Bölüm 5: Dizgi Algoritmaları

Algoritmalar

Dizgi Algoritmaları

- Metinlerle dolu bir dünyada yaşıyoruz.
- E-postalar, mesajlar, sosyal medya paylaşımları, haber metinleri...
- Bilgisayarlarımızda her gün sayısız metinle karşılaşıyoruz.
- Peki, bu metinler nasıl düzenlenir ve analiz edilir?
- Dizgi (String) algoritmaları,
  - metinlerde arama,
  - değiştirme,
  - karşılaştırma gibi işlemleri gerçekleştirir.

Dizgi Eşleştirme Algoritmaları

- Brute Force (Kaba Kuvvet):
  - Metindeki her konum örüntü ile eşleştirmek için kontrol edilir.
  - Maksimum sayıda karşılaştırma gerektirebilir.
- Knuth-Morris-Pratt (KMP)
  - Başlangıçta tablo oluşturularak arama süresi azaltılır,
  - Karakter karşılaştırmalarını azaltarak hızlı çalışır.
- Boyer-Moore
  - Uzun aramalarda etkili. Kök bulma ve kaydırma stratejisi kullanır.
- Rabin-Karp Algoritması
  - Olasılıksal bir algoritma. Hashing kullanır.

Dizgi Sıkıştırma Algoritmaları

- Sıralı Sıkıştırma Kodlaması (Run Length Encoding)
  - Aynı veri değerleri tek bir değer ve sayı olarak saklanır.
  - Tekrar eden değerler yerine tekrar eden veri sayısı saklanır.
- Lempel-Ziv-Welch (LZW)
  - GIF gibi formatlarda kullanılan sözlük tabanlı sıkıştırma algoritması.
  - Tekrar eden örüntüleri sözlük oluşturarak kısa sembollerle temsil eder.
  - Dinamik bir sözlük kullanarak sıkıştırma sağlar.

Dizgi Sıralama Algoritmaları

- Sözlüksel Sıralama (Lexicographic Order)
  - Dizgiler, alfabetik sıraya benzer sıralanır.
  - Her karakterin ASCII değeri karşılaştırılarak sıralama yapılır.
- Radix Sıralama
  - Karşılaştırmalı olmayan bir tam sayı sıralama algoritmasıdır.
  - Veriler tamsayı anahtarlarına sahiptir.
  - Aynı konumda aynı değeri paylaşan verileri gruplandırarak sıralar.
  - Her basamak için ayrı ayrı işlem yapılır.

Dizgi Düzenleme Mesafesi Algoritmaları

- Levenshtein Mesafesi
  - İki dizgi arasındaki benzerliği ölçen bir metrik,
  - Bir dizgiden diğerine dönüştürmek için gereken minimum tek karakterli düzenleme sayısı olarak tanımlanır.
- En Uzun Ortak Alt Dizi (Longest Common Subsequence - LCS)
  - İki dizginin ortak olan en uzun alt dizisi,
  - Karakterlerin sıralı olmasını gerektirmez, ancak sıra korunmalıdır.
  - Dizgiler arasındaki benzerlik veya farkı belirlemek için kullanılır.

Dizgi Dönüşüm Algoritmaları

- Sonek Dizisi (Suffix Array)
  - Bir dizginin tüm son eklerinin bir dizisi.
  - Dizgi içindeki alt dizgilerin bir temsili olarak kullanılır.
- Burrows-Wheeler Dönüşümü (BWT)
  - Bir dizginin tersine dönüştürülmesiyle elde edilen yeni bir form,
  - Bzip2 gibi sıkıştırma algoritmaları için önişlem adımı olarak kullanılır.

Dizgi Ayrıştırma (Parsing) Algoritmaları

- Düzenli İfadeler (Regular Expressions)
  - Bir arama örüntüsünü tanımlayan karakter dizisi,
  - Belirli bir örüntüye uyan tüm dizgileri bulmak için kullanılır
- Sonlu Durum Makineleri (Finite State Machines - FSM)
  - Dizgi içindeki örüntüleri tanımak için kullanılan hesaplama modelleri,
  - Belirli bir girdi dizisindeki geçişlerin durumlarını izleyen bir otomat,
  - Karmaşık ayrıştırma ve analiz işlemlerinde kullanılır.

Düzenli İfadeler (Regular Expressions)

- Düzenli ifadeler, metinlerde belirli örüntüleri tanımlamak ve bu örüntülere uyan kısımları eşleştirmek için kullanılır.
- Metin manipülasyonlarını gerçekleştirmek için yaygın kullanılır.
- Karmaşık metin işlemlerini otomatikleştirir.

Temel Operatörler ve Kavramlar

- .: Herhangi bir tek karakteri temsil eder.
- *: Önceki karakterin sıfır veya daha fazla tekrarını temsil eder.
- +: Önceki karakterin bir veya daha fazla tekrarını temsil eder.
- ?: Önceki karakterin sıfır veya bir kez tekrarını temsil eder.
- [ ]: Belirli karakterlerin bir kümesini temsil eder.
- ^: Belirtilen örüntünün dizginin başında olmasını sağlar.
- $: Belirtilen örüntünün dizginin sonunda olmasını sağlar.
- ( ): Örüntülerin gruplandırılmasını sağlar ve alt ifadeleri tanımlar.

Karmaşık Operatörler ve Kavramlar

- { }: Belirli bir sayıda tekrarın belirtilmesini sağlar.
- |: Alternatif örüntüler arasında seçim yapmayı sağlar.
- \b: Kelimenin başı veya sonu gibi belirli sınırları tanımlar.
- \d: Bir rakamı tanımlar.
- \w: Bir kelimeyi tanımlar.
- \s: Boşluk karakterini tanımlar.

Düzenli İfadelerin Temelleri

- Karakter ve Metin Eşleşmesi:
  - a: Tek bir 'a' karakteriyle eşleşir.
  - abc: "abc" ile tam olarak eşleşir.
- Özel Karakterler:
  - .: Yeni satır hariç herhangi bir karakterle eşleşir.
  - \d: Bir rakam ile eşleşir (0-9).
  - \w: Bir kelime ile eşleşir (alfanumerik + alt çizgi).
  - \s: Bir boşluk ile eşleşir (boşluk, tab, yeni satır vb.).

Düzenli İfade Karakter Sınıfları

- Karakter Sınıfları:
  - [abc]: 'a', 'b' veya 'c' karakterlerinden biriyle eşleşir.
  - [a-z]: Küçük harflerden herhangi biriyle eşleşir.
  - [A-Z]: Büyük harflerden herhangi biriyle eşleşir.
  - [0-9]: Rakamlarla eşleşir.
- Tümleyen Karakter Sınıfları:
  - [^abc]: 'a', 'b' veya 'c' dışındaki herhangi bir karakterle eşleşir.
  - [^0-9]: Rakamlar dışındaki herhangi bir karakterle eşleşir.

Yinelenen Karakter ve Sayılar

- a*: Sıfır veya daha fazla 'a' karakteriyle eşleşir.
- a+: Bir veya daha fazla 'a' karakteriyle eşleşir.
- a?: Sıfır veya bir 'a' karakteriyle eşleşir.
- a{3}: Tam olarak üç 'a' karakteriyle eşleşir.
- a{2,4}: İki ile dört arasında 'a' karakteriyle eşleşir.

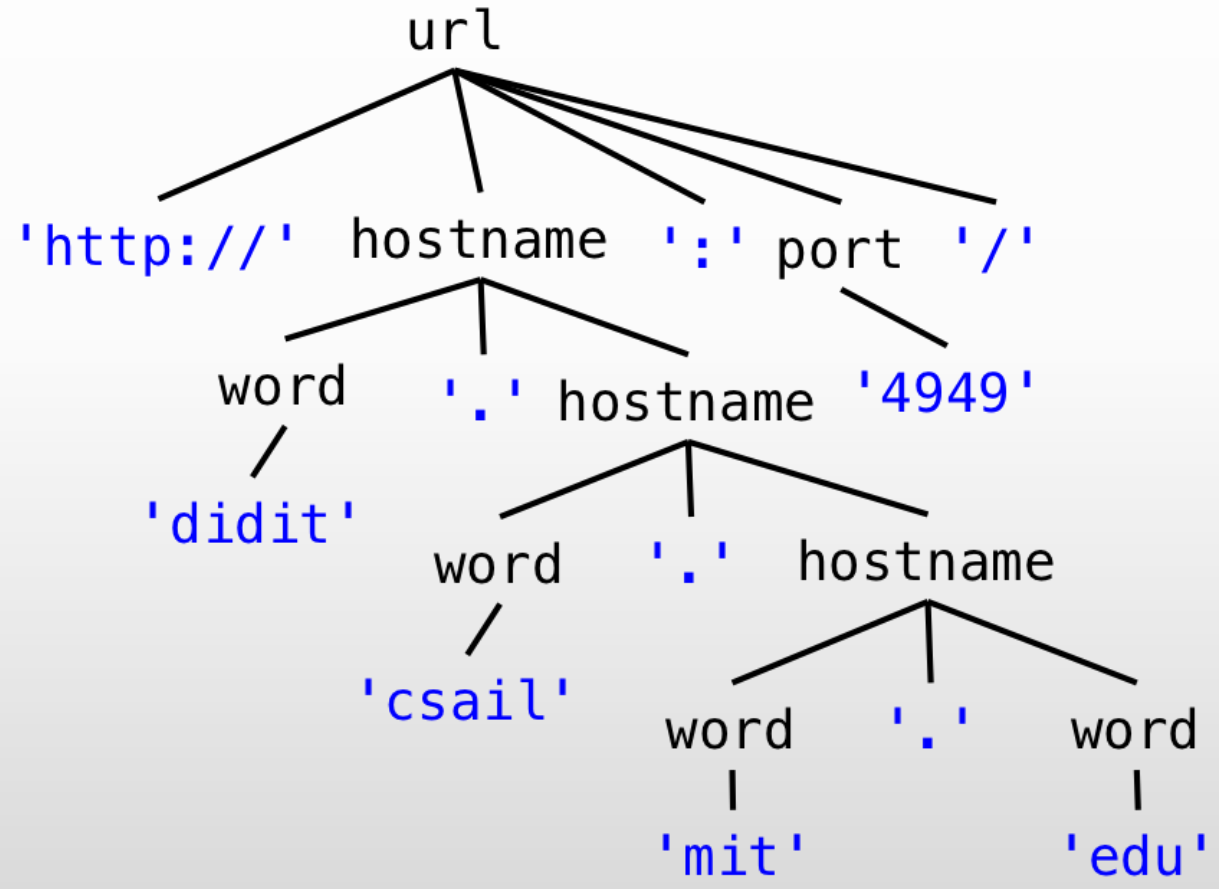
Gruplama

- (abc): "abc" ile tam olarak eşleşir ve gruplar.
- (a|b|c): 'a', 'b' veya 'c' karakterleriyle eşleşir ve gruplar.

Lookahead ve Lookbehind

- a(?=b): 'a' karakteri, 'b' karakterinden önce gelirse eşleşir.
- a(?!b): 'a' karakteri, 'b' karakterinden önce gelmezse eşleşir.
- (?<=b)a: 'a' karakteri, 'b' karakterinden sonra gelirse eşleşir.
- (?<!b)a: 'a' karakteri, 'b' karakterinden sonra gelmezse eşleşir.

Düzenli İfadeler (Regular Expressions)



Örnek

- E-posta Doğrulama:
  - `^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$`
- Telefon Numarası:
  - `^(\+\d{1,3}[- ]?)?\d{10}$`
- URL:
  - `^https?:\/\/[^\s/$.?#].[^\s]*$`

Sonlu Durum Makineleri (Finite State Machines)

- Bir dizgiyi işlemek için kullanılan matematiksel modeldir.
- Bir durum kümesi;
  - başlangıç durumu,
  - girdi alfabesi ve
  - durumlar arasındaki geçişlerin kümesinden oluşur.
- Basit ve anlaşılması kolay bir modeldir.
- Karmaşık dizgi analizi problemlerini ele alabilir.

Temel Bileşenler

- Durumlar (*States*):
  - Makinenin bulunabileceği farklı durumlar.
- Alfabe (*Alphabet*):
  - FSM'nin kabul ettiği giriş sembolleri kümesi.
- Başlangıç Durumu (*Start State*):
  - İşleme başlamak için seçilen durum.
- Geçişler (*Transitions*):
  - Girdiye göre durumlar arasındaki geçişler.
- Son Durum (*Final States*):
  - Dizgi işlendiğinde olunabilecek son durumlar.

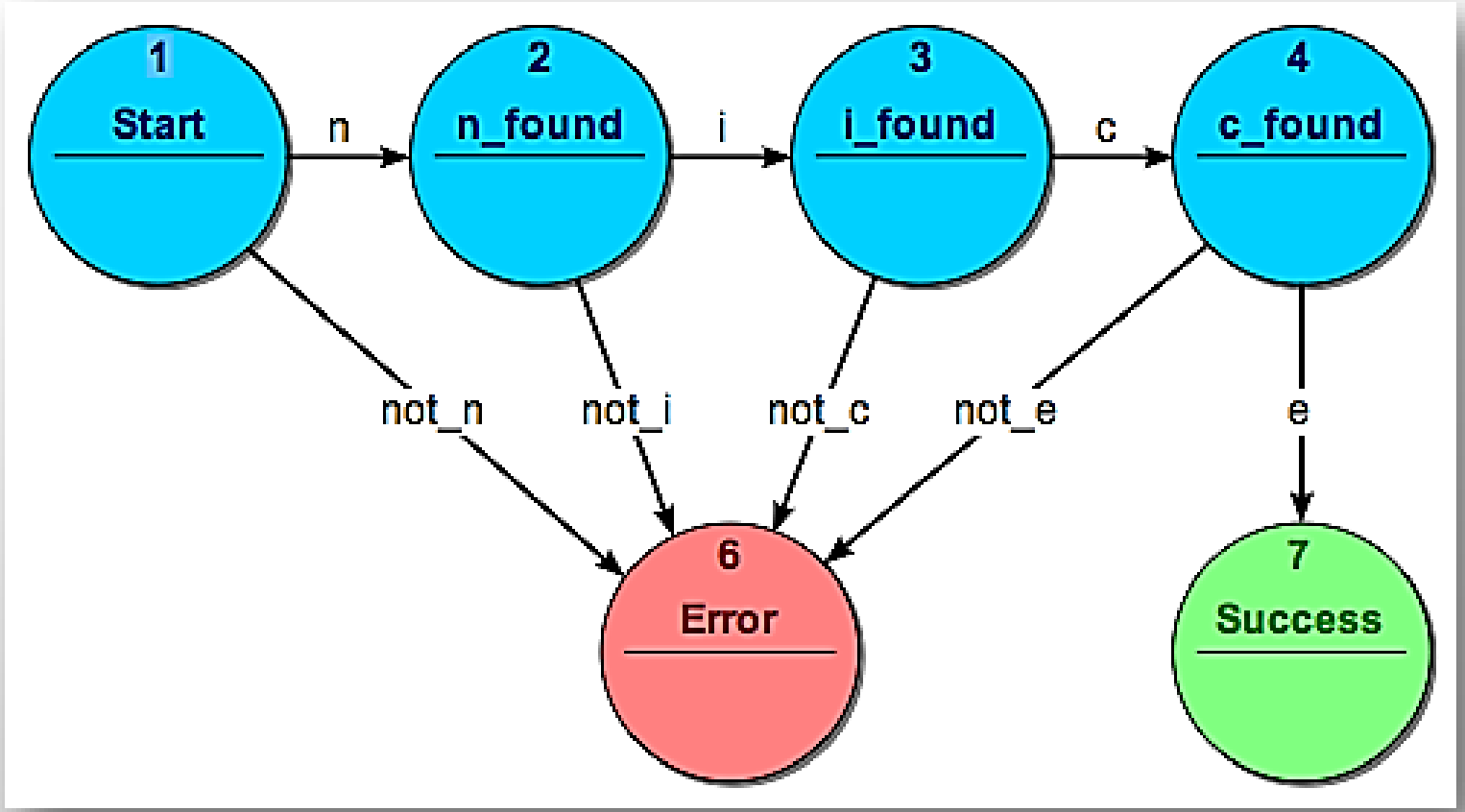
FSM Türleri

- *Deterministic Finite Automaton (DFA):*
  - Her durum ve giriş sembolü çifti için yalnızca bir geçiş tanımlıdır.
- *Non-Deterministic Finite Automaton (NFA):*
  - Bir durum ve giriş sembolü çifti için birden fazla geçiş tanımlıdır.
- *Epsilon-NFA (ε-NFA):*
  - Boş geçişlerin (epsilon geçişleri) mümkün olduğu NFA türü.

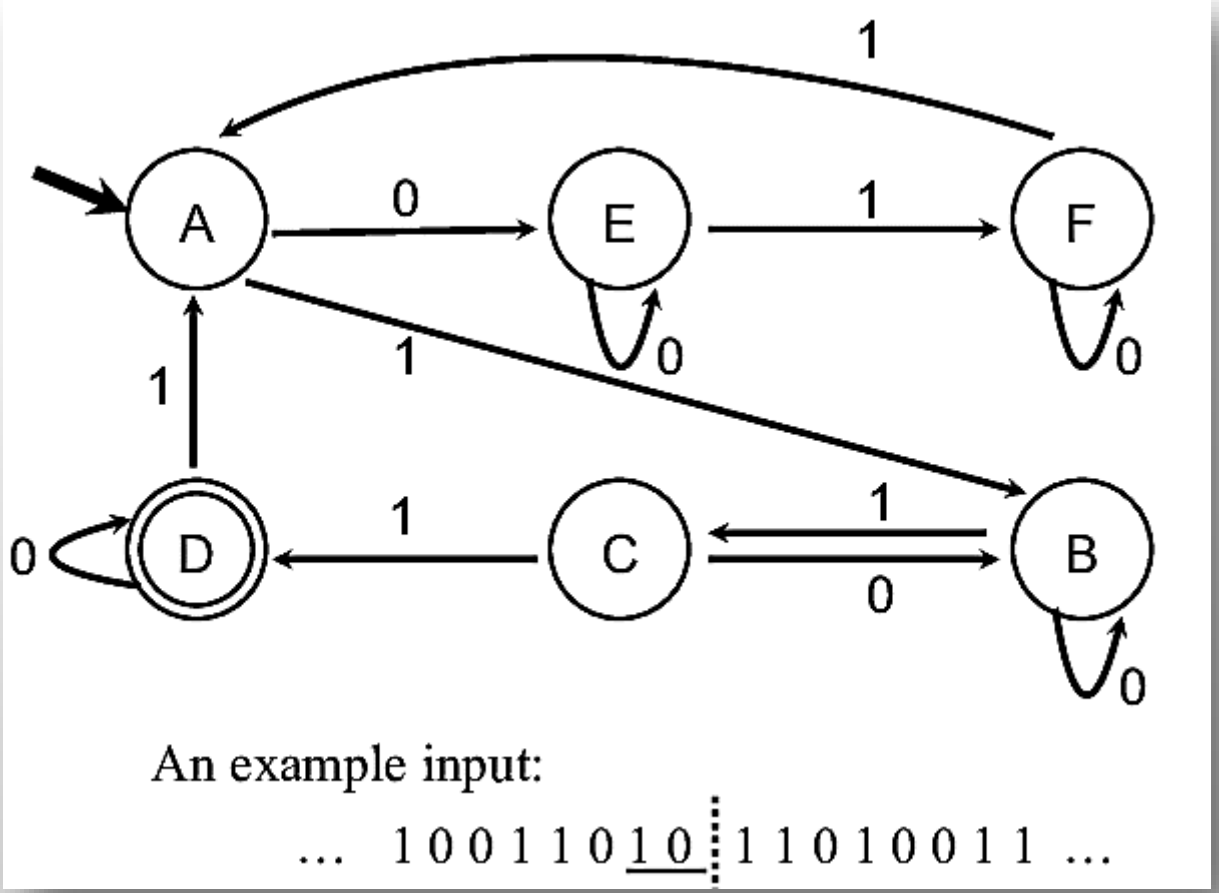
Kullanımı

- Başlangıç durumuna geçiş.
- Her bir karakter için durumların güncellenmesi.
- Dizgi tamamlandığında son durumun kontrol edilmesi.

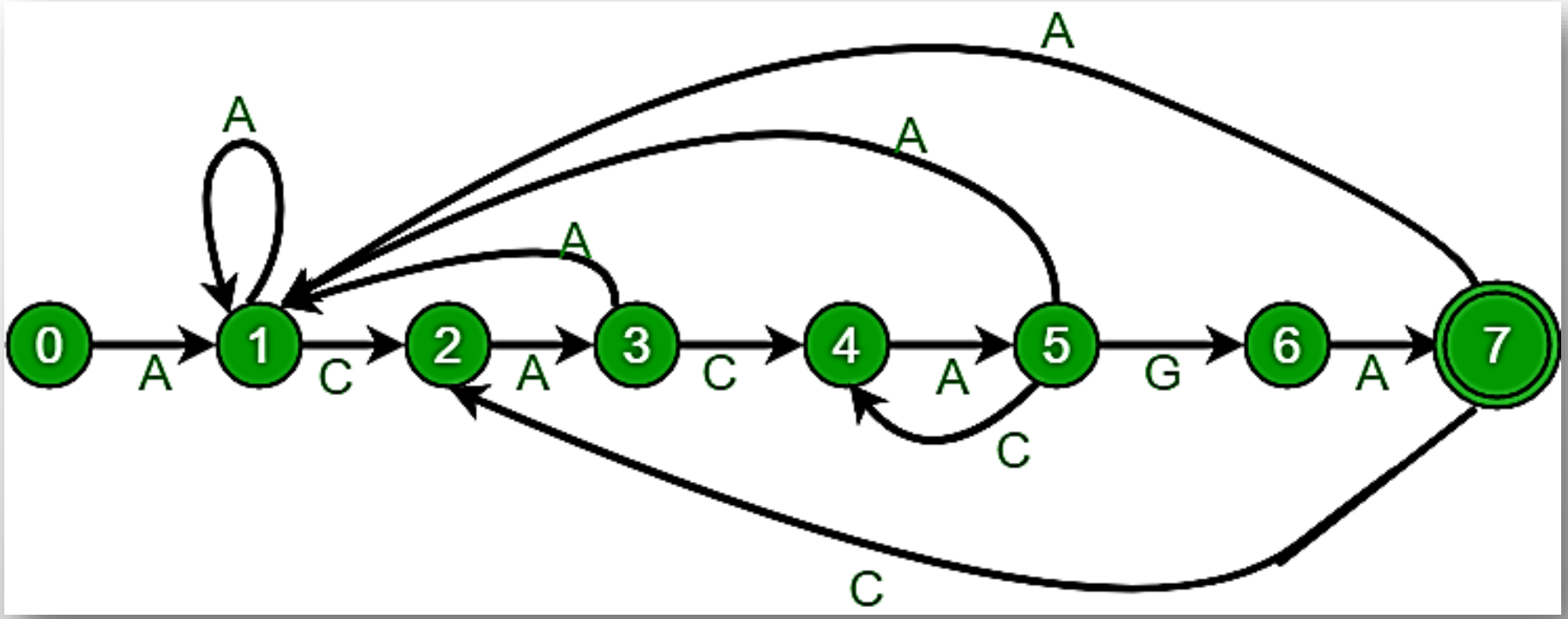
Sonlu Durum Makineleri



Sonlu Durum Makineleri



Sonlu Durum Makineleri



SON