SAMURAI SUDOKU ÇÖZME

Berk Sunduri

Bilgisayar Mühendisliği Bölümü

Kocaeli Üniversitesi

berksunduri@gmail.com

1-)Projenin Tanımı

Bu kısım sadece projenin açıklamasını okuyup edindiğim ön bilgiye göre yazılmıştır.

Bize verilen pdf dosyasında projenin amacını öğrendim. Verilen isterleri dikkatlice okudum

Projede bizden istenen şeyin bir multi-threading kullanarak bize sunulacak txt dosyasına dayanarak bir Samurai Sudoku çözmek olduğunu anladım.

1.1-)Problem Tanımı

Bu kısımda bizden yapmamız istenilenler içermektedir.

Birinci adımda bizden her bir sudoku için bir başlangıç noktası seçip bunlara birer thread ayarlayarak çözüme ulaşmamız istenmiştir.

İkinci ister olarak aynı şekilde sadece thread ve başlangıç noktası değerlerini değiştirerek sudokuyu çözmemiz istenmiştir.

Üçüncü aşamada ise bizden bulduğumuz çözüm kare sayısı ve geçen zaman arasında bir grafik oluşturmamız istenmiştir.

2-)Yapılan Araştırmalar ve Karşılaşılan Sıkıntılar

Bu kısım proje öncesi ve sonrası araştırmaları ve de projenin yapım aşamasındaki sıkıntıları ve çözümlerini içermektedir.

İlk karşılaştığım sorun hangi programlama dilini kullanacağım olduğuydu. Bunun için Python dilini kullanmada karar kıldım. Veri işlemek için uygun bir dil olacağını düşündüm fakat threading kısmında beni zorladı.

Daha sonrasında Samurai Sudoku'yu çözmek için hangi algoritmayı kullanıcağım konusunda kararsız kaldım. İlk olarak Brute-Force algoritmasını denedim fakat çözmesi hem çok uzun sürüyor hem de çözümler her zaman tutarlı değildi. Daha sonra backtracking algoritmasını denedim. Çözüme ulaşmada sıkıntı çekmedim fakat kesişen noktalarda program sürekli takılıyordu. Bende en son olarak Eliminasyon metodunu kullanmakta karar kıldım.

2.1-Proje Sırasında Yararlanılan Teknolojiler

Projeyi Python dili kullanarak PyCharm IDE'sinde yazdım.

Programı yaparken Python'un birkaç kütüphanesinden yararlandım.

3-)Tasarım

3.1-Akış Diyagramı

Kısım ektedir.(1)

4-)Genel Yapı

4.1-Kullanıcı Kısmı

Program çalıştığında ilk olarak karşımıza kullanıcının txt dosyasının bulunduğu klasörü ve txt dosyasının ismini yazması beklenmektedir.

Kullanıcı bu kısmı tamamladıktan sonra program otomatik olarak kendisini tamamlıyıp Sudoku'yu çözücektir

4.2-Kod Kısmı

Kod kısmına baktığımızda ise kod tamamiyle fonksiyonlar yazılarak geliştirilmiştir. En başta ilk önce Sudoku'nun çeşitli parçalarının nasıl ayrılacağı işlemlerini yapan değişkenler bulunmaktadır.

Bundan sonra programın çalışması ve yazılabilirliğini kolaylaştırmak için tam oniki tane fonksiyon gelmektedir.

Kullandığım fonksiyonlar txt dosyasını okumaktan başlayıp çözmek ve sonunda görüntülemek için tüm işlemleri yapmaktadır.

def cartesianProd()

Yollanan iki değer için kartezyen çarpanlarını döndürür.

def replaceFileData()

Dosyadan alınan tüm sayıları çözümü kolaylaştırmak için bir koordinat sistemine benzer bir şekilde kodlar.

def parseGrid()

Girilecek olan gride kutulara gelecek muhtemel olan sayılar ekler. Başlangıç olarak tüm kutucuklar herhangi bir sayı olabilir. Dosya okuma işlemi yaptıktan sonra kutucuk değerleri değişir.

def flatten()

Yollanacak arrayi sortlar.

def gridValues()

Yollanacak gridi hangi Sudoku olduğunu anlayıp (Sol üst veye merkez) onları düzgün bir şekilde diğer işlemler için formatlar.

def assign()

Dosyadan girilen veriler dışında diğer her sayıyı dosyadan girilen veriler içinde olmayacak şekilde düzelt. Muhtemel değerleri geri döndürür eğer yoksa False döndürür.

def eliminate()

Yollanan sayıları eğer kutucukta olabilir mi diye kontrol eder. Proje'nin ana algoritması budur. Eğer kutucuktan sayı zaten elimine edilmişse bir sonra ki kutucuğa geçer. Eğer verilen sayı yanlışsa False döndürüp kendini tekrarlar. Doğru sayı bulunup elimine edilirse sayıyı döndürüp bir sonra ki kutucuğa geçer.

def display()

Yollanan sayılar ve hangi sudoku kutusuna ait olduğu bilgileriyle sudokuyu konsola daha güzel bir şekilde yazdırmaya yarar.

def displaySamurai()

Sudokuyu konsolda sol üstten başlayarak doğru şekilde yazdırma işlemidir. Ayrıca bu işlemde çözülen sudokunun doğruluğuda saptanır. Doğruluk testi ayrı bir dosyada yapılmıştır.

def solve()

Yollayacağımız gridi sadece doğru bir şekilde kodlamamız için var olan bir fonksiyon.

def search()

Depth First Search yapılarak tüm olası sayılar bulunur.

def some()

İnternet yardımı alınarak yazılan bir kod parçacığıdır. Search için gereklidir. Yollanılan sekansta doğru olan tüm sayıları geri döndürür.

Kullandığum kütüphaneler raporun 5. bölümünde belirtilmiştir.

5-)Kütüphaneler

Bu kısımda projeye include ettiğim kütüphaneler bulunmakta:

1-)time

2-)ThreadPoolExecutor, Concurrent.Futures librarysinden.

7-)Referanslar

1-)Solving Every Sudoku Puzzle

"norvig.com/sudoku.html"

2-) Slaying the Sudoku Puzzle

" levelup.gitconnected.com "

3-) Stack Over Flow
"https://www.stackoverflow.com"

4-)Pyhton Sudoku Solver

www.techwithtim.com

6-)Corey Schafer-Pyhton Threading Tutorial

7-)Multithreading in Pyhton

"www.geeksforgeeks.com"