DFA SIMÜLATÖRÜ

HAZIRLAYANLAR:

Faruk MUSA
22010903132

Umut Kuzey GÖRGEÇ
22010903075

Taha Melih ÖZTAŞ
B210109001

Doç. Dr. Zeynep GARİP
Arş. Gör. MUhammed Yusuf KÜÇÜKKARA

1. Projenin Tanımı ve Amacı

1.1 Projenin Tanımı :

DFA, bilgisayar bilimi ve diller teorisinde sıklıkla kullanılan bir modeldir. DFA'lar, düzenli dillerin tanınması, girdi dizilerinin kabul edilip edilmediğinin kontrolü ve alfabeye dayalı olarak durumlar arasında geçiş yapılması gibi çeşitli uygulamalara sahiptir. Bu projede, DFA simülatörü oluşturulmuş ve şu işlevsellikler sağlanmıştır:

- Kullanıcının DFA tanımı yapabilmesi,
- Katarların girilerek DFA tarafından kontrol edilmesi,
- DFA'nın grafiksel olarak görselleştirilmesi.

1.2 Projenin Amacı:

Projenin amacı, aşağıdaki temel hedefler üzerine odaklanır:

- Kullanıcı tarafından girilen DFA tanımını alma (durumlar, alfabe, geçiş fonksiyonları, başlangıç ve kabul durumları).
- Girilen katarların DFA tarafından kabul edilip edilmediğini kontrol etme.
- DFA'nın doğruluğunu ve etkinliğini test etmek için kullanıcı dostu bir arayüz sunma.
- DFA'nın görselleştirilmesiyle durumsal geçişlerin daha kolay anlaşılmasını sağlama.

2. Kullanılan Algoritmalar ve Araçlar

2.1 DFA Tanımı:

Bir DFA, aşağıdaki bileşenlerle tanımlanır:

- **Durumlar (Q):** Sonlu sayıda durumların kümesi.
- **Alfabe (Σ):** Dilin sembollerinin kümesi.
- **Geçiş Fonksiyonu (δ):** Bir durumdan başka bir duruma geçişi tanımlayan fonksiyon.
- **Başlangıç Durumu (q0):** DFA'nın başladığı durum.
- **Kabul Durumları (F):** Girdi dizisi işleminden sonra DFA'nın bu durumlardan birinde olması kabul anlamına gelir.

2.2 Algoritma :

DFA simülatörünün çalışma mantığı şu adımlardan oluşur:

1. **Kullanıcının DFA'yı tanımlaması:**
    - Durumlar, alfabe, geçiş fonksiyonları, başlangıç ve kabul durumları kullanıcıdan alınır.
    - Geçiş fonksiyonları, kullanıcıdan adım adım istenir ve eksik geçişler tamamlanır.
2. **DFA'nın girdi dizisini işlemesi:**
    - DFA, başlangıç durumundan başlar ve girdi dizisindeki sembolleri sırasıyla işler.
    - Her sembol için, geçiş fonksiyonuna göre yeni bir duruma geçiş yapılır.
    - Girdi dizisi tamamlandığında son durum kontrol edilir:
        - Eğer son durum kabul durumlarından biriyse, dizinin kabul edildiği sonucuna varılır.
        - Aksi takdirde, dizi reddedilir.

3. **Son Durumun Kontrol Edilmesi:**
    - Girdi dizisinin tüm sembolleri işlendiğinde elde edilen son durumun kabul durumu olup olmadığı kontrol edilir.
    - Sonuç, kullanıcıya iletilir.

4. **Görselleştirme:**
    - DFA'nın durumları ve geçişleri bir grafik olarak çizilir.
    - Graphviz kütüphanesi kullanılarak oluşturulan grafik, DFA'nın görselleştirilmesini sağlar.

2.3 Kullanılan Araçlar

- **Python Sınıfı (Class):** DFA'yı temsilen bir sınıf kullanılmıştır.
- **Geçiş Fonksiyonları:** Sözlük (Dictionary) veri yapısı ile geçişler saklanmıştır.
- **Graphviz Kütüphanesi:** DFA'yı görselleştirmek için kullanılmıştır.
- **OS Kütüphanesi:** Ekranı temizlemek için kullanılmıştır.

3. Bir DFA Örneği

İlk önce DFA'nın tanımı yapılıyor.

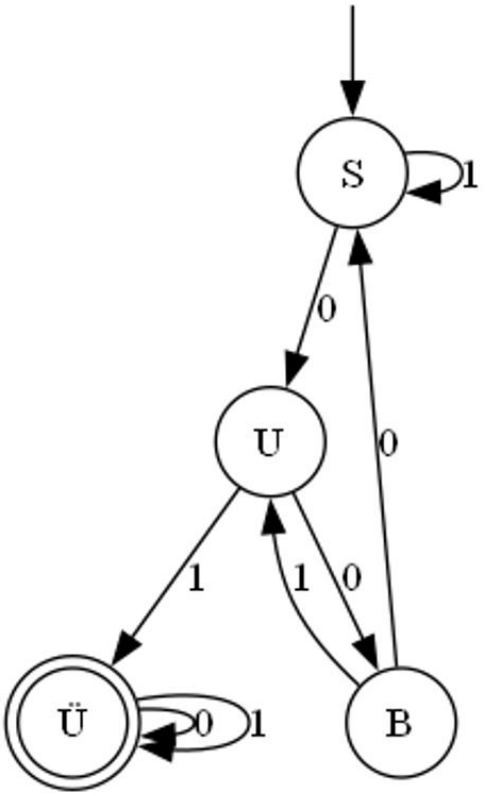
```
DFA tanımlama
Durumlar (virgülle ayrılmış): S,U,B,Ü
Alfabe sembolleri (virgülle ayrılmış): 0,1
Başlangıç durumu: S
Kabul durumları (virgülle ayrılmış): Ü
Geçişleri tanımlayın [format: (mevcut_durum) (sembol) (yeni_durum)]. 'bitti' yazarak çıkabilirsiniz.
Geçiş: S 0 U
Geçiş: S 1 S
Geçiş: U 0 B
Geçiş: U 1 Ü
Geçiş: B 0 S
Geçiş: B 1 U
Geçiş: Ü 0 Ü
Geçiş: Ü 1 Ü
Geçiş: bitti
```

Tanım yapıldıktan sonra menüden görüntüsünü veya istenilen bir kataın denemesi yapılabiliyor.

```
Seçenekler:
1 - Girdi kontrol et
2 - DFA'yı görselleştir
3 - Çıkış
Seçiminiz:
```

```
Dosya adı (varsayılan 'dfa'):
Görselleştirme oluşturuldu: dfa.png

Devam etmek için Enter'a basın...
```



İstenilen şekilde görüntüyü oluşturup çıktı alıyoruz.

```
Girdi dizisini girin: 1101
Başlangıç durumu: S
Yeni durum: S
Yeni durum: S
Yeni durum: U
Yeni durum: Ü
Girdi kabul edildi.

Devam etmek için Enter'a basın...
```

Verilen girdinin kabul durumuna ulaşıp ulaşmadığını kontrol edip kabul edilip edilmediğine karar veriyoruz.

4. Kaynakça

"FSM Graphviz Gallery": Graphviz: https://graphviz.org/Gallery/directed/fsm.html

"DFAs and Graphviz DOT". Jamie Wong Blog: https://jamie-wong.com/2010/10/16/dfas-and-graphviz-dot/.

Automata Theory: Deterministic Finite Automata (DFA) - Formal Language and Automata Theory: Resources: https://www.geeksforgeeks.org/designing-deterministic-finite-automata-set-1