Samurai Sudoku Çözme

Kaan Kalaycı, Alperen Ünlü

Bilgisayar Mühendisliği Bölümü

Kocaeli Üniversitesi

ahmet.alper96@gmail.com.tr, kaan.klyc1@hotmail.com

Problem Tanımı:

Yazılım Laboratuvarı I dersinin ikinci projesi olarak bizden multithread yapısını kullanarak verilecek samurai sudokuyu çözen bir program istenmektedir. Samurai sudokunun başlangıç değerleri .txt uzantılı dosyada verilecek ve verilen değerler dinamik olarak programa aktarılacaktır. Projenin 3 isteri bulunmaktadır. Birincisi verilen samurai sudokunun içindeki her bir sudoku için başlangıç noktası seçerek 5 thread ile çözülmesi, ikincisi samurai sudokunun içindeki her bir sudoku için 2 başlangıç noktası seçerek 10 thread ile çözülmesi, üçüncüsü ise ilk iki isterlerin yaptıkları çözümlerin çalışma zamanını ve doğru buldukları kare sayısını bir metin belgesi veya veri tabanında tutarak ve ardından bu bilgileri çekerek bir karşılaştırma grafiği oluşturulmasıdır.

Yapılan Araştırmalar:

Projemizde ilk karşılaştığımız sorun .txt dosyasında samurai sudokuda bulunan boşlukları belirten bir öğenin bulunmamasıydı. Bunu çözmek için haritaOluştur metodumuzda .txt dosyasının o anda okunan satırının uzunluğuna bakarak uygun yerlere 0 sayısını yerleştirdik. Ara yüzümüze sudoku tahtasını çizerken karenin içinde bulunan sayıyı yerleştirmesi için drawString metodunu kullandık ama bu metot sadece string tipinde elemanları kullanabildiği için String.valueOf metoduyla bu sorunu çözdük. Köşelerdeki sudokular çözülürken ortayla çakışan elemanlara dikkat edilmiyordu. Bunun için Yerleştir metodumuzun içine gönderilen sudokuya göre çakıştığı yerlerdeki elemanlara dikkat eden bir if'i barından iç içe for döngüsü oluşturduk. Threadler sudokuları çözemezse çözmeyi tekrar denemesi için giriş adında bir başlangıç noktası oluşturduk ve else durumunda continue kullanarak çözümü giriş'ten tekrar başlattık.

Yazılım Mimarisi:

Bu programı Java programlama dilini kullanarak Apache Netbeans geliştirme ortamında yazdık. Ara yüz tasarımı için Swing ve Graphics kütüphanelerini kullandık. Programımızı yazarken aşağıdaki metotları tasarlayıp kullandık.

-static void dosyayıOku(): Bu metot içine yazılan dosyanın her satırını okuyup haritaOluştur metoduna gönderiyor.

-static void haritaOluştur(String dosyasatır): Bu metot parametre olarak satırın uzunluğuna bakarak uygun şekilde samurai sudoku tahtasını tutan diziyi dolduruyor.

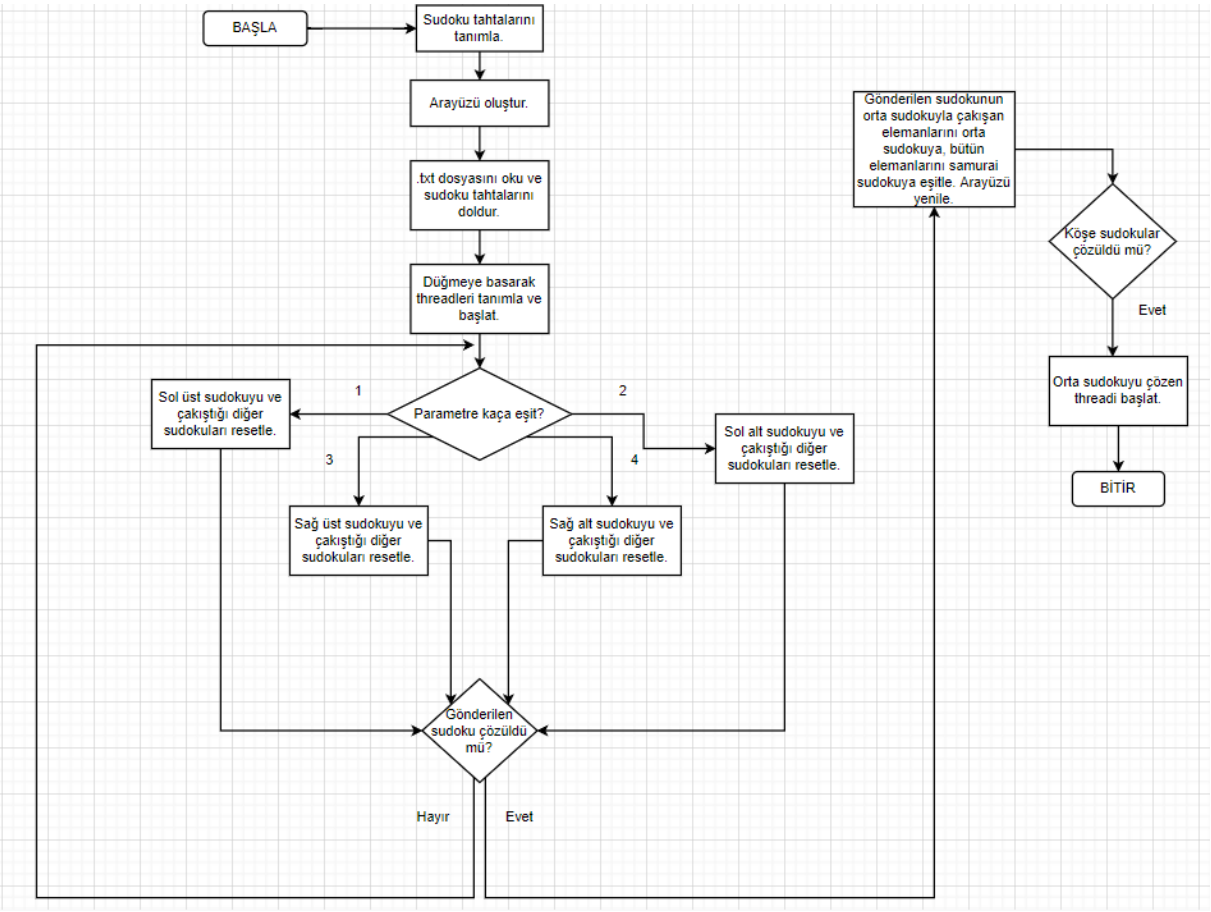
-static void parçala(int[][] samurai): Bu metot gönderilen samurai sudoku tahtasını 5 parçaya bölüyor.

-public static boolean SudokuÇöz(int[][] tahta, int n, int parametre): Bu metot içine gönderilen sudoku tahtasının dolu olup olmadığına bakıyor. Ardından dolu değilse tahtanın bütün karelerine Yerleştir metodunu kullanarak uygun sayıları yerleştirmeye çalışıyor.

-public static boolean Yerleştir(int[][] tahta, int tahtasatır, int tahtasütun, int sayı, int parametre): Bu metot içine gönderilen sayının tahtaya yerleştirmeye uygun olup olmadığına bakıyor.

-static void çöz(): Bu metot düğmeye basıldığında samurai sudokunun 4 köşesini çözen 4 threadi oluşturup başlatıyor. Threadlerin yapısından dolayı 4 thread de köşeleri çözebildiyse otomatik olarak orta kısmı çözen 5.thread başlıyor ve program sonlanıyor.

Akış Şeması:



Veri Tabanı Diyagramı:

Çalışmamızın grafik oluşturma kısmını tamamlayamadığımız için bir veri tabanımız bulunmuyor.

Genel Yapı:

Programımız gönderilen samurai sudoku tahtasını 5 parçaya bölüp ilk önce 4 köşenin çözülmesinin ardından orta kısmın çözülmesi şeklinde çalışıyor. Bunun için her parçaya ait birer dizi ve bütün samurai sudoku tahtasını tutan bir dizi oluşturduk. .txt dosyası okunurken samurai sudokuyu tutacak dizi dolduruluyor ardından parçala metoduyla samurai sudokuyu tutan dizinin elemanları 5 parçaya ayrılıyor. Sonradan ara yüz açılıyor. Ara yüz samurai sudokuyu tutan diziye göre sudoku tahtasını oluşturuyor. Ara yüzde bulunan butona bastığımızda sudokunun parçalarını çözen çöz metodu çağrılıyor. Bu metot köşeleri çözecek olan 4 thread oluşturuyor ve bu threadleri başlatıyor. Bu köşeleri çözen threadler orta kısmın köşelerle çakışan elemanlarını dolduruyor. Ayrıca samurai

sudoku dizisine bulunan elemanları yolluyor ve ara yüzü yeniden çizdiriyor. Eğer gönderilen sudoku çözülemediyse else durumuna giriyor ve threadin başına dönüp tekrar çözüm deneniyor. Threadlerin içinde bir kontrol sayacı bulunuyor. Bu sayaç bütün köşe elemanları bulunduğunda threadde bulunan bir if'i çalıştırıyor ve orta kısmı çözecek olan thread tanımlanıp çalışmaya başlıyor. Orta kısım da bulunduğunda program sonlanmış oluyor.

Referanslar:

https://www.geeksforgeeks.org/sudoku-backtracking-7/

https://www.geeksforgeeks.org/multithreading-in-java/

https://www.javatpoint.com/java-string-valueof

https://www.youtube.com/watch?v=mTGdtC9f4EU

https://stackoverflow.com/questions/877096/how-can-i-pass-a-parameter-to-a-java-thread

https://www.youtube.com/watch?v=-IMys4PCkIA