İçindekiler

Sort Algoritmaları Nedir?

Sıralama algoritmaları en çok kullanılan algoritma türlerinden biridir. Pek çok sıralama algoritmaları vardır. Bu algoritmalar bilgisayar bilimcileri tarafından zaman ve hafıza verimliliği açısından karşılaştırılıp duruma göre hangisi daha verimli ise o kullanılır.

Seçmeli Sıralama (Selection Sort) bu sıralama algoritmalarından birisidir.

Selection Sort Algoritması

Verilerin hafızada sıralı biçimde tutulması için geliştirilen algoritmalardan biridir. Her adımda dizideki en küçük sayının nerede olduğunu bulur ve en küçük sayı ile baştaki sayı arasında yer değiştirme işlemi yapar.

Seçmeli sıralama algoritması daha çok küçük verili dizilerde veya önceden bir kısmı sıralanmış olan dizilerde tercih edilir.

Selection Sort Algoritması Çalışma Mantığı

Selection Sort algoritmasının çalışma mantığı oldukça basittir.

- Dizinin ilk öğesi başlangıçta en küçük sayı olarak kabul edilir. Daha sonra ilk terimi diğer terimler ile karşılaştırır ve bulduğu en küçük terim ile yer değiştirir.
- Dizide seçilen en küçük terimi çıkarınca bir sağdaki terimi baz alarak aynı işlemi tekrar eder.
- Bu işlemi bütün elemanlar üzerinde uygular. Her adım başladığında seçilen elemanın sol tarafı dizili bir hal almış olur. Sağ taraftaki sırasız altdizi bitince sıralama işlemi son bulur.

Örnek: 3, 17, 86, -9, 7, -11, 38 sayılarını küçükten büyüğe doğru sıralayalım.

- İlk adımda 3 sayısı en küçük sayı olarak kabul edilir ve diğer elemanlar ile kıyaslama işlemi yapılır. Diğer sayılar ile kıyaslandığında en küçük sayı -11 olduğu için 3 ile -11 arasında takas işlemi gerçekleşir.
- 3, 17, 86, -9, 7, -11, 38 ❼ İlk aşamada 3 sayısını seçtik
- Swap(4,-10) ❼ -11, 17, 86, -9, 7, 3, 38
- Swap(18,-8) ❼ -11, -9, 86, 17, 7, 3, 38
- Swap(87, 4) ❼ -11, -9, 3, 17, 7, 86, 38
- Swap(8, 18) ❼ -10, -9, 3, 7, 17, 86, 38
- Swap(38,86) ❼ -10, -9, 3, 7, 17, 38, 86
- -10, -9, 3, 7, 17, 38, 86 ❼ Normalde sıralama tamamlanmıştı fakat 86 terimini son kez kontrolden geçiriyor.

Not: Gördüğünüz gibi 7 elemanlı işlemimiz 7 aşamada gerçekleşti. Yani “n” elemanlı bir işlem için “n” tane sıralama işlemi gerekiyor.

Selection Sort Zaman Karmaşıklık Analizi

En küçük ilk sayıyı bulurken (n-1), ikinci sayıyı bulurken (n-2) şeklinde azalarak gidiyor.

Toplam Kontrollerin sayısı (n *(n-1)/2) tanedir. Algoritma karmaşıklığı bu nedenle n^2 dir.

Best-Case (Big Omega)

Selection Sort için **Best-Case** yani en iyi durum dizimizin küçükten büyüğe sıralı olarak gelmesi durumudur. Liste zaten sıralı olsa dahi algoritmamız tek tek bütün elemanları kontrol edeceği için algoritma karmaşıklığı n^2 olacaktır. *Ω(n^2)*

Worst-Case (Big O)

Worst-Case yani en kötü durum dizimizin büyükten küçüğe sıralı biçimde olması durumudur. Bu durumda da tüm dizinin üzerinden tek tek geçilecektir. Best Case ile aradaki tek fark elemanların yelerinin değiştirilmesi olacaktır. Ama yapılan işlem aynı olacaktır yani en kötü durum için algoritma karmaşıklığı n^2 dir. O(n^2)

Average-Case (Big Theta)

Best-Case ve Worst-Case durumlarının karmaşıklığı n^2 olduğundan ortalama durum içinde değişen bir şey olmayacaktır. Avarage-Case'nin karmaşıklığıda n^2 dir. Θ(n^2)

Selection Sort Algoritması Örnek C kodu

```c
#include <stdio.h>

int dizi[100];
int temp;
int minimum;

void selectionSort(int dizi[], int elemanSayisi)
{
    int i,j;

    for (i=0; i<elemanSayisi; i++)
    {
        minimum = i;
        for(j=i+1; j<elemanSayisi; j++)
        {
            if(dizi[j]<dizi[minimum])
            {
                minimum=j;
            }
        }

        //Swap işlemimiz
        temp = dizi[i];
        dizi[i] = dizi[minimum];
        dizi[minimum] = temp;
    }
}

int main()
{
    int secim,i;
    while(1==1)
    {
        printf("\n \n\nKac adet sayi gireceksiniz: ");
        scanf("%d",&secim);

        for(i=0; i<secim; i++)
        {
            printf("\n Lutfen %d. sayiyi giriniz: ", i+1);
            scanf ("%d",&dizi[i]);
        }

        selectionSort(dizi,secim);

        for (i=0; i<secim; i++)
        {
            printf("%d \t", dizi[i]);
        }
    }

    return 0;
}

```