

## SHELL SORT ALGORİTMASI

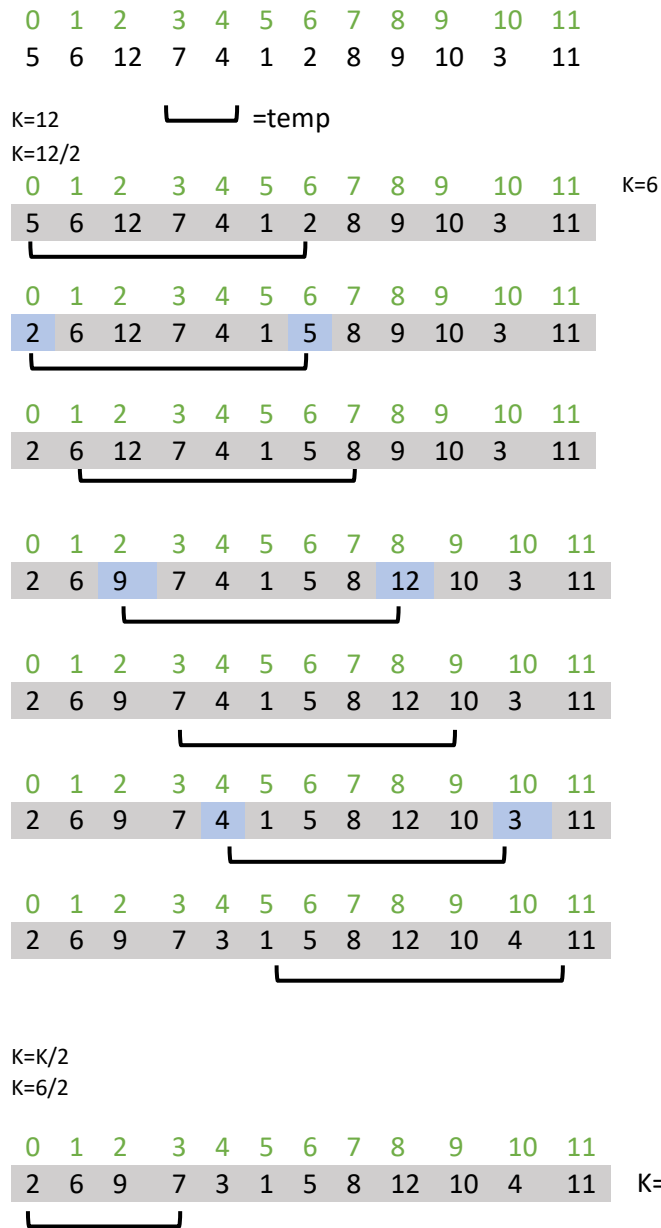
Verileri bellekte böl ve fethet yöntemi kullanarak sıralı tutmak için geliştirilmiş algoritmalardan biridir.

Önce birbirinden uzaktaki öğeleri sıralar ve sıralanacak öğeler arasındaki aralığı art arda azaltır.

Öğeler arasındaki aralık, kullanılan sıraya göre azaltılır. Ve öğeler karşılaştırılarak sıralanır.

### Örnek Problem:

Aşağıda sıralanmamış bir dizi verilmiştir. Shell sort algoritması kullanılarak sıralayınız.



0	1	2	3	4	5	6	7	8	9	10	11
2	6	9	7	3	1	5	8	12	10	4	11

0	1	2	3	4	5	6	7	8	9	10	11
2	3	9	7	6	1	5	8	12	10	4	11

0	1	2	3	4	5	6	7	8	9	10	11
2	3	1	7	6	9	5	8	12	10	4	11

0	1	2	3	4	5	6	7	8	9	10	11
2	3	1	5	6	9	7	8	12	10	4	11

0	1	2	3	4	5	6	7	8	9	10	11
2	3	1	5	6	9	7	8	12	10	4	11

0	1	2	3	4	5	6	7	8	9	10	11
2	3	1	5	6	9	7	8	12	10	4	11

0	1	2	3	4	5	6	7	8	9	10	11
2	3	1	5	6	9	7	8	12	10	4	11

0	1	2	3	4	5	6	7	8	9	10	11
2	3	1	5	6	9	7	4	12	10	8	11

0	1	2	3	4	5	6	7	8	9	10	11
2	3	1	5	4	9	7	6	12	10	8	11

0	1	2	3	4	5	6	7	8	9	10	11
2	3	1	5	4	9	7	6	12	10	8	11

0	1	2	3	4	5	6	7	8	9	10	11
2	3	1	5	4	9	7	6	11	10	8	12

K=3/2

K=1

0	1	2	3	4	5	6	7	8	9	10	11
2	3	1	5	4	9	7	6	11	10	8	12

0	1	2	3	4	5	6	7	8	9	10	11
2	3	1	5	4	9	7	6	11	10	8	12

K=1

0	1	2	3	4	5	6	7	8	9	10	11
2	1	3	5	4	9	7	6	11	10	8	12

0	1	2	3	4	5	6	7	8	9	10	11
1	2	3	5	4	9	7	6	11	10	8	12

0	1	2	3	4	5	6	7	8	9	10	11
1	2	3	5	4	9	7	6	11	10	8	12

0	1	2	3	4	5	6	7	8	9	10	11
1	2	3	4	5	9	7	6	11	10	8	12

0	1	2	3	4	5	6	7	8	9	10	11
1	2	3	4	5	9	7	6	11	10	8	12

0	1	2	3	4	5	6	7	8	9	10	11
1	2	3	4	5	7	9	6	11	10	8	12

0	1	2	3	4	5	6	7	8	9	10	11
1	2	3	4	5	7	9	6	11	10	8	12

0	1	2	3	4	5	6	7	8	9	10	11
1	2	3	4	5	7	6	9	11	10	8	12

0	1	2	3	4	5	6	7	8	9	10	11
1	2	3	4	5	6	7	9	11	10	8	12

0	1	2	3	4	5	6	7	8	9	10	11
1	2	3	4	5	6	7	9	11	10	8	12


0	1	2	3	4	5	6	7	8	9	10	11
1	2	3	4	5	6	7	9	11	10	8	12

0	1	2	3	4	5	6	7	8	9	10	11
1	2	3	4	5	6	7	9	10	11	8	12


0	1	2	3	4	5	6	7	8	9	10	11
1	2	3	4	5	6	7	9	11	10	8	12

0	1	2	3	4	5	6	7	8	9	10	11
1	2	3	4	5	6	7	9	11	8	10	12


0	1	2	3	4	5	6	7	8	9	10	11
1	2	3	4	5	6	7	9	8	11	10	12




0	1	2	3	4	5	6	7	8	9	10	11
1	2	3	4	5	6	7	8	9	11	10	12




0	1	2	3	4	5	6	7	8	9	10	11
1	2	3	4	5	6	7	8	9	11	10	12



0	1	2	3	4	5	6	7	8	9	10	11
1	2	3	4	5	6	7	8	9	10	11	12



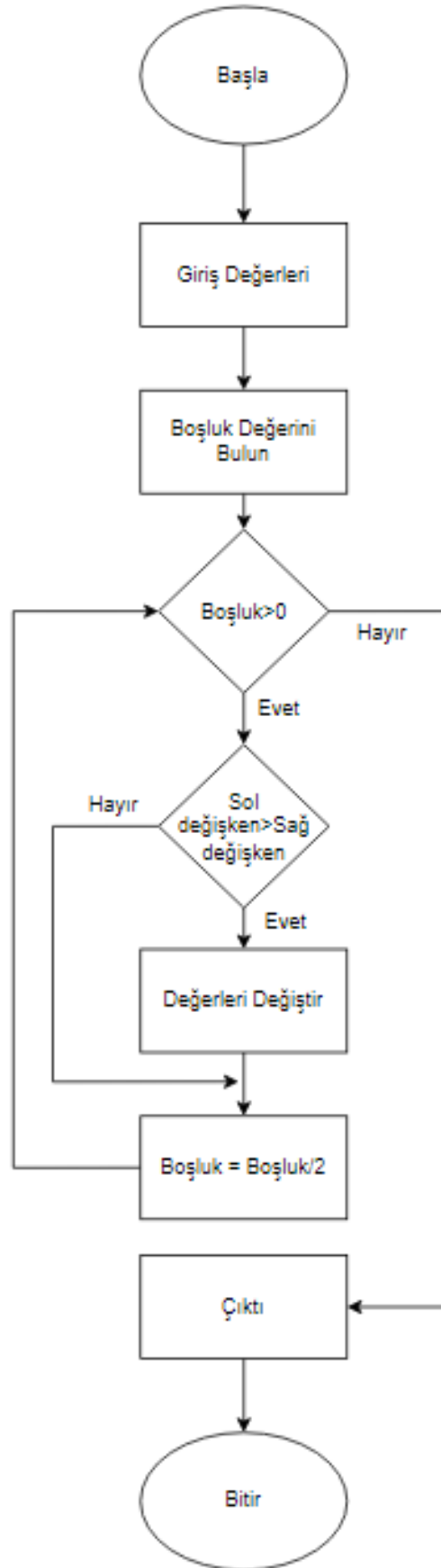
0	1	2	3	4	5	6	7	8	9	10	11
1	2	3	4	5	6	7	8	9	10	11	12



Sıralanmış dizi:

0	1	2	3	4	5	6	7	8	9	10	11
1	2	3	4	5	6	7	8	9	10	11	12

**Çözümün Akış Diyagramı:**



### Shell Sort C++ Kod:

```
1. //bedirhanbuyukozShellSort
2. #include <iostream>
3. using namespace std;
4.
5. // Shell sort
6. void shellSort(int array[], int n) {
7.     //Öğeleri her n / 2, n / 4, n / 8, ... aralıklarla yeniden düzenleyin
8.     for (int gap = n / 2; gap > 0; gap /= 2) {
9.         for (int i = gap; i < n; i += 1) {
10.            int temp = array[i];
11.            int j;
12.            for (j = i; j >= gap && array[j - gap] > temp; j -= gap) {
13.                array[j] = array[j - gap];
14.            }
15.            array[j] = temp;
16.        }
17.    }
18. }
19.
20. // Diziyi Yazd.fonk.
21. void printArray(int array[], int size) {
22.     int i;
23.     for (i = 0; i < size; i++)
24.         cout << array[i] << " ";
25.     cout << endl;
26. }
27.
28. // Sürücü Kod
29. int main() {
30.     int data[] = {5, 6, 12, 7, 4, 1, 2, 8, 9, 10, 3, 11};
31.     int size = sizeof(data) / sizeof(data[0]);
32.     shellSort(data, size);
33.     cout << "Sıralanmış Dizi: \n";
34.     printArray(data, size);
35. }
```

## Shell Sort Algoritma Analizi:

### En iyi durum $\in O(n \log n)$ :

En iyi durum, dizinin zaten sıralanmış olmasıdır. Ardından, artışa dayalı sıralamalarının her biri için karşılaştırma sayısı, dizinin uzunluğudur. Bu nedenle şunları belirleyebiliriz:

$n$  , bir sıralama için  
+  $3 * n/3$  , öğeleriyle 3 elemanlı sıralama için  
+  $7 * n/5$  , ile 5 elemanlı sıralama için  
+  $15 * n/15$  , 15 elemanlı sıralama için  
...  
Her terim  $n$ 'dir . Terim sayısı  $k$  değeridir .  
 $2^k - 1 < n$

Yani  $k < \log(n+1)$  , en iyi durumda sıralama süresinin  $n * \log(n+1) = O(n * \log(n))$  olacaktır.

### En kötü durum $\in O(n^2)$ :

Çok kötü bir senaryoda , her sıralama ikinci dereceden zaman olacaktır.

$n^2$  , 1 sıralama için  
+  $3 * (n/3)^2$  , 3 elemanlı sıralama için  
+  $7 * (n/5)^2$  , 5 elemanlı sıralama için  
+  $15 * (n/15)^2$  , 15 elemanlı sıralama için

Ve böylece, karşılaştırma sayısının şu şekilde sınırlandırıldığını görebiliriz:

$$n^2 * (1 + 1/3 + 1/5 + 1/15 + 1/31 + \dots) < n^2 * (1 + 1/2 + 1/4 + 1/6 + 1/16 + \dots)$$

=  $n^2$  en kötü durum

Son adım, geometrik serinin toplamını kullanır

### En kötü durum $\in \Omega(n^2)$ :

Tüm çift konumlu öğelerin medyandan daha büyük olduğu dizidir. Son 1'lik artışa ulaşana kadar tek ve çift öğeler karşılaştırılmaz. Son yineleme için gereken karşılaştırma/değişim sayısı  $\Omega(n^2)$ 'dir

Özet olarak, Shell Sort şunları içerir:

- Böl ve fethet yöntemini kullanır
- En iyi durum zaman karmaşıklığı  $O(n \log n)$  [tüm öğeler sıralanmışsa]
- $O(n^2)$ 'nin en kötü durum zaman karmaşıklığı [listedeki öğeler sıralanmamışsa]
- $O(1)$  uzay karmaşıklığı