1 ve 4 rakamları mutlaka bu iki karede yer almalıdır. f7 ve g7'de görülen ihtimaller içinde başka sayılar olsa bile f7 ve g7'ye yalnız 1 ve 4 yazılabilir. f7 ve g7'de diğer ihtimalleri eleyebiliriz. Yedinci satırda 1 ve 4 rakamlarının yerini tam olarak belirlememiz(!) ve bu yerlerdeki diğer itimalleri silebilmemiz bize yedinci satıra başka rakamların ihtimalleri ile ilgili önemli bir ipucu kazandırdı.

Yedinci satırda 7 rakamın sadece d7'de yazılı olduğunu görebiliyoruz. Bu satıra gelebilecek 7 rakamı sadece d7'de bulunabilirmiş. Bu satırda başka hiç yere gelebilme ihtimali yokmuş.

O zaman d7'de kesin olarak 7 rakamı olmalıdır.

1 ve 4 ikilisi sayesinde yedinci satırda bir rakamın yerini kesin olarak bulmuş olduk.

Sözde Kod

```
1  fonksiyon FileReader ( Sudoku, kaynak ):

2 Txt'yi aç

3 read()

4 Format sudokuyu oluştur.

5  int[][] Format = {}

6  for() {for() { yeniSudoku[x][y] = Format[x][y];

7  Threadleri çağır //new,

8  thread1.start(); //başlatma

9  fonksiyon SudokuCoz():

10 Threadleri gönder

11 fonksiyon Onay():

12 //Senkronizasyonu kontrol et
13 fonksiyon SudokuYazdir():

14 Thread.currentThread().getName()

15 Thread bilgilerini bastır.
```

Yazılım Mimarisi

```
1  fonksiyon FileReader ( Sudoku, kaynak ):

2 Txt'yi aç

3 read()

4 Format sudokuyu oluştur.

5  int[][] Format = {}

6  for() {for() { yeniSudoku[x][y] = Format[x][y];

7  Threadleri çağır //new,

8  thread.start(); //tüm threadleri başlatma

9  fonksiyon SudokuCoz():

10 try{ thread.interrupt(); } catch(Exception exep){ System.out.println(Error); }
   //Threadleri kontrol et

11 Threadleri gönder

12 fonksiyon Onay():

13 Satırları kontrol et

14 Sütunları kontrol et

15 /* if (++sutun == 9){ sutun = 0; if(++satir == 9) satir = 0;} */

16 3x3'lük parçalanmış matrisleri kontrol et

17 //Senkronizasyonu kontrol et
18 fonksiyon SudokuYazdir():

19 Thread.currentThread().getName()

20 Thread bilgilerini bastır.
```

Referanslar

- https://www.samurai-sudoku.com/
- https://sudoku.matematiktutkusu.com/yontemler/
- *https://www.yusufsezer.com.tr/java-thread/web.karabuk.edu.tr/*
- *https://www.geeksforgeeks.org/multithreading-in-java/*