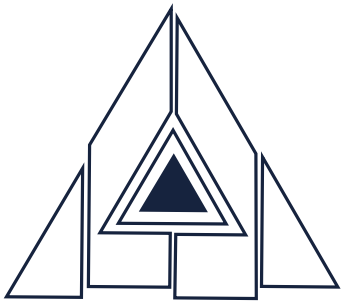

TURKSTUDENTCO

1. Arama Algoritmaları (Search Algorithms)

a. Doğrusal Arama (Linear Search)

Bir dizide hedef elemanı bulmak için sırayla her bir elemanı kontrol eder.

- **Zaman Karmaşıklığı**:
- **En İyi Durum (Best Case)**: O(1) (İlk eleman hedefse.)
- **Ortalama Durum (Average Case)**: O(n) (Hedef eleman dizinin ortasında veya sonunda olabilir.)
- **En Kötü Durum (Worst Case)**: O(n) (Hedef eleman dizinin sonunda veya dizide yok.)
- **Mekan Karmaşıklığı**: O(1) (Ekstra bellek kullanmaz.)

b. İkili Arama (Binary Search)

Dizi sıralı olduğunda ortadaki elemanı kontrol ederek aramayı bölerek devam ettirir.

- **Zaman Karmaşıklığı**:
- **En İyi Durum**: O(1) (İlk kontrol edilen eleman hedefse.)
- **Ortalama Durum**: O(log n) (Her adımda diziyi ikiye böler.)
- **En Kötü Durum**: O(log n)
- **Mekan Karmaşıklığı**: O(1) (Yineleme kullanılırsa), O(log n) (Özyinelemeli versiyonu için.)

2. Sıralama Algoritmaları (Sorting Algorithms)

a. Bubble Sort

Her elemanla yanındaki elemanı karşılaştırarak büyük olanları sona doğru kaydırır.

- **Zaman Karmaşıklığı**:
- **En İyi Durum**: O(n) (Dizi zaten sıralıysa.)
- **Ortalama Durum**: $O(n^2)$ (Her eleman diğerleriyle karşılaştırılır.)
- **En Kötü Durum**: $O(n^2)$ (Tamamen ters sıralı dizi.)
- **Mekan Karmaşıklığı**: O(1)

b. Merge Sort

Diziyi sürekli olarak ikiye böler, daha sonra bu alt dizileri sıralı olarak birleştirir.

- **Zaman Karmaşıklığı**:

- **En İyi, Ortalama ve En Kötü Durum**: O(n log n)
- **Mekan Karmaşıklığı**: O(n) (Ekstra bellek gerektirir.)

c. Quick Sort

Bir pivot eleman seçer ve pivotun soluna daha küçük, sağına daha büyük elemanları yerleştirir. Bu işlemi yineleyerek sıralama yapar.

- **Zaman Karmaşıklığı**:
- **En İyi ve Ortalama Durum**: O(n log n)
- **En Kötü Durum**: O(n^2) (Eğer pivot her seferinde en büyük veya en küçük eleman seçilirse.)
- **Mekan Karmaşıklığı**: O(log n) (Rekürsif çağrılar için kullanılan bellek.)

3. Dinamik Programlama Algoritmaları (Dynamic Programming Algorithms)

a. Fibonacci Serisi (Memoization Yöntemi)

Fibonacci sayılarının hesaplanması, önceki sonuçları kaydederek (memoization) optimize edilir.

- **Zaman Karmaşıklığı**: O(n) (Her Fibonacci sayısı bir kez hesaplanır.)
- **Mekan Karmaşıklığı**: O(n) (Her adımda sonuçları depolamak için bellek kullanılır.)

b. En Uzun Ortak Alt Dizi (Longest Common Subsequence - LCS)

İki dizi arasındaki en uzun ortak alt diziyi bulmak için dinamik programlama kullanılır.

- **Zaman Karmaşıklığı**: O(n * m) (n ve m, iki dizinin uzunlukları.)
- **Mekan Karmaşıklığı**: O(n * m) (Tüm alt dizi sonuçlarını saklamak için bir tablo kullanılır.)

4. Graf Algoritmaları (Graph Algorithms)

a. Dijkstra Algoritması

En kısa yolu bulmak için kenar ağırlıklı bir graf kullanır.

- **Zaman Karmaşıklığı**:
- O((V + E) log V) (V, düğüm sayısı; E, kenar sayısı.)
- **Mekan Karmaşıklığı**: O(V) (Düğümler için mesafeleri saklamak için.)

b. Kruskal Algoritması

Bir grafın minimum yayılım ağacını (minimum spanning tree) bulmak için kullanılır.

- **Zaman Karmaşıklığı**: O(E log E) (Kenarlar sıralanır ve birleştirme işlemi yapılır.)
- **Mekan Karmaşıklığı**: O(E + V) (E, kenar sayısı; V, düğüm sayısı.)

5. Böl ve Yönet Algoritmaları (Divide and Conquer Algorithms)

a. Strassen Matris Çarpımı

İki matrisi çarpmanın daha hızlı bir yolunu bulur.

- **Zaman Karmaşıklığı**: O(n^2.81) (Standart O(n^3) yerine daha hızlıdır.)
- **Mekan Karmaşıklığı**: O(n^2) (Ekstra matrisler oluşturmak için bellek kullanılır.)

b. En Yakın Çift Problemi (Closest Pair of Points)

Bir düzlemdeki en yakın iki noktayı bulmak için noktalar bölünerek en kısa mesafe hesaplanır.

- **Zaman Karmaşıklığı**: O(n log n) (Böl ve yönet tekniği ile hızlandırılmıştır.)
- **Mekan Karmaşıklığı**: O(n) (Ekstra alan gerektiren bölme işlemleri.)