Bölüm 5: Trie Veri Yapısı

Algoritmalar

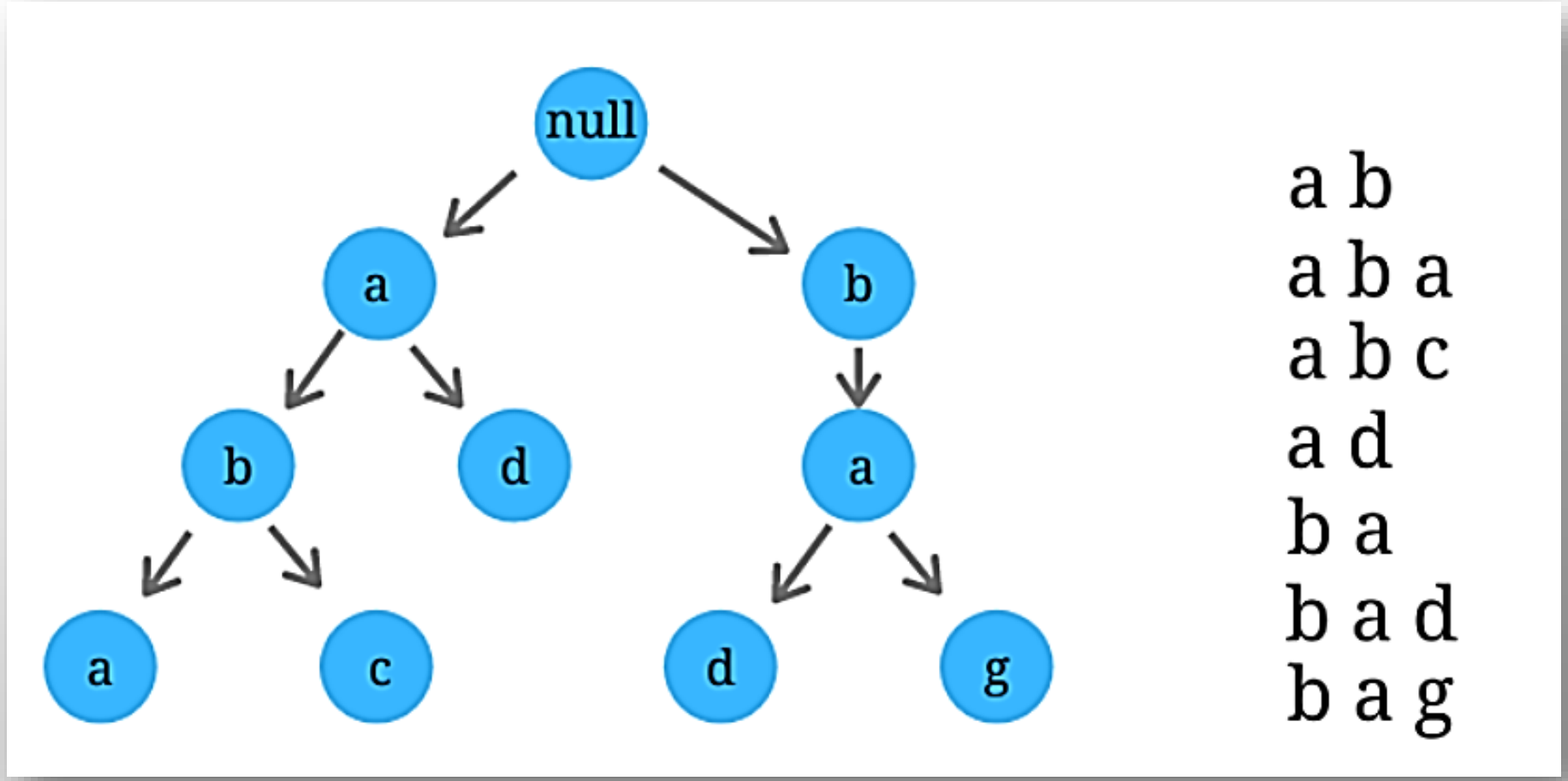
Trie

- Dizgi'leri (*String*) hızlı arama yapılabilecek şekilde saklar.
- Ağaç benzeri bir veri yapısı.
- Kullanım Alanları:
  - Otomatik tamamlama, IP yönlendirme, yazı denetimi.
- Kök düğüm boştur, her düğüm bir karakteri temsil eder.
- Kelimeler, kökten yapraklara giden yollarla saklanır.

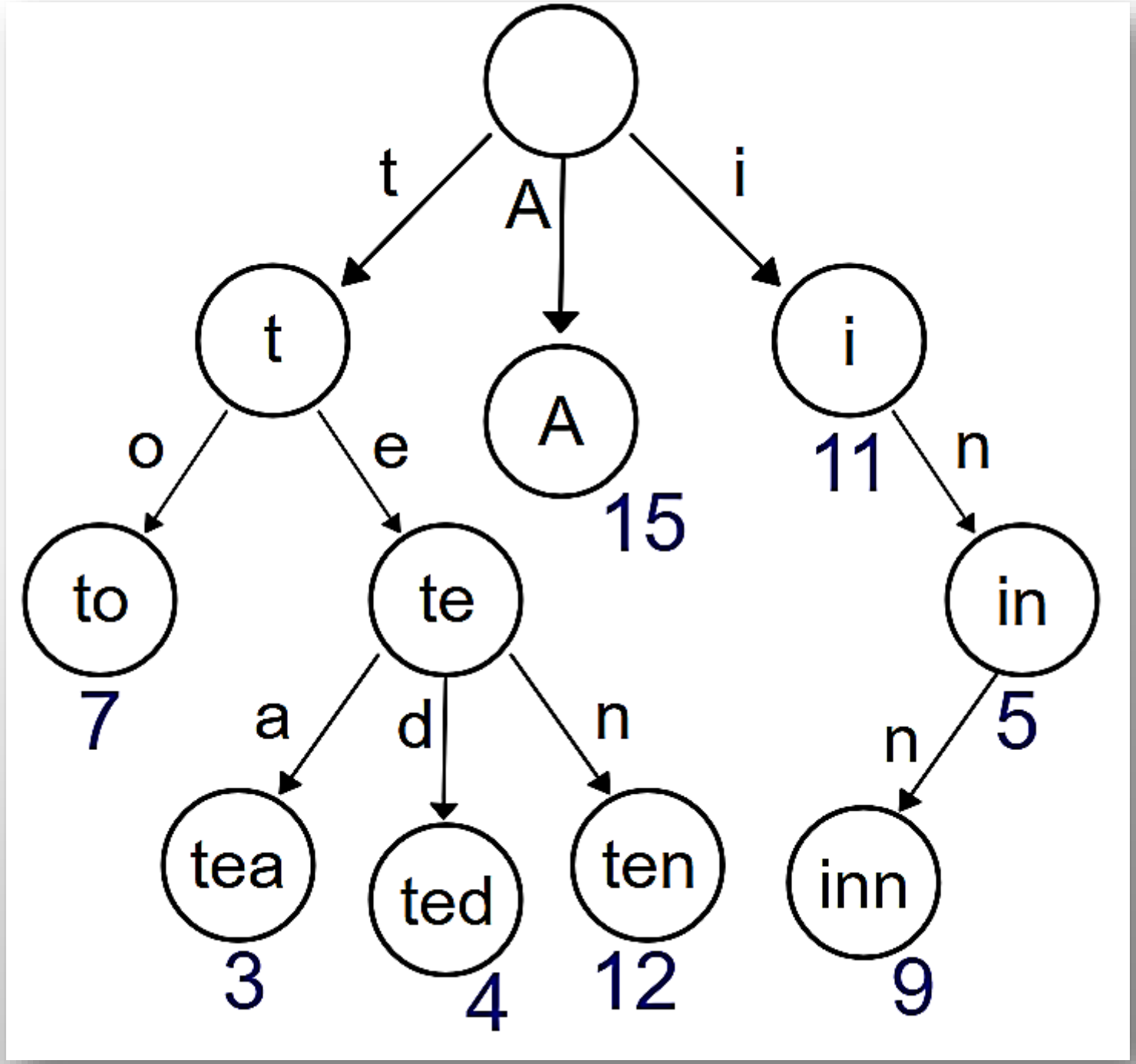
Temel Bileşenler

- **Kök (*Root*):** Boş bir düğüm.
- **Düğümler (*Nodes*):** Her düğüm bir karakter ve çocuk düğümler içerir.
- **Kenarlar (*Edges*):** Düğümler arası bağlantılar (*karakterler*).
- **İşaretçiler**: Kelime sonunu belirten işaretler (*is_end_of_word*).

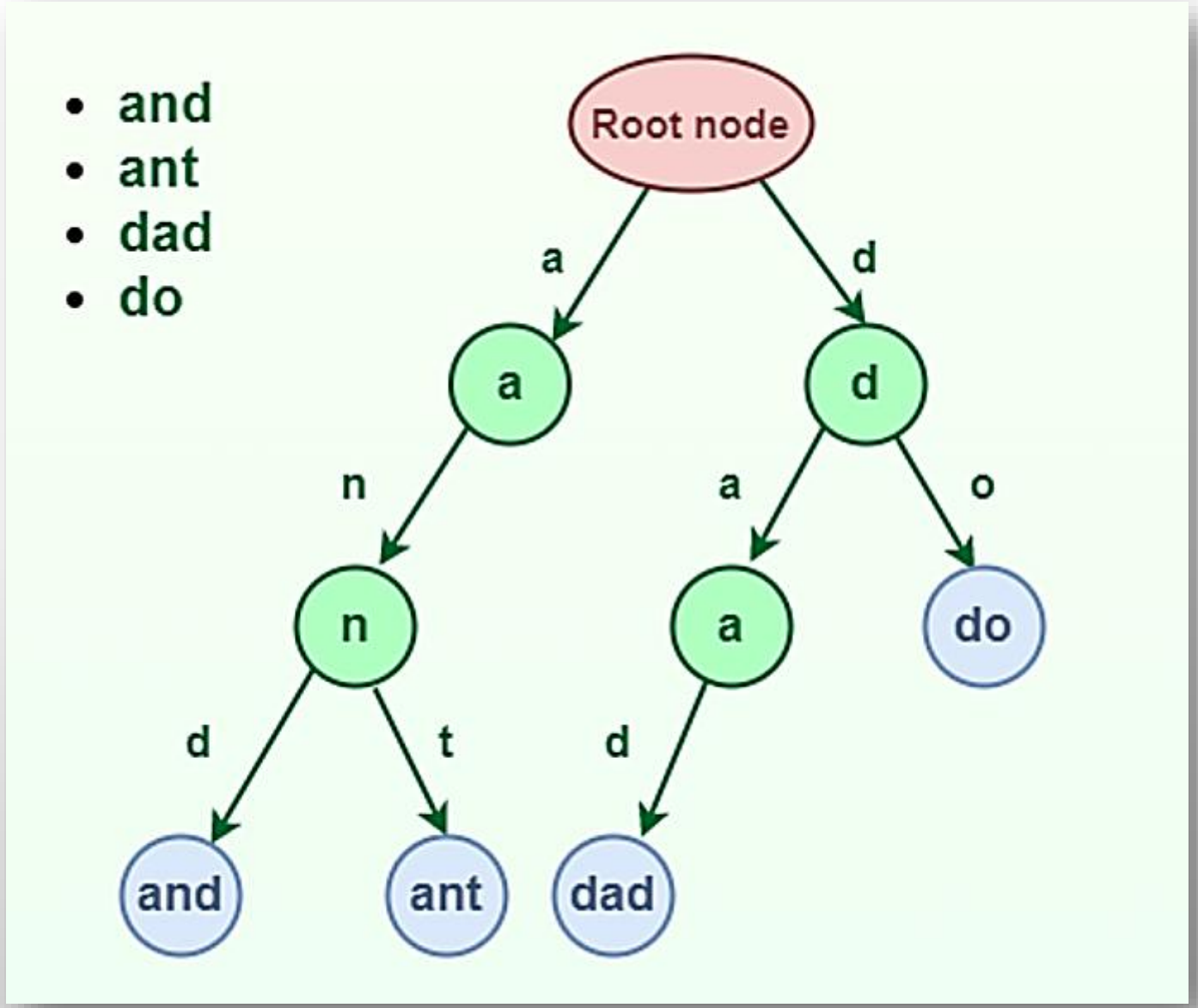
Trie



Trie



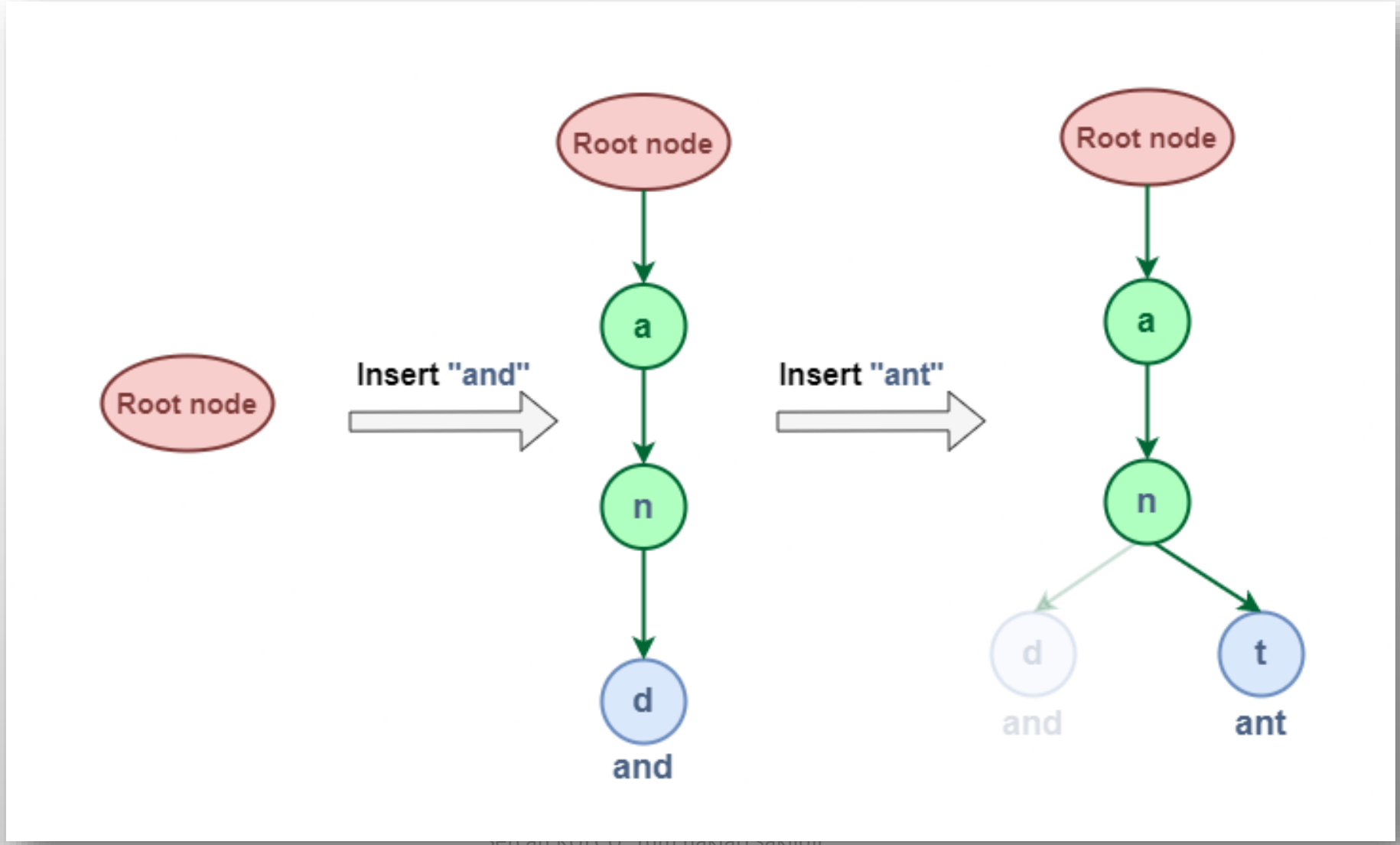
Trie



Trie Düğüm Sınıfı

```
public class TrieDugumu {

    TrieDugumu[] cocuk;

    boolean kelimeSonu;

    public TrieDugumu() {

        kelimeSonu = false;

        cocuk = new TrieDugumu[26];

    }

}
```

Ekleme



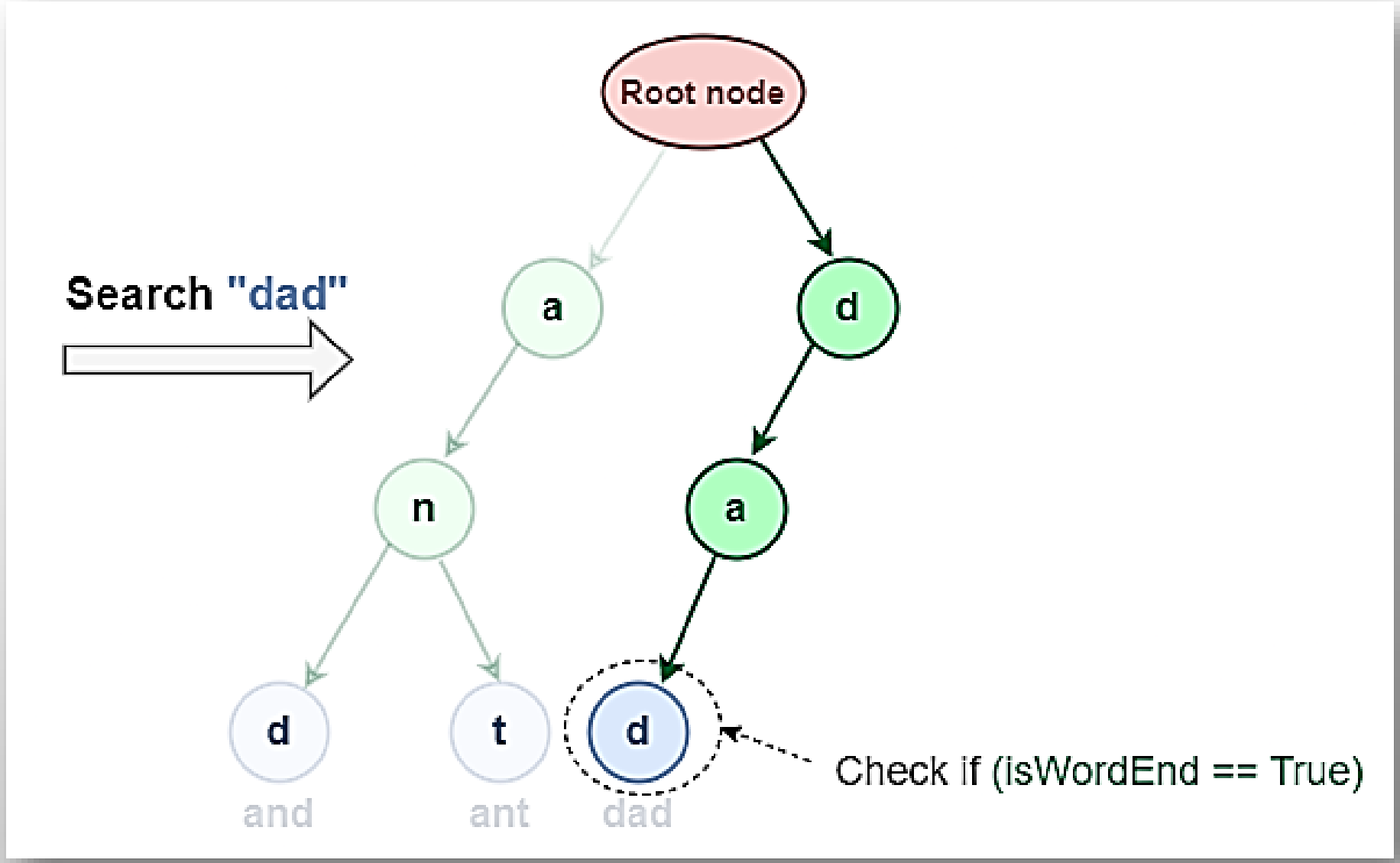
Trie Veri Yapısına “and” Ekleme

- Kök düğümden başlanır. Kök düğüm herhangi bir karakter içermez ve *kelimeSonu* değeri 0’dır.
- İlk karakter “a”: *cocuk[0]* kontrol edilir. Eğer *null* ise, *“a”* karakterini içeren, *kelimeSonu* değeri 0 olan yeni bir TrieDugumu oluşturulur.
- İkinci karakter “n”: *cocuk[13]* kontrol edilir. Eğer *null* ise, *“n”* karakterini içeren, *kelimeSonu* değeri 0 olan yeni bir TrieDugumu oluşturulur.
- Üçüncü karakter “d”: *cocuk[3]* kontrol edilir. Eğer *null* ise, *“d”* karakterini içeren, *kelimeSonu* değeri 1 olan yeni bir TrieDugumu oluşturulur.

Trie Veri Yapısına “ant” Ekleme

- Kök düğümden başlanır. Kök düğüm herhangi bir karakter içermez ancak eklenen her kelimenin ilk karakterini takip eder.
- İlk karakter “a”: *cocuk[0]* kontrol edilir. Önceki eklemeden dolayı *“a”* düğümü mevcut. Mevcut “a” düğümüne ilerlenir.
- İkinci karakter “n”: *cocuk[13]* kontrol edilir. “n” düğümü mevcut, var olan *“n”* düğümüne ilerlenir.
- Üçüncü karakter “t”: *cocuk[19]* kontrol edilir. Eğer *null* ise, *“t”* karakterini içeren, *kelimeSonu* değeri 1 olan yeni bir TrieDugumu oluşturulur.

Arama


Root node
Search "dad"
a
d
n
a
d
t
d
Check if (isWordEnd == True)
and
ant
dad

Trie Veri Yapısında Arama

- Kök düğümden (*başlangıç noktası*) başlanır.
- Aranan kelimenin karakterlerine göre Trie üzerinde ilerlenir. Her karakter için Trie içinde karşılık gelen dala (çocuğa) gidilir.
  - Eğer dal yoksa, kelime Trie içinde bulunmamaktadır.
- Kelimenin sonuna ulaşıldığında *kelimeSonu* kontrol edilir.
  - Eğer *kelimeSonu* 1 ise, kelime Trie içinde bulunmuştur.
  - *kelimeSonu* değeri 0 ise, kelime Trie içinde bulunmaz. (var olan bir kelimenin ön eki (prefix'i) olduğu anlamına gelir.)

Avantajlar ve Dezavantajlar

- Hızlı önek (*prefix*) tabanlı arama (*autocomplete*).
- Kelime arama *O(L)* zaman (*L: kelime uzunluğu*).
- Çakışan önekler bellek verimliliği sağlar.
- Yüksek bellek tüketimi (özellikle büyük alfabelerde).
- Bellek Karmaşıklığı: *O(N * M)* (*N: düğüm sayısı, M: alfabe boyutu*).

Soru

- Trie'de "apple" ve "app" varsa, "app" silindiğinde yapı nasıl değişir?
- **Cevap:**
  - Sadece son 'p' düğümünün *is_end_of_word* işareti kaldırılır.

SON