**SIRALAMA ALGORİTMALARININ ÇALIŞMA**

**SÜRELERİNİN KARŞILAŞTIRILMASI-2**

**Ad:** Mustafa

**Soyad:** Yıldırım

**No:** 20060392

Verilen değerler, C programlama dilinde Dev-C++ IDE 5.11.0.0 sürümünde derlenip çalıştırılmıştır. Kullanılan bilgisayarın özellikleri,

#### CPU: Intel® Core™ i5-10300H CPU @ 2.50GHz

#### RAM: 8GB

#### İşletim Sistemi: Microsoft Windows 11 Pro (64 bit)-21H2 (10.0.22000) sürümü

**Sistem Türü:** 64 bit işletim sistemi, x64 tabanlı işlemci

 Quick Sort algoritması, sıralanacak bir diziyi daha küçük iki parçaya ayırıp oluşan bu küçük parçaların kendi içinde sıralanması mantığıyla çalışır. Her bir adımda, pivot eleman seçilir. Daha sonra dizinin bütün elemanları, üzerinde gezilir. Pivot elemandan büyük olanlar sağa, küçük olanlar sola atılır. Bu şekilde verilen dizi sıralanır.

Merge Sort, diziyi ardışık olarak en küçük alt dizilerine kadar yarılar. Daha sonra da onları sıraya koyarak birleştiren özyinelemeli bir algoritmadır. Yarılama işlemi, en büyük alt dizi en çok iki elemanlı olana kadar sürer. Sonra "Merge (Birleştirme)" işlemiyle bu diziler ikişer ikişer bölünüş sırasıyla sıralı olarak bir üst dizide birleşir. Süreç sonunda en üstte sıralı diziye ulaşılır.

Heap Sort algoritması kısaca, sarka planda bir [yığın ağacı(heap)](http://bilgisayarkavramlari.com/2008/08/09/yigin-agaci-heap/)oluşturur ve bu ağacın en üstündeki sayıyı alarak sıralama işlemi yapar.

Algoritmaların çalışma durumlarına göre zaman karmaşıklığı,

|  |  |  |  |
| --- | --- | --- | --- |
| **Zaman Karmaşıklığı** | **Merge Sort** | **Heap Sort** | **Quick Sort** |
| **En kötü durum(worst case)** | O(n log(n)) | O(n log(n)) | O(n^2) |
| **Ortalama durum** | θ(n log(n)) | θ(n log(n)) | **θ(n log(n))** |
| **En iyi durum(best case)** | Ω(n log(n)) | Ω(n log(n) | **Ω(n log(n))** |

Şeklindedir.

|  |  |  |  |
| --- | --- | --- | --- |
| İnput.txt | Heap Sort | Quick Sort | Merge Sort |
| Çalışma Süresi | 0.002s | 0.001s | 0.002s |

Verilerin rastgele sıralı olduğu durumda, bu 3 algoritmanın zaman karmaşıklığı birbirine yakın değerler verir. Tabloda belirtildiği gibi Quick sort algoritması en hızlı çalışmıştır. Merge Sort algoritması ile Heap Sort algoritması aynı zaman aralığına sahip olmalarına rağmen Heap Sort algoritması bellekte Merge Sort algoritmasına göre daha az yer tutar ve bu algoritmalar çalıştığında Heap Sort algoritması daha hızlı çalışır.

Quick Sort ise bir dizinin elemanlarını sıraya koymak için sistematik bir yöntem olarak çalışan etkili bir sıralama algoritmasıdır. Buna karşılık, Merge Sort algoritması ise; etkili, genel amaçlı, karşılaştırma tabanlı bir sıralama algoritmasıdır.   
Quick Sort, küçük veri kümeleri için daha hızlı çalışırken, Merge Sort tüm veri kümeleri için tutarlı bir hızda çalışır.

Eğer sıralama algoritması O1 düzeyinde bir belleğe ihtiyaç duyuyorsa bu algoritma in-place dır.

Quick sort in-place bir algoritmadır, fazla hafıza ihtiyacı yoktur. Merge sort ekstra hafızaya ihtiyaç duyar. Bu sebeple, böl ve yönet tekniğini kullanan bu iki algoritma, 6000 veri için Quick Sort daha hızlı çalışmıştır.

|  |  |  |  |
| --- | --- | --- | --- |
| sirali.txt | Heap Sort | Quick Sort | Merge Sort |
| Çalışma Süresi | 0.005s | 0.004s | 0.005s |

Sirali.txt veriler; küçükten büyüğe sıralı olduğu için bu, algoritmaların zaman karmaşıklığı için en iyi durumdur.

Quick Sort un en hızlı çalışma sebebi, ekstra bellek gerektirmemesinden kaynaklanıyor. Merge Sort ekstra bellek gerektiriyor. En iyi durumda iki sıralama algoritmasında O(n log(n)) karmaşıklığında sonuç ürettiğini alırsak, ekstra bellek gerektirmeyen **Quick Sort**algoritması Merge Sort tan daima bir adım önündedir.

|  |  |  |  |
| --- | --- | --- | --- |
| Tersten\_sirali.txt | Heap Sort | Quick Sort | Merge Sort |
| Çalışma Süresi | 0.002s | 0.025s | 0.001s |

Tersten\_sirali.txt veriler; büyükten küçüğe sıralı olduğu için bu, algoritmaların zaman karmaşıklığı için en kötü durumdur.

Quick Sort algoritmasında, pivot seçimi kötü yapılırsa yani bölme noktası sürekli olarak sıralanmış haldeki listenin ilk veya son elemanı olursa karmaşıklık doğrudan O(n**²**) ye gider. Ancak doğru pivot seçimi için geliştirilmiş ve çok iyi sonuç üreten yaklaşımlar vardır. Tersten\_sirali.txt sıralarken Quick sort en kötü durumdaki zaman karmaşıklığına sahip olmuştur. Quick sort un en yavaş çalışmasına sebep olmuştur. Quick sort, doğru pivot seçimi ile daha hızlı çalışmasını sağlayabiliriz. Merge Sort algoritması ile Heap Sort algoritması aynı zaman aralığına sahip olmalarına rağmen Heap Sort algoritması bellekte Merge Sort algoritmasına göre daha az yer tutar ve bu algoritmalar çalıştığında Heap Sort algoritması daha hızlı çalışır.

Sonuç olarak, Quick sort diğer sıralama algoritmalarından daha hızlı çalışmıştır. Böl-yönet yaklaşımı kullanmasından dolayı, ekstra bellek tüketimine gerek duymaz ve bunu yaparken pivot seçimi kullanır. Eğer pivot seçimi doğru yapılırsa, daha hızlı çalışabilir. Merge sort ve Heap sort sıralama algoritmalarının daha fazla belleğe ihtiyaç duyması, Quick sort a göre, daha yavaş çalışmalarına sebep olmuştur. Heap Sort algoritmasının, kullanımı sınırlıdır çünkü Quick Sort ve Merge Sort pratikte, Heap sort tan daha iyidir. Heap sort, seçerek sıralama yaptığından diğer algoritmalara göre daha yavaş kalmıştır.

<https://visualgo.net/en/sorting> sitesini kullanarak görsel olarak analiz ettim.