MULTITHREAD KULLANARAK SAMURAI SUDOKU ÇÖZME

YAZILIM LABORATUVARI-II PROJESİ

Yaren KASIMOĞLU-Dilara ÇATALTEPE

160201060 - 180201002

Bilgisayar Mühendisliği Bölümü

Kocaeli Üniversitesi

ÖZET

Thread-Multithread öğrenimini pekiştirmek için verilen bu projede Samurai Sudoku çözmemiz istenmektedir.

Samurai sudoku birden fazla sudokunun (5 tane) iç içer hali ile çözülmesidir.

Önce ortadakini çözmeye başlayıp ardından sırasız bir şekilde sağ sol alt-üst kısımlarının çözümleriyle devam edilir.

1.Giriş

Projede bizden multithread ile samurai sudoku çözmemiz istenmektedir. Threadler varlıkları bir işleme bağlı olan ve yine aynı anda birden fazla işi yapmaya yarayan yapılardır. Ancak thread'in var olması için önce bir işlemin var olması gerekir. Multi-Threaded programlama bir programda aynı anda birden fazla işin yapılabilmesidir. Yani bir kod parçası işlemi gerçekleştirirken aynı anda ona paralel olarak bir başka kod parçasının çalışması demektir.

Projede istenenler şunlardır :

Hazır kütüphaneleri hiçbir şekilde kullanmadan verilen samurai sudoku içindeki herbir sudoku için 2 tane başlangıç noktası seçmeliyiz.

Ve 10 thread ile çözüme ulaşmalıyız.

Threadler çözüme eş zamanlı başlamalıdır. Senkronizasyon problemini göz önünde bulundurarak istediğimiz bir yöntem ile çözüm yapmamız istenmektedir.

2.Tasarım

Projenin programlanma aşamaları aşağıda belirtilen başlıklar altında açıklanmıştır.

2.Temel Bilgiler

Program Python dilinde yazılmış olup proje gelişiminde;

Geliştirme Ortamı olarak "Anaconda / Spider" kullanılmıştır. Projeyi başlangıçta 5 threadle başlatıyoruz ama aslında 2 farklı nokta seçerek başlattığımız için eş zamanlı 10 thread çalıştırmış oluyoruz.

Öncelikle normal sudoku çözüer gibi ortadaki büyük sudokunun çözümü gerçekleşiyor ve ardından sağ-sol alt-üst kısımlar gerçekleşiyor.
Ortadaki sudokunun bir çok çözümü olabilir aslında. Fakat samurai sudokunun içerisinde olduğu için ortadaki sudokunun çözüm sayısı da oldukça düşüyor. Program önce ortadaki sudokunun çözümü tamamlandıktan sonra diğer sudokulara geçiyor. Yanlış çözüm olduğu takdirde sudokular sıfırlanıyor ve diğer çözüme geçiliyor.
Bu işlemlerin tekrar etmesi sonucunda tammalanan sudoku ekrana yazdırılıyor.
Toplamda 8 tane sınıf kullanılmıştır.

class cell:
Txt den getirdiğimiz değeri ve sudokunun içersinine yerleştirilmesi gereken değeri tanımlayan sınıftır.

class sudokuSolver:
Hücrelerin, satır-sütun ve her bir 9lu kutucuk için, boş olup olmadığını boş ise doğru yer olup olmadığını kontrol ederek sudokunun çözülmesini gerçekleştiren sınıftır.

class sudokuSolverReverse:
Sudokuyu tersten de çözmeye başyalan sınıftır. Sağ alt kısımdan başlayarak sağdan sola aşağıdan yukarıya doğru çözüm yapar.

class centerSudoku:
Bu sınıf merkez sudokunun çözüldüğü sınıftır.
Öncelikle sırasız bir şekilde sağ üst sol üst sağ alt sol alt kısımlar yani merkez sudokunun diğer sudokularla olan ortak kısımlarının çözülmesi yapılır. Ardından birbirine döre doğrulukları kontrol edilir ve kalan kısmın çözümü yapılmaya başlar.
Sudokunun çözümündeki doğru olan adımlar ‘trueSteps’ txt dosyasına kaydedilir.

***class centerSudokuReverse:**
Multithread kullandığımız için senkronizasyonun sağlanması açısından merkez sudokuyu tersten çözüdürmeye başlayan sınıftır.

runSudoku:
Bu sınıfta threadleri oluşturuyoruz ve bir array içerisine atıyoruz sonra hepsini aynı anda başlatıyoruz, tek tek her bir sudokunun başarıya ulaşıp ulaşmadığını kontrol ediyoruz.

Yanlış çözüme ulaşmış yani çözülememiş sudoku varsa tüm sudokuları
resetliyoruz txt değerlerinegeri döndürüyoruz ve yeniden baştan alarak devam ediyor.

loadSudoku:

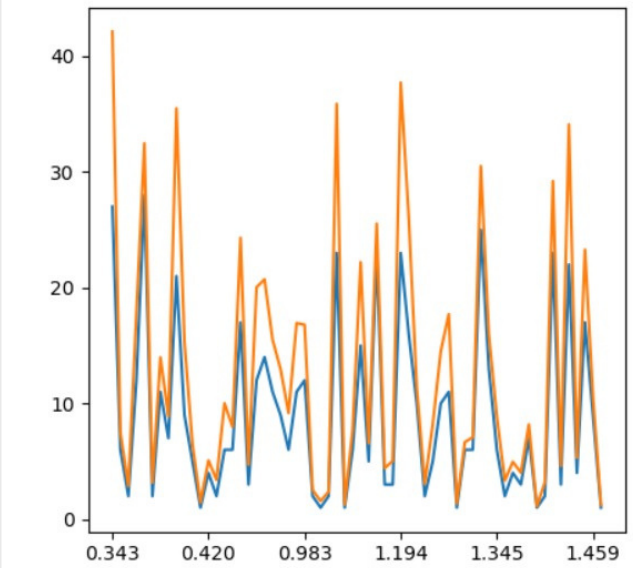
Kendi yüklediğimiz txt deki sudokuyu karakter karakter okutma işleminin yapıldığı sınıftır.
Boş değerleri yani txt de * olan değerleri 0 olarak atayıp diğerlerinin txt deki değerlerini giriyoruz.
Bu sınıf içerisinde ekstra bir thread daha kullandık. Programı çalıştırırken aynı zamanda adımları da yazdırıyor olacağımız için beklememek için adım yazdırmayı da bir thread e attık. True steps txt dosyasına yazdırılıyor.

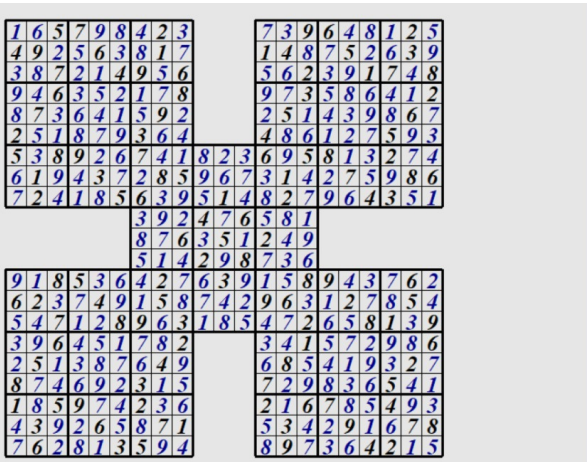
class UserInterface:

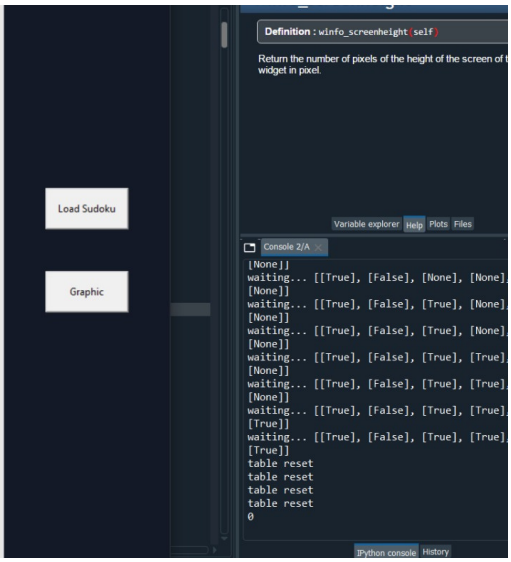
Bu sınıfta arayüz kısmını oluşturduk. Sol tarafta sudoku tablosunu ve sağ tarafta iki tane buton. Sağdaki butonlardan birisi bilgisayardan bir txt dosyasını yüklememizi sağlıyor. Diğer buton ise süreleri grafiğe döken butondur.
Grafik kısmının arayüzünde grafiği yakınlaştırıp uzaklaştırma, kaydetme gibi özellikler ekli.

Kaynakçalar:

https://github.com/busraerkannn/MULTITHREAD-PROJESI
https://www.samurai-sudoku.com/
https://www.youtube.com/watch?v=rr_xeeJeaY8
https://www.youtube.com/watch?v=GqHLztqy0PU


40
30
20
10
0
0.343
0.420
0.983
1.194
1.345
1.459




Load Sudoku
Graphic
Definition : winfo_screenheight(self)
Return the number of pixels of the height of the screen of t
widget in pixel.
Variable explorer
Help
Plots
Files
Console 2/A
[None]]
waiting... [[True], [False], [None], [None]
[None]]
waiting... [[True], [False], [True], [None]
[None]]
waiting... [[True], [False], [True], [None]
[None]]
waiting... [[True], [False], [True], [True]
[None]]
waiting... [[True], [False], [True], [True]
[None]]
waiting... [[True], [False], [True], [True]
[True]]
waiting... [[True], [False], [True], [True]
[True]]
table reset
table reset
table reset
table reset
0
IPython console
History


tk
Load Sudoku
Graphic
LSP Python: ready