Zaman Karmaşıklığı (Time Complexity)

1. **`addnode` Fonksiyonu:**
    - Bu fonksiyon, yeni bir düğüm oluşturup onu listenin başına ekliyor. Her seferinde sabit bir zaman alır, yani O(1) zaman karmaşıklığına sahiptir.
    - Eğer bu fonksiyonu `n` defa çağırırsanız toplam zaman karmaşıklığı O(n) olacaktır.
2. **`group` Fonksiyonu:**
    - Dıştaki `while` döngüsü, listenin tüm düğümlerini gezer. Bu, en kötü durumda `n` n kez döner.
    - İçteki `while` döngüsü, aynı veri değerine sahip ardışık düğümleri saymak için kullanılır. Bu döngü, en kötü durumda tüm listenin tamamını gezebilir, yani bu durumda da `n` n kez çalışabilir.
    - Dolayısıyla, `group` fonksiyonunun toplam zaman karmaşıklığı O(n) olarak kabul edilebilir, çünkü her düğüm sadece bir kez işlenir.
3. **`print` Fonksiyonu:**
    - Bu fonksiyon, listeyi gezerek tüm düğümlerin verilerini yazdırır. Yine, en kötü durumda `n` n düğüm için O(n) zaman alır.

Toplam Zaman Karmaşıklığı

- Kodun genel akışında, `addnode`, `group` ve `print` fonksiyonları toplamda:O(n)+O(n)+O(n)=O(n) şeklinde zaman karmaşıklığına sahiptir.

Alan Karmaşıklığı (Space Complexity)

1. **Düğüm Belleği:**
    - Her `addnode` çağrısı yeni bir düğüm oluşturur ve bu düğümler listeye eklenir. Eğer listeniz `n` n düğüm içeriyorsa, bu düğümlerin her biri için dinamik bellek tahsis edilmiştir.
    - Dolayısıyla, toplam bellek kullanımı O(n) olacaktır.
2. **Geçici Değişkenler:**
    - `group` fonksiyonunda kullanılan geçici değişkenler (örneğin, `gezer`, `previous`, `newNode`, `next`) sabit sayıda değişkendir ve bunlar için ek bellek kullanılmaz. Bu nedenle, bu kısım sabit O(1) alan kullanır.

Toplam Alan Karmaşıklığı

- Kodun genelinde alan karmaşıklığı, dinamik olarak tahsis edilen düğümlerin sayısından kaynaklandığı için:O(n) olarak değerlendirilir.

Özet

- **Zaman Karmaşıklığı:** O(n)
- **Alan Karmaşıklığı:** O(n)