1-)Selection sort, merge sort ve quicksort algoritmalarının en iyi durum, en kötü durum ve ortalama durum karmaşıklıkları nedir?

Cevap:

Selection Sort:

* En iyi durumda, dizi zaten sıralanmış olduğu için karmaşıklık O(n^2) dir.
* En kötü durumda, dizi tamamen ters sıralanmış olduğunda karmaşıklık yine O(n^2) dir.
* Ortalama durumda da, karmaşıklık O(n^2) dir.

Merge Sort:

* En iyi durum, en kötü durum ve ortalama durumda karmaşıklık O(n log n)'dir.

Quicksort:

* En iyi durumda, pivot elemanı dizinin medyanı gibi seçildiğinde, karmaşıklık O(n log n) dir.
* En kötü durumda, pivot elemanının seçimi kötü olduğunda (örneğin dizinin en küçük ya da en büyük elemanı seçildiğinde), karmaşıklık O(n^2) dir.
* Ortalama durumda, pivot elemanının rastgele seçildiği varsayıldığında karmaşıklık O(n log n) dir.

2-)"Append" işlemi bazen başka bir öğe eklemek için alanı tükendiğinde nasıl O(1) karmaşıklığına ulaşabilir?

Cevap: "Append" işlemi genellikle O(1) karmaşıklığına sahiptir çünkü yalnızca sona bir öğe eklemek için kullanılır. Ancak bazen, bellek yeniden boyutlandırma işlemi gerektiği için O(n) karmaşıklığına sahip olabilir. Yeniden boyutlandırma işlemi, her öğe ekleme işlemi için yapılmaz, bu nedenle "append" işleminin amortize edilmiş karmaşıklığı O(1)'dir.

3-)Listeler için birleştirme işlemi operatörü olan + işleminin karmaşıklığı nedir?

Cevap: Listeler için birleştirme işlemi operatörü olan + işleminin karmaşıklığı, listenin boyutuyla doğru orantılıdır. Yani, iki liste birleştirildiğinde, birleştirilen listenin uzunluğu, birinci listenin uzunluğuna ve ikinci listenin uzunluğuna eşittir. Dolayısıyla, + operatörü, her iki listenin toplam boyutuna bağlı olarak O(n+m) karmaşıklığına sahiptir, burada n ve m birinci ve ikinci listenin uzunluğudur.

4-)Listeler için silme işlemi operatörünün karmaşıklığı nedir?

Cevap: Listeler için silme işlemi operatörünün karmaşıklığı silinecek öğenin konumuna bağlıdır. Son öğe silinecekse, O(1) karmaşıklığına sahiptir. Ancak, listenin başındaki veya belirli bir konumundaki öğe silinecekse, O(n) karmaşıklığına sahiptir.

5-)Bir listedeki öğeleri sıralarken, bu öğeler için hangi yöntemin tanımlanması gerekir ve neden?

Cevap: Öğelerin karşılaştırılabilmesi için, bu öğelerin veri tipine göre bir karşılaştırma yöntemi tanımlanmalıdır. Örneğin, sayılar için karşılaştırma yöntemi "büyüklük" veya "küçüklük" olabilirken, metin dizeleri için alfabetik sıralama kullanılabilir. Tanımlanan karşılaştırma yöntemi, sıralama algoritmasının doğru şekilde çalışabilmesi için gereklidir.

6-) Neden quicksort, merge sort'tan daha iyi performans gösterir?

Cevap: Quicksort, merge sort'a göre daha iyi performans gösterir, çünkü genellikle daha az işlem yapar. Quicksort, her adımda bir elemanı doğru pozisyona yerleştirmeye çalışır ve bu işlemi yaparken verileri parçalara ayırır. Buna karşılık, merge sort, birleştirme işlemi sırasında ek bellek kullanır ve bu nedenle daha yavaş çalışabilir.

7-)Merge sort'un quicksort'tan daha iyi performans göstermesi hangi koşullar altında mümkündür?

Cevap: Merge sort, bazı durumlarda quicksort'tan daha iyi performans gösterebilir. Özellikle, veri kümesi büyük olduğunda ve bellek kısıtlamaları mevcut olduğunda merge sort daha iyi performans gösterebilir.

8-)Bir liste bölündüğünde neler olduğunu özetleyin.

Cevap: Bir liste bölündüğünde, listenin elemanları, belirli bir elemana göre iki parçaya ayrılır. Bu bölme işlemi, bir elemanın (genellikle listenin son elemanı veya rastgele bir elemanı) seçilmesi ve listenin geri kalan elemanlarının bu elemana göre sıralanmasıyla gerçekleştirilir.

Bu işlemin sonunda, seçilen elemanın doğru konumda olduğu garantilenir ve bu elemanın sağında kalan elemanlar daha büyük, solunda kalanlar ise daha küçük olur. Bu işlem genellikle quicksort algoritmasında kullanılır ve listenin sıralanmasına yardımcı olur.

9-)Merge sort algoritmasında iki liste nasıl birleştirilir? Özetleyin.

Cevap: Merge sort algoritmasında, iki sıralanmış liste birleştirilirken, listenin iki yarısı birbirine karşılaştırılır ve birleştirilir. İlk olarak, her iki liste de tek tek elemanlarına ayrılır ve ardından elemanlar karşılaştırılarak birleştirilir.

Karşılaştırma sırasında, her iki listenin en küçük elemanı seçilir ve yeni birleştirilmiş listeye eklenir. Seçilen eleman, kaynak listenin başından kaldırılır ve kalan elemanlar tekrar karşılaştırılarak birleştirilir. Bu işlem, her iki listenin tüm elemanları yeni listeye eklenene kadar devam eder.

Birleştirme işlemi, iki sıralanmış listenin elemanlarını birleştirdiği için, sonuçta elde edilen liste de sıralanmış bir liste olur. Merge sort, bu birleştirme işlemini tekrarlayarak, veri kümesini küçük parçalara ayırarak sıralama işlemini gerçekleştirir.

10-)Selection sort algoritmasındaki select fonksiyonundaki başlangıç parametresinin amacı nedir?

Cevap: Selection sort algoritmasındaki select fonksiyonundaki başlangıç parametresinin amacı, sıralama işleminin belirli bir noktasından başlayarak, bu noktadan sonraki en küçük elemanı seçmektir. Bu parametre, selection sort algoritmasının verimliliğini artırmak için kullanılır.

11-)Bir liste veri yapısının rastgele erişilebilir bir liste uygulamasına göre bağlantılı liste uygulamasının avantajları nelerdir?

Cevap: Bağlantılı liste uygulaması, veri elemanlarının birbirine bağlantılı olduğu bir yapıda tutulur ve bu nedenle elemanların yerleri bellekte değiştirilebilir. Rastgele erişilebilir listenin aksine, bağlantılı liste uygulaması aşağıdaki avantajlara sahiptir:

1. Eleman ekleme ve silme işlemleri kolaydır: Bağlantılı listenin elemanları arasında bağlantılar olduğu için eleman ekleme ve silme işlemleri rastgele erişilebilir listenin aksine daha kolaydır. Bu işlemler için sadece bağlantıları değiştirmek yeterlidir.
2. Bellek kullanımı daha verimlidir: Rastgele erişilebilir liste, listenin tamamının bellekte sürekli bir blokta tutulmasını gerektirirken, bağlantılı liste sadece kullanılan bellek miktarı kadar yer kaplar.
3. Liste boyutu dinamik olarak değiştirilebilir: Bağlantılı listenin boyutu eleman ekleme ve çıkarma işlemleriyle dinamik olarak değiştirilebilirken, rastgele erişilebilir listenin boyutu önceden belirlenir ve daha sonra değiştirilemez.
4. Liste birleştirme işlemleri kolaydır: Bağlantılı liste uygulaması, iki liste arasında birleştirme işlemlerini daha kolay yapar. Sadece iki listenin son elemanlarının bağlantısını değiştirmek yeterlidir.

12) "Sıralı liste veri tipinin bağlantılı liste uygulamasına göre rastgele erişilebilir liste uygulamasının avantajları nelerdir?"

Cevap: Rastgele erişilebilir listenin, bağlantılı liste uygulamasına göre bazı avantajları vardır:

1. Elemanlara doğrudan erişim sağlama: Rastgele erişilebilir listenin elemanları, doğrudan indeks kullanarak erişilebilir. Bu, elemanların bulunmasını ve değiştirilmesini kolaylaştırır.
2. Bellek kullanımı: Rastgele erişilebilir liste, bellek kullanımı açısından daha az verimlidir. Çünkü listenin tüm elemanları, bellekte sürekli bir blokta tutulmalıdır.
3. Liste boyutunun değiştirilmesi: Rastgele erişilebilir liste boyutu önceden belirlenir ve değiştirilemez. Bu, eleman ekleme ve çıkarma işlemlerini zorlaştırabilir.
4. Liste birleştirme işlemleri: Rastgele erişilebilir liste uygulaması, iki listenin birleştirilmesi işlemlerini daha zor hale getirir.
5. Bir yığın (stack), bir kuyruk (queue) ile erişim açısından nasıl farklılık gösterir

Cevap: Bir yığın (stack) ve bir kuyruk (queue) veri yapıları arasındaki temel fark, nasıl erişildikleridir. Yığın, son eklenen elemanın en önce çıkarılması mantığına dayanır. Bu nedenle, yığından eleman çıkarırken, son eklenen eleman ilk çıkarılır (LIFO - son giren ilk çıkar). Diğer yandan, kuyruk, ilk eklenen elemanın en önce çıkarılması mantığına dayanır. Bu nedenle, kuyruktan eleman çıkarırken, ilk eklenen eleman ilk çıkarılır (FIFO - ilk giren ilk çıkar).

1. Radix sort algoritmasının karmaşıklığı nedir?

Cevap: Radix sort, diğer sıralama algoritmalarından farklı olarak rakamların basamak değerlerine göre sıralama yapar. Algoritma, önce en düşük basamaktan başlayarak elemanları sıralar, sonra ikinci düşük basamaktan başlayarak tekrar sıralama yapar ve bu işlem en yüksek basamakta tamamlanır.

Bu algoritmanın en kötü ve ortalama zaman karmaşıklığı O(d \* (n + b))'dir, burada "n" sıralanacak elemanların sayısı, "d" ise elemanların en fazla kaç basamaklı olduğunu belirtir ve "b" ise kullanılan sayı sistemine göre rakam sayısını ifade eder. Özellikle, "b" sayısı genellikle 10'dur (onlu sayı sistemi için) ve bu durumda karmaşıklık O(d \* n)'dir.