# Quick Sort Algoritması

# (Hızlı Sıralama Algoritması)

Quick Sort (Hızlı Sıralama) algoritması C.A.R.Hoare tarafından bulunan etkin bir sıralama yöntemidir. Siyaset biliminde çok kullanılan *"böl ve yönet"* stratejisine dayanan basit ve hızlı bir sıralama yöntemi kullanır. Zaman karmaşası, ortalama $O(nlogn)$ ve en kötü durumda ise $O(n^2)$ dir.

Bu algoritma, başlarken dizinin terimleri arasından bir terimi *mihenk (pivot)* olarak seçer. Sonra verilen diziyi üç alt diziye ayrıştırır. Mihenk'ten küçük olan terimlerin hepsini (soldaki) birinci altdiziye taşır. İkinci alt dizi biricik öğesi mihenk olan tek terimli {mihenk} altdizsidir. Mihenk'ten büyük olan terimlerin hepsini (sağdaki) ikinci altdiziye taşır. Sonra sol ve sağ altdizilere aynı ayrıştırma yöntemini, altdiziler tek terimli birer diziye indirgenene kadar uygular ve sıralama işlemi biter. Algoritma özyinelgendir (recursive).

Quick Sort algoritmasını bir örnek üzerinde açıklayacağız. {2, 17, -4, 42, 9, 26, 11, 3, 5, 28} dizisi (array) verilsin. Bu dizinin öğelerinin değerlerini birer karta yazalım ve aşağıdaki gibi on gözü olan bir kutuya verilen dizinin indislerinin artan sırasıyla yerleştirelim.

| a[0] | a[1] | a[2] | a[3] | a[4] | a[5] | a[6] | a[7] | a[8] | a[9] |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 17 | -4 | 42 | **9** | 26 | 11 | 3 | 5 | 28 |

Şimdi dizinin terimleri arasından birisini *mihenk (pivot)* olarak seçelim. Hangi terimin mihenk seçildiği algoritmayı etkilemez. Hatta, dizinin öğeleri arasında olmayan bir sayı bile *mihenk* olarak seçilebilir. Ama yapacağımız işlemde eş uygulamayı sağlamak için mihengi şöyle seçelim: Ayrıştırılacak (alt) dizinin ilk teriminin indisi ile son teriminin indislerini toplayıp 2 ye bölelim. Bölümün tamsayı parçasını mihengin indisi olarak seçelim. Bu yolla her altdizinin mihengini bulabiliriz. Örneğin, henüz ayrışmamış bir altdizinin ilk teriminin indisi $s$, son teriminin indisi $t$ ise mihengin indisi $\lfloor (s+t)/2 \rfloor$ olacaktır. Bu indis $(s+t)/2$ tamsayı bölümüdür; yani $(s+t)/2$ bölümünün tamsayı kısmıdır. Başka bir deyişle, (s+t)/2 den daha büyük olmayan en büyük tamsayıdır.

Başlangıçta, verilen dizinin ilk teriminin indisi 0, son teriminin indisi 9 olduğuna göre, ilk mihengin indisi $\lfloor (0+9)/2 \rfloor = 4$ olur. O halde $a[4]=9$ terimini mihenk olarak seçeceğiz. Bu terimi sabit (mihenk, pivot) tutarak, diziyi 9 un solundakiler, 9 un sağındakiler ve {9} olmak üzere üç alt diziye ayrıştıracağız. (Tabii, istenirse $0$ ile $9$ arasındaki herhangi bir indisin seçilebileceğini ve seçilen indisin algoritmayı etkilemeyeceğini unutmuyoruz.)

Şimdi $(a[4]=9)$ mihenginin solunda 9 dan büyük olan terimler varsa onları sağ tarafa taşıyacağız. Benzer şekilde, 9 un sağında 9 dan küçük terimler varsa, onları sol tarafa taşıyacağız. Özel olarak mihenge eşit olan başka terimler varsa, onları sağa ya da sola taşıyabiliriz. Bu işleri yaparken, kutudaki gözlerin sayısının 10 olduğunu ve gözlerin sayısının artamayacağını, iki kartı üst üste koyamayacağımızı unutmayacağız ve ayrıştırmada mümkün olduğu kadar az takas işlemi yapmaya çalışacağız.

Bunun için pratik bir yöntem bulabiliriz. Sol elimizin işaret parmağı en soldaki $a[0]$ dan başlayarak sağa doğru terimleri tarasın. Belli bir anda işaret ettiği gözün indisine *sol* diyelim. Sol parmağımızın sırayla işaret ettiği her terimi *mihenk* ile karşılaştıralım. Parmağımız solda mihenkten $(a[4]=9)$ büyük olan ilk terimi işaret edince, o gözdeki kartı sol elimize alalım. Bu sayı $a[1]=17$ dir.

Sol elimizde 17 yazılı kart dururken, sağ elimizin işaret parmağı en sağdaki $a[9]$ dan başlayarak sola doğru terimleri tarasın. Belli bir anda işaret ettiği gözün indisine *sağ* diyelim. Sağ parmağımızın sırayla işaret ettiği her terimi *mihenk* ile karşılaştırsın. Sağda 9 dan küçük olan ilk terimi içeren kartı

sağ elimize alalım. Bu terim *a[8]=5* sayısıdır. Şimdi sol elimizdeki *17* ile sağ elimizdeki *5* sayılarının yerlerini kendi aralarında değiştirelim (takas). *17* yazılı kartı *a[8]* gözüne, *5* yazılı kartı *a[1]* gözüne taşıyalım. Bu işlem, sol ve sağ ellerimizdeki kartların alındığı gözlerin kendi aralarında değiştirilmesi; yani iki sayının yerlerinin takas (mübadele) edilmesi işlemidir. Bu eylemde göz sayısı artmaz ve iki kart üst üste gelmez. Her takas işlemi, diziyi ayrıştıran adımlardan birisidir.

| a[0] | a[1] | a[2] | a[3] | a[4] | a[5] | a[6] | a[7] | a[8] | a[9] |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 5 | -4 | 42 | 9 | 26 | 11 | 3 | 17 | 28 |

Aynı işlemi solda 9 dan büyük, sağda 9 dan küçük olan her sayı çifti için tekrarlayacağız. Hemen görüldüğü gibi, 42 ile 3 sayılarının yerleri değişecektir.

| a[0] | a[1] | a[2] | a[3] | a[4] | a[5] | a[6] | a[7] | a[8] | a[9] |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 5 | -4 | 3 | 9 | 26 | 11 | 42 | 17 | 28 |

Yukarıda anlatılan takas işlerini, işaret parmaklarımız mihenk üzerinde çakışana kadar sürdüreceğiz.

Bu eylem sonunda, diziyi {2, 5, -4, 3}, {9} ve {26, 11, 42, 17, 28} olmak üzere üç altdiziye ayırmış oluruz.

| a[0] | a[1] | a[2] | a[3] | a[4] | a[5] | a[6] | a[7] | a[8] | a[9] |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 5 | -4 | 3 | 9 | 26 | 11 | 42 | 17 | 28 |

Soldaki altdizideki her terim, sağdaki altdizideki her teriminden küçüktür. Şimdi soldaki ve sağdaki altdizileri yeni dizilermiş gibi düşünüp, yukarıdaki yöntemle onları ayrıştırabiliriz. Sonra onların da sol ve sağ alt dizilerini ayrıştırabiliriz. Bu süreç, alt diziler tek terimli birer diziye indirgenene kadar devam edecektir. Örneğin, soldaki alt diziyi ayrıştıralım. Her adımda, altdizilerin mihenkleri yeşil ile, takas edilen öğeler ise kırmızı renkle işaretlidir.

*Sol alt dizinin ayrıştırılması:*

| | |
|---|---|
| {2, 5, -4, 3} | *a[1] = mihenk = 5, indis = (0+3)/2 = 1* |
| {2, 3, -4, 5} | *takas: 3 ile 5* |
| {2, -4, 3, 5} | *takas: -4 ile 3* |
| {2, -4, 3} {5} | *alt dizilere ayrışma* |
| {-4, 2, 3} {5} | *takas: -4 ile 2* |
| {-4}, {2}, {3}, {5} | *tek öğeli dizilere indirgenmiş* |
| {-4, 2, 3, 5} | *Sıralanmış sol alt dizi* |

*Sağ alt dizinin ayrıştırılması:*

| | |
|---|---|
| {26, 11, 42, 17, 28} | *a[7] = mihenk = 42, indis = (6+9)/2 = 7* |
| {26, 11, 28, 17, 42} | *takas: 42 ile 28* |
| {26, 11, 28, 17} {42} | *ayrıştır* |
| {11, 26, 42, 17, 28} | *takas: 26 ile 11* |
| {11} {26, 28, 17} {42} | *ayrıştır* |
| {11} {26, 17, 28} {42} | *takas: 28 ile 17* |
| {11} {26, 17} {28} {42} | *ayrıştır* |
| {11} {17, 26} {28} {42} | *takas: 17 ile 26* |
| {11} {17} {26} {28} {42} | *ayrıştır* |
| {11, 17, 26, 28, 42} | *Sıralanmış sağ alt dizi* |

Sıralanmış dizi:

| a[0] | a[1] | a[2] | a[3] | a[4] | a[5] | a[6] | a[7] | a[8] | a[9] |
|---|---|---|---|---|---|---|---|---|---|
| -4 | 2 | 3 | 5 | 9 | 11 | 17 | 26 | 28 | 42 |

Yukarıda söylediklerimizin, her adımda dizide yaptığı değişiklikleri aşağıdaki tablodan görebiliriz. Altdizilerin sol ve sağ indislerinin ve mihengin değişimlerine dikkat ediniz.

| sol | sağ | mihenk | a[0] | a[1] | a[2] | a[3] | a[4] | a[5] | a[6] | a[7] | a[8] | a[9] | Açıklama |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 2 | 17 | -4 | 42 | 9 | 26 | 11 | 3 | 5 | 28 | |
| 1 | 8 | 9 | 2 | 5 | -4 | 42 | 9 | 26 | 11 | 3 | 17 | 28 | 1 ↔ 8 |
| 3 | 7 | 9 | 2 | 5 | -4 | 3 | 9 | 26 | 11 | 42 | 17 | 28 | 3 ↔ 7 |
| 4 | 4 | 9 | 2 | 5 | -4 | 3 | 9 | 26 | 11 | 42 | 17 | 28 | *ayrışma* |
| 1 | 3 | 5 | 2 | 3 | -4 | 5 | 9 | 26 | 11 | 42 | 17 | 28 | 1 ↔ 3 |
| 1 | 2 | | 2 | -4 | 3 | 5 | 9 | 26 | 11 | 42 | 17 | 28 | 1 ↔ 2 |
| | | -4 | 2 | -4 | 3 | 5 | 9 | 26 | 11 | 42 | 17 | 28 | *ayrışma* |
| 0 | 1 | | -4 | 2 | 3 | 5 | 9 | 26 | 11 | 42 | 17 | 28 | 0 ↔ 1 |
| | | 42 | -4 | 2 | 3 | 5 | 9 | 26 | 11 | 42 | 17 | 28 | *ayrışma* |
| 7 | 9 | | -4 | 2 | 3 | 5 | 9 | 26 | 11 | 28 | 17 | 42 | 7 ↔ 9 |
| 5 | 8 | 11 | -4 | 2 | 3 | 5 | 9 | 26 | 11 | 28 | 17 | 42 | *ayrışma* |
| 5 | 6 | | -4 | 2 | 3 | 5 | 9 | 11 | 26 | 28 | 17 | 42 | 5 ↔ 6 |
| 6 | 8 | 28 | -4 | 2 | 3 | 5 | 9 | 11 | 26 | 28 | 17 | 42 | *ayrışma* |
| 7 | 8 | 28 | -4 | 2 | 3 | 5 | 9 | 11 | 26 | 17 | 28 | 42 | 7 ↔ 8 |
| 6 | 7 | 26 | -4 | 2 | 3 | 5 | 9 | 11 | 26 | 17 | 28 | 42 | *ayrışma* |
| 6 | 7 | 26 | -4 | 2 | 3 | 5 | 9 | 11 | 17 | 26 | 28 | 42 | 6 ↔ 7 |
| | | | -4 | 2 | 3 | 5 | 9 | 11 | 17 | 26 | 28 | 42 | sıralandı |

Böylece verilen dizinin öğeleri küçükten büyüğe doğru sıralanmış olarak kutunun gözlerine yerleştirilmiş olur.

Quick Sort Algoritmasını yapan bir java metodu yazalım:

```
void quickSort(int[] a, int altindis, int üstindis) {
            // altindis o adımda sıralanan altdizinin ek küçük indisidir
            // üstindis o adımda sıralanan altdizinin ek büyük indisidir

            int i = altindis, j = üstindis, h;
            // x terimi, mukayesenin yapılacağı mihenk'dir (pivot)
            int x = a[(altindis + üstindis) / 2];
            // Takas eylemiyle diziyi ayrıştırma
            do {
                  while (a[i] < x)
                        i++;
                  while (a[j] > x)
                        j--;
                  if (i <= j) {
                        h = a[i];
                        a[i] = a[j];
                        a[j] = h;
                        i++;
                        j--;
                  }
            } while (i <= j);
            // yinelge (recursion)
            if (altindis < j)
                  quickSort(a, altindis, j);
            if (i < üstindis)
                  quickSort(a, i, üstindis);
      }
```

Şimdi bunu bir java uygulamasında çalıştıralım:

```
//package hızlıSıralama;

import java.util.Arrays;

public class HızlıSıralama {
	int[] arr = { 2, 17, -4, 42, 9, 26, 11, 3, 5, 28 };

	void quickSort(int[] a, int altindis, int üstindis) {
		// altindis o adımda sıralanan altdizinin ek küçük indisidir
		// üstindis o adımda sıralanan altdizinin ek büyük indisidir

		int i = altindis, j = üstindis, h;
		// x terimi, mukayesenin yapılacağı mihenk'dir (pivot)
		int x = a[(altindis + üstindis) / 2];
		// Takas eylemiyle diziyi ayrıştırma
		do {
			while (a[i] < x)
				i++;
			while (a[j] > x)
				j--;
			if (i <= j) {
				h = a[i];
				a[i] = a[j];
				a[j] = h;
				i++;
				j--;
			}
		} while (i <= j);
		// yinelge (recursion)
		if (altindis < j)
			quickSort(a, altindis, j);
		if (i < üstindis)
			quickSort(a, i, üstindis);
	}

	public static void main(String[] args) {
		HızlıSıralama qs = new HızlıSıralama();
		System.out.println("Sıralamadan önce: ");
		System.out.println(Arrays.toString(qs.arr));
		qs.quickSort(qs.arr, 0, 9);
		System.out.println("Sıralamadan sonra:");
		System.out.println(Arrays.toString(qs.arr));
	}
}
```

Aşağıdaki quickSort() metodu aynı işi yapar.

```
//package  hızlıSıralama;
import java.util.Arrays;

public class HızlıSıralama {
     int[]  arr = {2, 17, -4, 42, 9, 26, 11, 3, 5, 28} ;
  void qSortInt(int a[], int min, int max) {
```

```
    int altIndis = min;
    int üstIndis = max;
    int mid;

    if (max > min) {
      /*
       * mihenk (pivot) terimi bulunuyor
      */
      mid = a[(min + max) / 2];

      // isdisler çakışana ya da çapraz olana kadar döngü sürüyor
      while (altIndis <= üstIndis) {
        /*
         * Soldan başlayarak, mihenkten küçük olanlara dokunmadan geç
         * mihenk'e eşit ya da büyük olanı bulunca dur ve bekle
         */
        while ((altIndis < max) && (a[altIndis] < mid ))
          ++altIndis;

        /*
         *Sağdan başlayarak, mihenkten büyük olanlara dokunmadan geç
         * mihenk'e eşit ya da küçük olanı bulunca dur ve bekle
         */
        while ((üstIndis > min) && (a[üstIndis] > mid ))
          --üstIndis;

        // Eğer altIndis <= üstIndis ise takas yap
        if (altIndis <= üstIndis) {
          int t = a[üstIndis];
          a[üstIndis] = a[altIndis];
          a[altIndis] = t;

          ++altIndis;
          --üstIndis;
        }
      }

      /*
       * Eğer üstIndis arrayin sol ucuna erişmediyse
       * soldaki alt diziyi ayrıştır
       */
      if (min < üstIndis)
        qSortInt(a, min, üstIndis);
      /*
      * Eğer altIndis arrayin sağ ucuna erişmediyse
      * sağdaki alt diziyi ayrıştır
       */
      if (altIndis < max)
        qSortInt(a, altIndis, max);
    }
  }
    public static void main(String[] args) {
        HızlıSıralama intSort = new HızlıSıralama();
        System.out.println("Sıralamadan önce: ");
        System.out.println(Arrays.toString(intSort.arr));
        intSort.qSortInt(intSort.arr, 0, 9);
        System.out.println("Sıralamadan sonra:");
        System.out.println(Arrays.toString(intSort.arr));
    }
  }
```

Tabii, quick sort algorştması sıralanabilen her veri tipine uygulanabilir. Aşağıda, algoritmanın string tipi verilere uygulanışına bir örnek verilmiştir.

```
//    package hızlıSıralama;
import java.util.Arrays;

public class QuickSortString {
      String[] str = { "Kars", "Zonguldak", "Ankara", "Denizli", "Manisa",
                  "Giresun", "Erzurum", "Hakkari" };

      void qSortStr(String a[], int min, int max) {
            int altIndis = min;
            int üstIndis = max;
            String mid;

            if (max > min) {
                  // mihenk (pivot) terimi bulunuyor
                  mid = a[(min + max) / 2];
                  // isdisler çakışana ya da çapraz olana kadar döngü
sürüyor
                  while (altIndis <= üstIndis) {
                        /*
                         * Soldan başlayarak, mihenkten küçük olanlara
dokunmadan geç
                         * mihenk'e eşit ya da büyük olanı bulunca dur ve
bekle
                         */
                        while ((altIndis < max) &&
(a[altIndis].compareTo(mid) < 0))
                              ++altIndis;
                        /*
                         * Sağdan başlayarak, mihenkten büyük olanlara
dokunmadan geç
                         * mihenk'e eşit ya da küçük olanı bulunca dur ve
bekle
                         */
                        while ((üstIndis > min) &&
(a[üstIndis].compareTo(mid) > 0))
                              --üstIndis;

                        // Eğer altIndis <= üstIndis ise takas yap
                        if (altIndis <= üstIndis) {
                              String t = a[üstIndis];
                              a[üstIndis] = a[altIndis];
                              a[altIndis] = t;
                              ++altIndis;
                              --üstIndis;
                        }
                  }
                  /*
                   * Eğer üstIndis arrayin sol ucuna erişmediyse soldaki
alt diziyi
                   * ayrıştır
                   */
                  if (min < üstIndis)
                        qSortStr(a, min, üstIndis);
                  /*
                   * Eğer altIndis arrayin sağ ucuna erişmediyse sağdaki
alt diziyi
```

```
                 * ayrıştır
                 */
                if (altIndis < max)
                      qSortStr(a, altIndis, max);
          }
    }

  public static void main(String[] args) {
    QuickSortString qs = new QuickSortString();
      System.out.println("Sıralamadan önce: ");
      System.out.println(Arrays.toString(qs.str));
      qs.qSortStr(qs.str, 0, 7);
      System.out.println("\nSıralamadan sonra:");
      System.out.println(Arrays.toString(qs.str));
  }
  }
```