**SIRALAMA ALGORİTMALARININ ÇALIŞMA**

**SÜRELERİNİN KARŞILAŞTIRILMASI**

**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

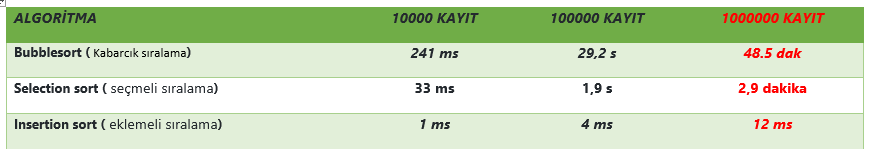
#### İşletim Sistemi: Microsoft Windows 11 Pro (64 bit)-21H2 (10.0.22000) sürümü

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| İnput.txt | Selection Sort | Bubble Sort | Insertion Sort | Shell Sort |
| Çalışma Süresi | 0.3391s | 0.3622s | 0.3269s | 0.3097s |

Selection Sort ve Insertion Sort, Bubble Sort algoritmasına göre tabloda görüldüğü gibi daha hızlıdır. Bubble sort algoritması, diğer sıralama tekniklerine göre en yavaş çalışmıştır. Selection Sort, Insertion Sort’ tan daha az atama işlemi yaparken daha fazla karşılaştırma işlemi yapar. Bu sebeple daha yavaş çalışmıştır. Insertion sort, 6000 adet rastgele sayı için Selection sort’ a göre daha hızlı çalışmıştır. Daha fazla veri olması durumunda dahi Insertion sort daha hızlı çalışır. Shell sort, Insertion sort un genelleştirilmiş halidir. Shell Sort un sıralama işlemi için öncelikle bir atlama miktarının olması, Insertion Sort a göre daha hızlı çalışmasını sağlamıştır.

Rastgele verilerin kayıt süreleri,

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| sirali.txt | Selection Sort | Bubble Sort | Insertion Sort | Shell Sort |
| Çalışma Süresi | 0.3456s | 0.3166s | 0.3057s | 0.328s |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Zaman Karmaşıklığı | Selection Sort | Bubble Sort | Insertion Sort | Shell Sort |
| En iyi durum(best case) | Ω (n2) | Ω (n) | Ω (n) | N^logn |
| En kötü durum(worst case) | O (n2) | O (n2) | O (n2) | N^logn |

Tabloda verildiği gibi, zaman karmaşıklığı en fazla Selection sort algoritmasına aittir. Bu sebeple en iyi durumda, en yavaş çalışmıştır. Shell sort, Bubble sort ve Insertion sort a göre daha yavaş çalışmıştır. Sebebi yine zaman karmaşıklığından dolayıdır. Bubble sort un Insertion Sort a göre yavaş olma sebebi, dizide sıralanacak sayı kalmayana kadar tekrar tekrar en başa dönmesidir.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tersten\_sirali.txt | Selection Sort | Bubble Sort | Insertion Sort | Shell Sort |
| Çalışma Süresi | 0.3387s | 0.3532s | 0.3294s | 0.3036s |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Zaman Karmaşıklığı | Selection Sort | Bubble Sort | Insertion Sort | Shell Sort |
| En iyi durum(best case) | Ω (n2) | Ω (n) | Ω (n) | N^logn |
| En kötü durum(worst case) | O (n2) | O (n2) | O (n2) | N^logn |

Tabloda verildiği gibi, zaman karmaşıklığı en az Shell sort algoritmasına aittir. Bu sebeple en kötü durumda, en hızlı çalışmıştır. Selection Sort, Insertion Sort’ tan daha az atama işlemi yaparken daha fazla karşılaştırma işlemi yapar. Bu sebeple, Insertion sort a göre daha yavaş çalışmıştır. Bubble sort en yavaş olması sebebi, dizide sıralanacak sayı kalmayana kadar tekrar tekrar en başa dönmesidir. Bu, en iyi durum haricinde Bubble Sort’ u en yavaş çalışan algoritma yapmıştır.

<https://www.toptal.com/developers/sorting-algorithms> sitesini kullanarak analiz ettim.

Sonuç olarak;

1-Veriler rastgele olduğu zaman,

Shell sort> Insertion sort> Selection sort> Bubble sort

Çalışma hızları bu şekildedir.

2- Veriler en kötü durumda olduğu zaman,

Shell sort> Insertion sort> Selection sort> Bubble sort

Çalışma hızları bu şekildedir.

3- Veriler en iyi durumda olduğu zaman,

Insertion Sort >Bubble Sort >Shell Sort >Selection Sort

Çalışma hızları bu şekildedir.