# VERİ YAPILARI
# VE
# ALGORİTMALAR

## FİNAL PROJESİ

Aydın Kağan Türkoğlu

18011060

Notlar:

1. TXT dosyasının sonunda gereksiz enter **yapılmadığından** lütfen emin olun yani son satır labirentin en alt duvarı olacak boş olmayacak. Ayrıca satır sonlarında gereksiz space olmamasına da dikkat edin lütfen.

2. Ben denemelerimi yolları 1 birim genişliğinde olan labirentler ile yaptım ama 2 birimlik labirentte de sorunsuz çalışıyor. Sadece 2 birim labirentte dikey çıkmaz sokaklarda teknik olarak gidebileceği bir yer olduğundan oralarda negatif puan yazmıyor. (1 birim labirentte sorun yok.) onun dışında yatay çıkmaz sokaklarda ikisinde de sorunsuz çalışıyor. (Bitişe ulaşma konusunda iki türde de sorun yok)

3. Pdf'te verilen örneğin 1 birimlik versiyonunda benim kod sadece 1 kere çıkmaz sokağa girdiğinden geri dönmeyi yapabildiği çok iyi anlaşılmıyor. Geri dönmeyi yapabildiğini görebilmeniz için 2 birim yollu labirenti veya benim kod sonuna bıraktığım kendi yaptığım labirenti kullanabilirsiniz

4. Adımlar arası bekleme süresini bekle fonksiyonunun ilk 2 satırını değiştirerek ayarlayabilirsiniz (pc'den pc'ye fark edeceği için böyle yaptım) (default 200ms)

Video linki: https://youtu.be/ztbn_ihMOB4

PROBLEM

Bu projede ana amacımız verilen bir labirentte başlangıç noktasından bitiş noktasına gitmek. Bunu yaparken de çıkmaz sokaklar için negatif puan ile elma toplandığında pozitif puan vermektir. Hareket ederken yollardan gidilmelidir duvarlardan geçmemek gerekir. Her bir hareket sonrası gidilen yön ve puan bilgisi interaktif olarak gösterilmelidir. Ayrıca bitiş olmayan bir çıkmaz sokağa girilirse veya o bölgedeki tüm noktalar daha önce gezilmişse gidilebilecek bir yer bulunana kadar geri dönülmelidir. Son olarak hareketleri dfs yaklaşımı ile yapmalıyız yani gidilebilecek bir kare olduğunda o kareye gidip yeni işlemi o kare üzerinden hesaplamalıyız.

ÇÖZÜM

Labirenti dosyadan okumamız gerekmektedir bunu yaparken aynı anda kontrol yaparak hareketlerimizi hesaplayacağımız hareket matrisini de oluşturabiliriz. Aynı zamanda başlangıç noktasının belirlenmesi için "b" okunduğu zamanki satır sütun değerleri kayıt edilir. Labirent matrislere kayıt edildikten sonra labirentte dolaşma işlemleri için dfs fonksiyonu çağırılır. İlk önce başlangıç noktasından gidilebilecek her yöne bakılmaktadır. Bu yönler yukarı sol sağ veya aşağı olabilir. Dfs mantığı ile çözüm yapıldığından sırası ile ilk gidilebilecek yöne gidilir ve ekranda gidilen yön" kuzey, batı, doğu, güney" olarak gösterilir. Hareket ederken her seferinde bulunduğumuz konuma bakarak puanımız hesaplanır ve puan ekrana yazdırılır. Puanlama yaparken mevcut konumda Elma varsa 10 puan eklenir eğer çıkmaz sokağa girildiyse -5 puan eklenir. Çıkmaz sokakta olduğumuzu anlamak için çevremize bakmak yeterlidir etrafımızda 3 veya daha fazla duvar varsa çıkmaz sokakta olduğumuz anlaşılır. Her hareket ettiğimizde daha önce gezdiğimiz yerleri yazdırabilmek için bir iz bırakılır. Gittiğimiz konumdan başka gidilebilecek bir yer yoksa fonksiyon kendini geri çağırmaz ve bir önceki konuma dönmüş oluruz. Geri dönme işlemi sonrası ekranda "geri dönülüyor" yazısı belirir. Ayrıca önceden bıraktığımız iz yeni tip özel bir iz ile değiştirilir. Bu izin amacı hem yazdırma yaparken gezildi olarak gösterilmesini engellemek hem de o konuma bir daha gidilmesini engellemektir. Dfs kendini bitiş noktasını bulana kadar çağırır bitiş noktası bulunduğunda son puan durumu ile bitişe ulaşıldığı bilgisi ekranda gösterilir.

ALGORİTMA

Bu kısımda ilk önce daha küçük fonksiyonları ardından da ana fonksiyonumuz olan DFS fonksiyonumuzu açıklayacağım.

void bekle(int final): adımlar arasın ne kadar beklenileceğini değiştirmeyi sağlar

int satirsay(char *dosyaadi,int* sutun) : txt dosyasındaki satır ve sütun sayısını daha sonra labirentleri matrise dinamik olarak kayıt yaparken kolaylık sağlanması açısından hesaplar.

void Dosyaoku(char *dosyaadi,char **matris,int* baslangicSatiri,

int* baslangicSutunu, int **matrisHareket) : Dosyadan teker teker semboller okunur bu semboller matris'e atanırken aynı zamanda sembole göre karşılığı olan sayı duvar 0 ,yol 1 ,bitiş 9 olarak matrisHareket'e kaydedilir. Bununla beraber "b" gördüğü yerde o noktanın konum bilgilerini de baslangicSatiri ve baslangicSutunu olarak kaydedilir.

void print_maze(int ToplamSatir, int ToplamSutun,char **maze,int **mazeHareket) : bu fonksiyon labirentimizi ekrana yazdırmaya yarar. Her hareket silinip sonrasında tekrar çağırılarak interaktiflik sağlanılır. Mevcut iterasyon adımındaki konumda eğer hareket matrisimizde gezildi anlamına gelen "2" değeri bulunmuyorsa orijinal labirentteki sembolü yazdırır. Eğer hareket matrisimizde "2" değeri varsa o sembol yerine " * " yazdırır.

void print_score_gidilenYon(int SCORE, int yon, int final) : gidilen yöne göre "KUZEY, BATI, DOĞU, GÜNEY) olarak yön bilgisi ile mevcut toplam skor yazdırılır. Bu fonksiyon da interaktifliği sağlamak için her adımda tekrar silinip çağırılır. Ayrıca final durumuna ulaşıldıysa buna özel bir yazdırma yapılır.

int score_hesapla(int satir, int sutun, int **mazeHareket,int SCORE,char **maze) : topladığımız puanları mevcut skor olarak tutmamızı sağlar. Her bir adım sonrası mevcut konumu değerlendirerek gerekirse puanı günceller. İlk önce bulunduğumuz konumun etrafındaki 4 nokta incelenir ve kaç tanesinin duvar olduğu hesaplanır. Eğer bulunduğumuz konumun 3 tarafı duvarlarla kaplı ise orası çıkmaz sokaktır ve skora -5 puan eklenir. Benzer şekilde bulunduğumuz konumda elma anlamına gelen "O" tespit edilirse +10 puan yazılır, aynı zamanda elmayı haritadan siler. Sonunda ise güncellenen skoru döndürür.

int dfs(int satir, int sutun, int **mazeHareket, int toplamSatir,

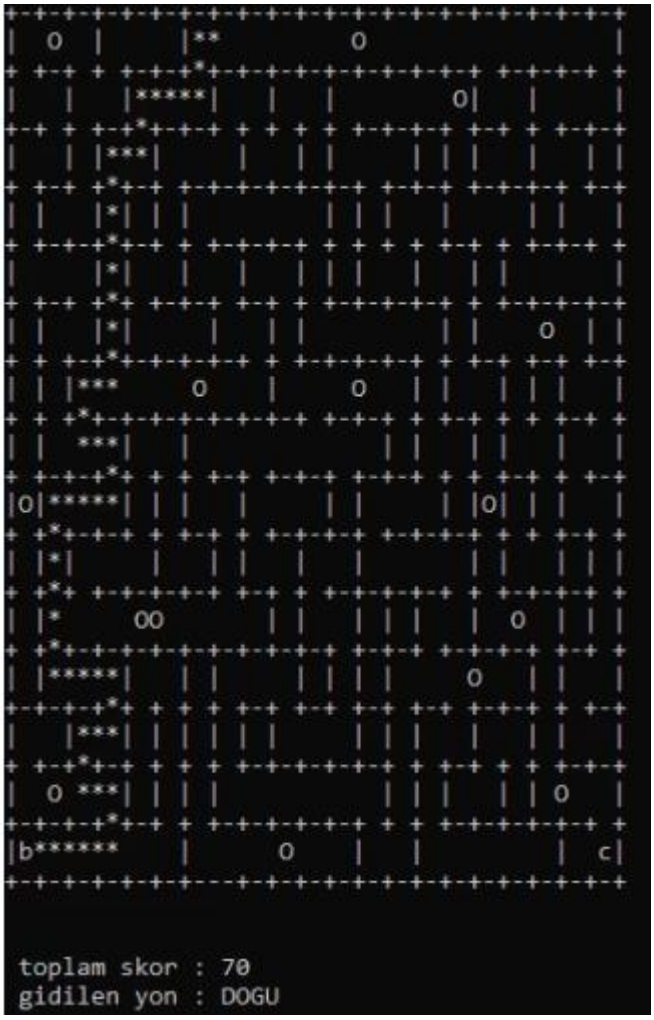
int toplamSutun, char **maze, int *SCORE, int *final) :

Labirentte her türlü hareketi dfs fonksiyonu ile yapmaktayız.

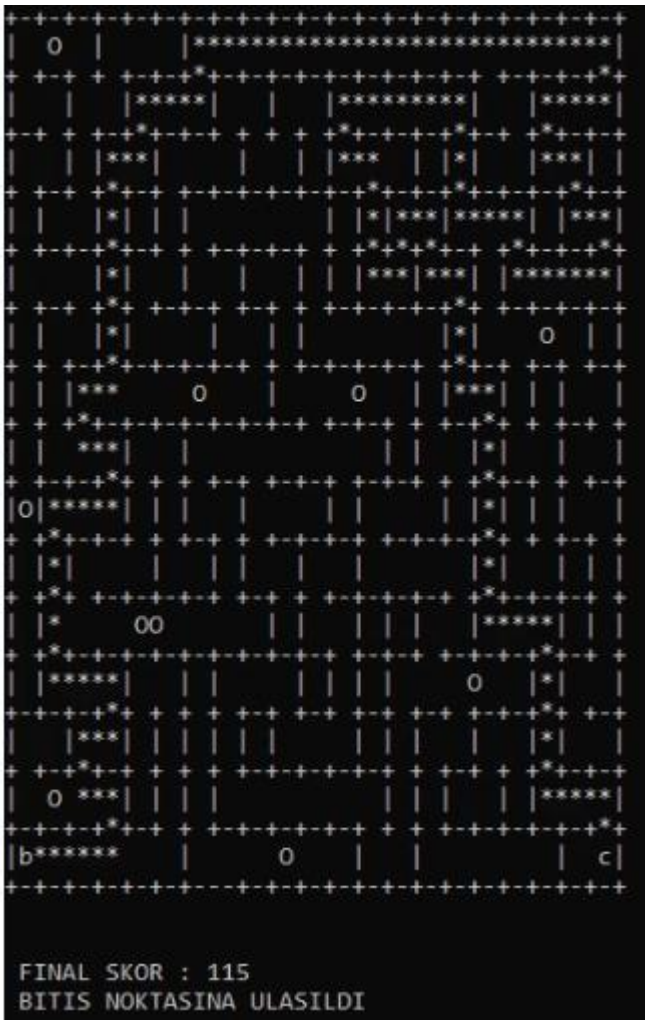
Dfs'yi özetlemek gerekirse bitişe ulaşılmayan her turda gidebileceği bir yer bulup kendini yeni konuma uygun değerler ile tekrar çağırması, gideceği bir yer bulamadığında ise kendini çağırmayarak bir önceki duruma geri dönmesidir.

Bu dfs fonksiyonunda da ilk önce final durumuna gelindi mi kontrolü yapılıyor final konumu "9" ile ayrıldığından konumumuz 9'a eşit olmadığı sürece fonksiyon devam etmektedir. Devam edilmesi durumunda sırası ile konumun çevresi kontrol edilir. Sırasıyla Yukarı sol sağ ve aşağısı olmak üzere kareler kontrol edilir eğer bunlardan birisi 1'e eşit ise orası gidilebilir anlamına gelir. Kontrolü sağlayan if'e girildikten sonra ilk olarak mevcut konumumuz gezildi anlamında "2" olarak işaretlenir. Daha sonra skor hesaplanır ve labirentin son durumu ekrana yazdırılır. Bu işlemler sonrasında yeni konuma gitmek üzere tekrar dfs çağırılır. Gidilebilecek yeni bir yer olduğu sürece bu işleme devam edilir. Eğer hiçbir yöne gidilemezse konum geri dönüldü olarak "3" ile işaretlenir. Bu ek işaretlemenin sebebi hem o karenin tekrar gezilmesine engel olmak hem de ekrana yazdırırken gezildi anlamındaki "*" yazdırılmasını geri almaktır. Daha sonra skor hesaplanır ve labirentin son durumu ekrana yazdırılır ancak son aşamada dfs çağırılmaz. Bu sayede fonksiyon yeni gidilebilir bir kare bulana kadar geri döner. Son duruma ulaşıldıysa başka bir işlem yapılmaz ve labirentin son durumu ile puanlar gösterilir.

Çalışma anından görüntüler


toplam skor : 70
gidilen yon : DOGU

bir genişliğindeki labirentin sonucu



İki genişliğindeki labirentin sonucu