ALGORİTMA VE PROGRAMLAMA

**2.PROJE**

HİTORİ OYUNU

**Ahmet Furkan ERDEM - 05180000057**

**Burak KIZILAY - 05180000063**

**Teslim Tarihi: 31.12.2018**

**İÇİNDEKİLER**

1. Analiz......................................................3
2. Kaynak Kod.............................................4
3. Programcı Kataloğu................................8
4. Kullanıcı Kataloğu...................................9

4.1. Kısıtlamalar....................................11

**1. Analiz**

Bizden Hitori bulmaca oyununun oynanmasını sağlayacak bir program yapılması istenmektedir. **Hitori oyunu**: hepsi [1-n] arasında bir sayı içeren karelerden oluşan nxn büyüklüğünde kare şeklindeki bir oyun alanı üzerinde oynanan bir mantık bulmacasıdır. Oyunda amaç, aşağıdaki kurallara göre hangi karelerin boşaltılması gerektiğinin bulunmasıdır:

1. Bir satır veya sütunda, aynı sayı birden fazla kez bulunamaz (Hiçbir sayı, bir satır veya sütunda birden fazla kez bulunamaz)

2. Boş kareler, yatay ya da dikey olarak birbirlerine komşu olamazlar (Boş kareler, yatay ya da dikey olarak yan yana (bitişik) yer alamazlar, çapraz olarak yer alabilirler)

3. Dolu karelerin hepsi, yatay ve/veya dikey olarak birbirine bağlı olmalıdır (Dolu karelerin hepsi, tek bir (yatay ve/veya dikey) bağlantılı grup oluşturmalıdır)

Programın çalışma aşamasında ise kullanıcıdan değiştirmek istediği sayının koordinatlarını satır sütun biçiminde isteyen ve aynı zamanda yapılacak işlemi isteyen bir kısım gereklidir. Yazılan **liste\_al** ve **değişiklik** fonksiyonları ile bu rahatça sağlanmaktadır.

Sonra yapılan bu değişikliklerin kontrolünün sağlanıp her kontrolde tablonun çizilmesi gerekmektedir. Yazılan **kontrol , butunluk\_kontrol** ve **tablo\_ciz** fonksiyonları ile bu da kolayca sağlanmaktadır.

**2. KAYNAK KOD**

import copy  
def liste\_al():  
 hitori = open("hitori\_bulmaca.txt","r+")  
 liste = []  
 satir = hitori.readline()  
 satir\_say = 0  
 while satir != "" :  
 ara\_liste = []  
 for sayilar in satir.rsplit():  
 ara\_liste.append("-")  
 ara\_liste.append(sayilar)  
 liste.append(ara\_liste)  
 satir = hitori.readline()  
 satir\_say += 1  
 sifirlama\_listesi = copy.deepcopy(liste) #bidaha for açmaya üşendim offff...  
 ters\_liste = []  
 for i in range(satir\_say):  
 ters\_liste.append([0] \* satir\_say)  
 tablo\_ciz(liste,satir\_say)  
 degisiklik(liste,sifirlama\_listesi,ters\_liste,satir\_say)  
 hitori.close()  
  
def tablo\_ciz(liste,satir\_say):  
 print(end=" ")  
 for satir in range(satir\_say):  
 print(end=" ")  
 print(satir+1,end=" ")  
 print()  
 for satir in range(satir\_say):  
 print(satir+1,end=" ")  
 for sutun in range(satir\_say):  
 if liste[satir][2 \* sutun] == "-":  
 print("-",liste[satir][2 \* sutun + 1],"-",sep="",end="")  
 else:  
 print("(", liste[satir][2 \* sutun + 1], ")", sep="", end="")  
 print()  
  
def degisiklik(liste,sifirlama\_listesi,ters\_liste,satir\_say):  
 try:  
 yapilacak\_degisiklik = input("satır,sütun,işlem")  
 degisiklik\_listesi = yapilacak\_degisiklik.rsplit()  
 while int(degisiklik\_listesi[0])<0 or int(degisiklik\_listesi[0])>satir\_say or int(degisiklik\_listesi[1])<0 or int(degisiklik\_listesi[1])>satir\_say or not(degisiklik\_listesi[2] in ["B","b","D","d","N","n"]):  
 yapilacak\_degisiklik = input("satır,sütun,işlem hatalı girdiniz!")  
 degisiklik\_listesi = yapilacak\_degisiklik.rsplit()  
 satir = int(degisiklik\_listesi[0])  
 sutun = int(degisiklik\_listesi[1])  
 islem = degisiklik\_listesi[2]  
 except ValueError or UnboundLocalError:  
 yapilacak\_degisiklik = input("satır,sütun,işlem hatalı girdiniz!")  
 degisiklik\_listesi = yapilacak\_degisiklik.rsplit()  
 if islem == "B" or islem == "b":  
 liste[satir - 1][2 \* sutun - 2] = "-"  
 liste[satir-1][2 \* sutun -1] = "X"  
 elif islem == "D" or islem == "d":  
 liste[satir - 1][2 \* sutun - 2] = "("  
 else:  
 liste[satir - 1][2 \* sutun - 2] = "-"  
 liste[satir - 1][2 \* sutun - 1] = sifirlama\_listesi[satir - 1][2 \* sutun -1]  
 tablo\_ciz(liste,satir\_say)  
 kontrol(liste,sifirlama\_listesi,ters\_liste,satir\_say)  
  
  
def butunluk\_kontrolu(liste, adres\_listesi, satir\_say, satir=0, sutun=0):  
 if satir == 0 and sutun == 0 and liste[satir][sutun] == "X":  
 butunluk\_kontrolu(liste, adres\_listesi, satir\_say, satir, sutun + 1)  
 if satir\_say - 1 > satir > 0 and satir\_say - 1 > sutun > 0 and liste[satir][sutun] != "X" and not (str(satir) + str(sutun) in adres\_listesi):  
 adres\_listesi[str(satir) + str(sutun)] = 1  
 butunluk\_kontrolu(liste, adres\_listesi, satir\_say, satir, sutun + 1) # sağını  
 butunluk\_kontrolu(liste, adres\_listesi, satir\_say, satir + 1, sutun) # aşağısını  
 butunluk\_kontrolu(liste, adres\_listesi, satir\_say, satir, sutun - 1) # solunu  
 butunluk\_kontrolu(liste, adres\_listesi, satir\_say, satir - 1, sutun) # yukarısını  
  
 elif satir == 0 and satir\_say - 1 > sutun > 0 and liste[satir][sutun] != "X" and not (str(satir) + str(sutun) in adres\_listesi):  
 adres\_listesi[str(satir) + str(sutun)] = 1  
 butunluk\_kontrolu(liste, adres\_listesi, satir\_say, satir, sutun + 1)  
 butunluk\_kontrolu(liste, adres\_listesi, satir\_say, satir + 1, sutun)  
 butunluk\_kontrolu(liste, adres\_listesi, satir\_say, satir, sutun - 1)  
  
 elif satir\_say - 1 > satir > 0 and sutun == satir\_say - 1 and liste[satir][sutun] != "X" and not (str(satir) + str(sutun) in adres\_listesi):  
 adres\_listesi[str(satir) + str(sutun)] = 1  
 butunluk\_kontrolu(liste, adres\_listesi, satir\_say, satir + 1, sutun)  
 butunluk\_kontrolu(liste, adres\_listesi, satir\_say, satir, sutun - 1)  
 butunluk\_kontrolu(liste, adres\_listesi, satir\_say, satir - 1, sutun)  
  
 elif satir\_say - 1 > satir > 0 and sutun == 0 and liste[satir][sutun] != "X" and not (str(satir) + str(sutun) in adres\_listesi):  
 adres\_listesi[str(satir) + str(sutun)] = 1  
 butunluk\_kontrolu(liste, adres\_listesi, satir\_say, satir, sutun + 1) # sağını  
 butunluk\_kontrolu(liste, adres\_listesi, satir\_say, satir + 1, sutun) # aşağısını  
 butunluk\_kontrolu(liste, adres\_listesi, satir\_say, satir - 1, sutun) # yukarısını  
  
 elif satir == satir\_say - 1 and satir\_say - 1 > sutun > 0 and liste[satir][sutun] != "X" and not (str(satir) + str(sutun) in adres\_listesi):  
 adres\_listesi[str(satir) + str(sutun)] = 1  
 butunluk\_kontrolu(liste, adres\_listesi, satir\_say, satir, sutun + 1) # sağını  
 butunluk\_kontrolu(liste, adres\_listesi, satir\_say, satir, sutun - 1) # solunu  
 butunluk\_kontrolu(liste, adres\_listesi, satir\_say, satir - 1, sutun) # yukarısını  
  
 elif satir == 0 and sutun == 0 and liste[satir][sutun] != "X" and not (str(satir) + str(sutun) in adres\_listesi):  
 adres\_listesi[str(satir) + str(sutun)] = 1  
 butunluk\_kontrolu(liste, adres\_listesi, satir\_say, satir, sutun + 1)  
 butunluk\_kontrolu(liste, adres\_listesi, satir\_say, satir + 1, sutun)  
  
 elif satir == 0 and sutun == satir\_say - 1 and liste[satir][sutun] != "X" and not (str(satir) + str(sutun) in adres\_listesi):  
 adres\_listesi[str(satir) + str(sutun)] = 1  
 butunluk\_kontrolu(liste, adres\_listesi, satir\_say, satir + 1, sutun) # aşağısını  
 butunluk\_kontrolu(liste, adres\_listesi, satir\_say, satir, sutun - 1) # solunu  
  
 elif satir == satir\_say - 1 and sutun == satir\_say - 1 and liste[satir][sutun] != "X" and not (str(satir) + str(sutun) in adres\_listesi):  
 adres\_listesi[str(satir) + str(sutun)] = 1  
 butunluk\_kontrolu(liste, adres\_listesi, satir\_say, satir, sutun - 1) # solunu  
 butunluk\_kontrolu(liste, adres\_listesi, satir\_say, satir - 1, sutun) # yukarısını  
  
 elif satir == satir\_say - 1 and sutun == 0 and liste[satir][sutun] != "X" and not (str(satir) + str(sutun) in adres\_listesi):  
 adres\_listesi[str(satir) + str(sutun)] = 1  
 butunluk\_kontrolu(liste, adres\_listesi, satir\_say, satir, sutun + 1) # sağını  
 butunluk\_kontrolu(liste, adres\_listesi, satir\_say, satir - 1, sutun) # yukarısını  
 return adres\_listesi  
  
def kontrol(liste,sifirlama\_listesi,ters\_liste,satir\_say):  
 rakam\_say=0  
 adres\_listesi={}  
 yatay\_ayni\_rakam\_say = 0  
 yan\_yana\_bosluk = 0  
 for satir in range(satir\_say):  
 for sutun in range(satir\_say):  
 if liste[satir][2 \* sutun + 1]!="X":  
 rakam\_say+=1  
 s1\_liste=liste[satir][:2\*sutun+1] #s1 ve s2 o an incelenen rakamı içermeyen 2 dilime ayrılmış satır listesinin halleridir  
 s2\_liste=liste[satir][2\*sutun+2:] #s --> slice  
 ters\_liste[sutun][satir] = liste[satir][2 \* sutun + 1]  
 rakam = liste[satir][2 \* sutun + 1]  
 if (rakam in s1\_liste or rakam in s2\_liste) and rakam != "X":  
 yatay\_ayni\_rakam\_say += 1  
 if sutun <= satir\_say - 2 and yatay\_ayni\_rakam\_say==0:  
 if liste[satir][2 \* sutun + 1] == liste[satir][2 \* (sutun + 1) + 1] and liste[satir][2 \* sutun + 1] == "X":  
 yan\_yana\_bosluk += 1  
  
 dikey\_ayni\_rakam\_say = 0  
 ust\_uste\_bosluk = 0  
 if yatay\_ayni\_rakam\_say==0 and yan\_yana\_bosluk==0:  
 for sutun in range(satir\_say):  
 for satir in range(satir\_say):  
 rakam = ters\_liste[sutun][satir]  
 s1\_liste = ters\_liste[sutun][:satir] #s1 ve s2 o an incelenen rakamı içermeyen 2 dilime ayrılmış satır listesinin halleridir  
 s2\_liste = ters\_liste[sutun][satir+1:]  
 if (rakam in s1\_liste or rakam in s2\_liste) and rakam != "X":  
 dikey\_ayni\_rakam\_say+=1  
 if satir <= satir\_say - 2:  
 if ters\_liste[sutun][satir] == ters\_liste[sutun][satir+1] and ters\_liste[sutun][satir] == "X":  
 ust\_uste\_bosluk += 1  
 if yatay\_ayni\_rakam\_say==0 and yan\_yana\_bosluk==0 and dikey\_ayni\_rakam\_say==0 and ust\_uste\_bosluk==0:  
 butunluk\_kontrolu(ters\_liste,adres\_listesi,satir\_say,0,0)  
 if ust\_uste\_bosluk>0 or yan\_yana\_bosluk>0 or dikey\_ayni\_rakam\_say>0 or yatay\_ayni\_rakam\_say>0:  
 degisiklik(liste,sifirlama\_listesi,ters\_liste,satir\_say)  
 elif len(adres\_listesi)!=rakam\_say:  
 print(len((adres\_listesi)))  
 degisiklik(liste, sifirlama\_listesi, ters\_liste, satir\_say)  
 else:  
 print("helal beee! bitti!!")  
 tablo\_ciz(liste,satir\_say)  
  
liste\_al()

**3. Programcı Kataloğu**

**Analiz** için ayırdığımız süre yaklaşık olarak 30 dakikadır. Bu aşamada problemde, programcıdan tam olarak ne istendiği belirlenmiş ve bu problemin çözümü için nasıl bir yol kullanılacağı belirlenmiştir.

**Tasarım** aşamasına ise yaklaşık 3 saat 30 dakika gibi bir süre ayrılmıştır. Bu aşamada ise programın pseudocode'u bir kağıda yazılmıştır.

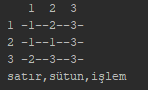
**Gerçekleştirim** aşamasına ise yaklaşık 8 saat 45 dakika gibi bir süre ayrılmıştır. Bu aşamada kağıda taslağını oluşturduğumuz pseudocode'un programlaştırılmış halini bilgisayara geçirip, onun üzerinde düzenlemeler yapılmıştır. Yazım hataları, input'ların düzenlenmesi gibi.

**Test** aşaması ise yaklaşık olarak 1 saat 50 dakika gibi bir sürede bitmiştir. Bu aşamada kodun doğru çalışıp çalışmadığı ve istenen her çıktının elde edilip edilmediği kontrol edilmiştir.

**Raporlama** aşamasına da yaklaşık 2 saat 45 dakika ayrılmıştır. Bu aşama programın yazılması ve test edilmesi gibi tüm aşamalar sırasında elde edilen veriler eşliğinde,alınan notlar ile oluşturulmuştur.

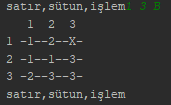
**4. Kullanıcı Kataloğu**

**a)** Programı çalıştırdığımızda karşımıza (oyunun zorluk derecesine göre) ilk olarak bu ekran geliyor.

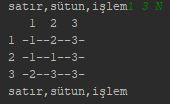
 Bu gördüğünüz en kolay seviyedeki oyunumuz.

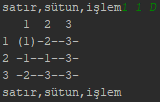
Kullanıcı bu aşamada işlem yapmak istediği satır, sütün , işlem sıralaması şeklinde girmelidir.

**b)** Kullanıcının girdisine göre aşağıdaki durumlar oluşmaktadır:

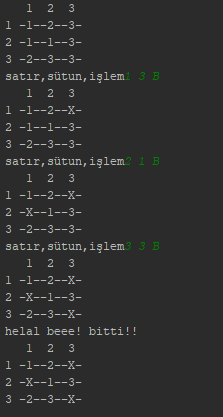


Bu 1. satır 3. sütundaki sayıyı Boş(B) haline getirmektedir.

 Bu işlemin olmasının amacı kullanıcı eğer girdisini eski hale getirmek istiyorsa Normal (N) ile hangi satır-sütündaki sayıyı değiştirmek istiyorsa onu girmelidir.

Bu işlemde de kullanıcı eğer o sayıda bir değişiklik yapmayacağına karar verdiyse kendine hatırlatma olarak kullanması için bulunmaktadır. Dolu(D) ile belirtilmiştir.

**c)** Son olarak da örnek bir bitmiş en kolay oyun çıktısı:



Kullanıcının girebileceği ve giremeyeceği değerler, oyunun bitme durumu gibi koşullar **Kısıtlamalar** bölümünde belirtilmiştir.

**4.1 . Kısıtlamalar**

* Kullanıcının satır, sütün ve işlem ile ilgili girmesi gereken verileri doğru girmesi gerekmektedir. Örneğin eğer 3x3 lük bir oyun ise sütuna veya satıra 3'ten büyük 1'den küçük bir değer girmesi bir şey ifade etmeyecektir. Diğer bir durum ise satır, sütun kısmına harf girilmesi hata ile karşılaşılmasına sebep olacaktır.
* Oyunun bitebilmesi için kuralların tamamının sağlanmış olması gerekmektedir. Genellikle 3. kural ,yani: **Dolu karelerin hepsi, yatay ve/veya dikey olarak birbirine bağlı olmalıdır (Dolu karelerin hepsi, tek bir (yatay ve/veya dikey) bağlantılı grup oluşturmalıdır).** , gözden kaçmaktadır. Lütfen kontrollerinizi doğru yapınız :).
* Kullanıcıdan istenen verilerin hiçbirinin boş bırakılmaması gerekmektedir. Aksi takdirde hata ile karşılaşılacaktır.