Veri Yapıları ve Algoritmalar

Sıralama Algoritmaları Analizi

Enver Arslan
enver@programmer.net

Sıralama Algoritmaları Analizi

- Bubble Sort
- Selection Sort
- Insertion Sort
- Quick Sort
- Merge Sort
- Radix Sort

Düz sıralı listede Bubble sort için en iyi durum söz konusudur. Bu yüzden algoritma karmaşıklığı O(n) olur. Ters sıralı listede ise bubble sort en kötü durumla karşılaşır. Her karşılaştırma yer değiştirme ile sonuçlanır ve operasyon sayısını arttırır. Bubble sort ters sıralı listede $O(n^2)$ zaman karmaşıklığı ile çalışır.

Düz sıralı liste Selection sort ve Quick Sort için ise en kötü durumlar olurlar. Selection sort en küçük değeri bulmak için dizinin üzerinden bir kere geçer. En küçük değeri dizinin ilk elemanında bulsa bile diğer elemanlarla karşılaştırma yapmak zorundadır. Bu yüzden zaman karmaşıklığı $O(n^2)$ dir.

Quick sort'un standart pivot seçimi baştaki ya da sondaki eleman olarak yapıldığı için, pivot elemanın baştan ya da sondan seçimine göre pivot elemandan daha küçük ya da daha büyük alt diziler hep bir elemanlı diziler olacaktır. Çok elemanlı sıralı dizilerde Quick sort implementasyonu eğer düzgün pivot seçimi yapılmazsa uygulamanın stack boyutunu aşarak “segmentation fault” verecektir. Aynı durum ters sıralı liste için de geçerlidir. Quick sortun ortalama zaman karmaşıklığı O(n logn) olmasına rağmen ters ve düz sıralı listeler Quick sort için en kötü durumlar olacaktır ve $O(n^2)$ seviyesinde çalışacaktır.

Insertion sort düz sıralı listede en iyi durumda olur. Dizinin üzerinden bir kere geçilip karşılaştırma yapıldığında bir değişim söz konusu olmayacağı için hızlı çalışacaktır. Dizinin üzerinden bir kere geçtiği için O(n) zaman karmaşıklığında çalışacaktır. Ters sıralı listede ise her eleman kaydırılarak sağa taşınacağı için operasyon sayısı ve zaman karmaşıklığı artarak $O(n^2)$ seviyesinde çalışacaktır.

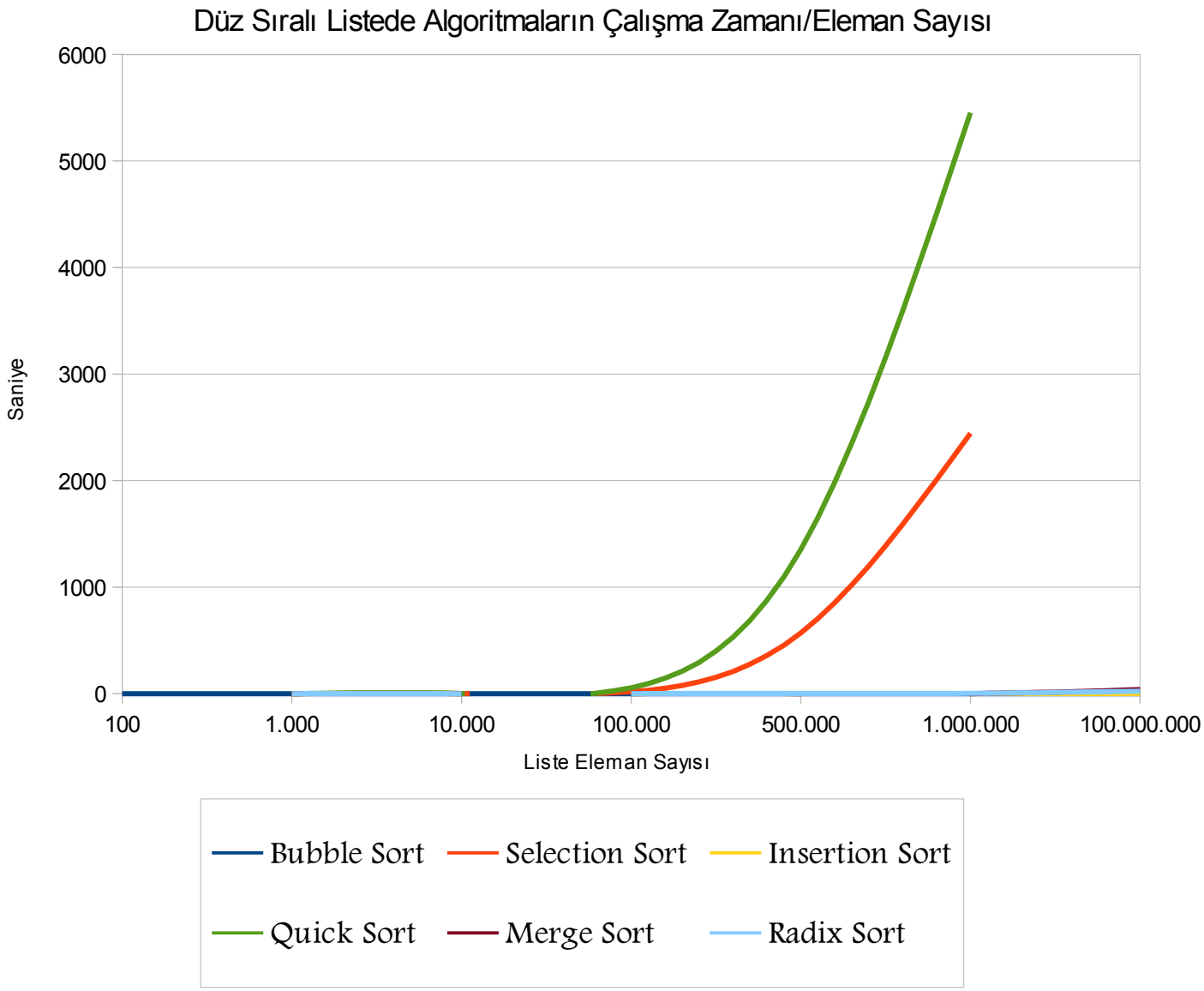
Merge sort için dizinin sıralı olması sadece yapılacak karşılaştırma ve yer değiştirmelerin sayısını değiştirecektir. Bu da uygulamanın içerisindeki operasyon sayısını belirler. Ayrıca merge sort'un ekstra alan ihtiyacı duymasından dolayı implementasyonu yapılırken uygulama stack boyutunu arttıracağını düşünerek dizilerin stackta değil heapta oluşturulması tercih edilebilir. Dizinin sıralı ya da ters sıralı olması merge sort ve radix sort'un zaman karmaşıklığını etkilemez.

Aşağıdaki tablolarda düz ve ters sıralı listelerin eleman sayıları ve çalışma zamanları verilmiş, grafiklerle gösterilmeye çalışılmıştır. Fakat bazı algoritmaların çalışma zamanları uzun zamanlar aldığı için (saatler ve belki de günler) tablo boş bırakılmıştır. C dilinde bu algoritmaların implementasyonları yapılmış ve büyük elemanlı dizilerde stack boyutu arttırılarak çalıştırılmıştır. Kaynak kodlar bu dosya ile aynı yerde bulunmaktadır. Kaynak kodlara aynı zamanda https://github.com/enverarslan/SortingAlgorithms adresinden erişilebilir.

Algoritma Çalışma Zamanı ve Eleman Sayısı Grafiği

A) Düz Sıralı Listede Algoritmaların Çalışma Zamanı/Eleman Sayısı Tablosu

Algoritma / Süre(saniye)	100	1000	10000	100000	500000	1000000	100000000	Karmaşıklık
Bubble	0	0	0	0	0	0	0	O(n)
Selection	0	0	1	18	569	2444	Saatler alır.	$O(n^2)$
Insertion	0	0	0	0	0	0	1	O(n)
Quick	0	0	1	55	1354	5453	Saatler alır.	$O(n^2)$
Merge	0	0	0	0	0	0	37	O(nlogn)
Radix	0	0	0	0	1	2	23	O(nk)



B) Ters Sıralı Listede Algoritmaların Çalışma Zamanı/Eleman Sayısı Tablosu

Algoritma / Süre(saniye)	100	1000	10000	100000	500000	1000000	100000000	Karmaşıklık
Bubble	0	0	1	39	994	4232	Saatler alır.	O(n²)
Selection	0	0	0	25	634	2546	Saatler alır.	O(n²)
Insertion	0	0	0	28	760	3044	Saatler alır.	O(n²)
Quick	0	0	0	34	851	3375	Saatler alır.	O(n²)
Merge	0	0	0	0	0	1	40	O(nlogn)
Radix	0	0	0	0	0	0	23	O(nk)