A)

Algoritma	Algoritma Analizi			Kararlılı	Yöntem	Açıklama
	En İyi	Ortalama	En Kötü			
Selection Sort	n^2	n^2	n^2	Kararsız	Seçerek	
Quick Sort	n log(n)	n log(n)	n^2	Kararsız	Parçala Fethet	
Merge Sort	n log(n)	n log(n)	n log(n)	Kararlı	Parçala Fethet	
Heap Sort	n log(n)	n log(n)	n log(n)	Kararsız	Seçerek	
Bubble Sort	n	n^2	n^2	Kararlı	Yer Değiştirme	
Radix Sort	n(k/t)	n(k/t)	n(k/t)	Kararlı	Gruplama / Sayma	k, en büyük eleman değeri, t ise tabandır
Insertion Sort	n	d+n	n^2	Kararlı	Sokma	d yer değiştirme sayısıdır ve n^2 cinsindendir
Bağlı Listelerde Sort(Priorty Queue)	n^2	n^2	n^2	Kararlı	Öncelik Sırasına göre	Bağlı listelere her türlü sort algoritması uyarlanabilmektedir.

En hızlı sonuçlar Quick Sort algoritmasından gelmektedir. Diğer yandan Bubble Sort algoritmasının çok fazla maliyet getirdiği ve uzun sürdüğü ortadadır. Heap ve Merge algoritmaları birbirlerine yakın süreler verir. Insertion ve Selection

algoritmaları tatmin edici hızlarda değildir. Tabi buradaki süreler Milisaniye cinsinden olup alsında çok fazla önemli görünmeyebilir. Ancak matematiksel hesaplamaların yoğun yapıldığı bilimsel uygulamalarda sıklıkla kullanıldıkları ve ihtiyaç duyuldukları da bilinmektedir.

B)

Birleştirme Sıralaması(Merge Sort)

Sıralanmak istenen verimiz:

```
5,7,2,9,6,1,3,7
```

Birleştirme sıralamasının çalışması yukarıdaki bu örnek dizi üzerinde adım adım gösterilmiştir. Öncelikle parçalama adımları gelir. Bu adımlar aşağıdadır.

1. adım diziyi ikiye böl:

5,7,2,9 ve 6,1,3,7

2. adım çıkan bu dizileri de ikiye böl:

5,7 ; 2,9 ; 6,1 ; 3,7

3. adım elde edilen parçalar 2 veya daha küçük eleman sayısına ulaştığı için dur (aksi durumda bölme işlemi devam edecekti)

4. adım her parçayı kendi içinde sırala

5,7 ; 2,9 ; 1,6 ; 3,7

5. Her bölünmüş parçayı birleştir ve birleştirirken sıraya dikkat ederek birleştir (1. ve 2. parçalar ile 3. ve 4. parçalar aynı gruptan bölünmüştü)

2,5,7,9 ve 1,3,6,7

6. adım, tek bir bütün parça olmadığı için birleştirmeye devam et

1,2,3,5,6,7,7,9

7. adım sonuçta bir bütün birleşmiş parça olduğu için dur. İşte bu sonuç dizisi ilk dizinin sıralanmış halidir.

Seçerek Sıralama(Selection Sort)

Sıralanmak istenen verimiz:

```
5,7,2,9,6,1,3,7
```

Seçerek sıralamanın çalışması yukarıdaki bu örnek dizi üzerinde adım adım gösterilmiştir.

0. adım: başlangıç adımı i=0 olarak ata.

1. adım: dizideki i. sırasından sonraki en küçük sayının yerini bul. Bu dizideki en küçük sayı 1'dir ve 5. sıradadır.

2. adım dizinin i. sırasındaki sayıyı bu en küçük sayı ile yer değiştir: 1,7,2,9,6,5,3,7

3. adım : i'yi bir arttır ve 1. adıma git.

Dolayısıyla 3. adımdan sonra i=1 olacak ve sonra ilk sayı olan 1 atlanaraka kalan sayılar olan 7,2,9,6,5,3,7 sayıları arasından en küçük sayı bulunur. Bu sayı 2'dir ve sırası da 2dir. sıradaki sayı olan yerdeki sayı ile yer değiştirir ve sonuç : 1,2,7,9,6,5,3,7 olarak bulunur.

Artık i değeri 2dir ve bu sayıdan itibaren en küçük sayı 7,9,6,5,3,7 arasında aranır. bulunan 3'ün sırası 6'dır. Bu sayı da sıradaki yerini alır ve sonuç : 1,2,3,9,6,5,7,7 olur. Bu işlem böylece devam eder ve dizinin değişimi aşağıda gösterilmiştir:

1,2,3,5,9,6,7,7

1,2,3,5,6,9,7,7

1,2,3,5,6,7,9,7

1,2,3,5,6,7,7,9

olarak bulunur.

C) -KODLAR-

```
class MergeSort {

     // İki arr alt dizisini birleştirir [].
     // İlk alt dizi arr [l..m]
     // İkinci alt dizi arr [m + 1..r]
    void birleştir(int arr[], int l, int m, int r)
    {
        // Birleştirilecek iki alt dizinin boyutlarını bulma
        int n1 = m - l + 1;
        int n2 = r - m;

        /* Geçici diziler oluşturma */
        int L[] = new int [n1];
        int R[] = new int [n2];

        /* Verileri geçici dizilere kopyala */
        for (int i=0; i<n1; ++i)
            L[i] = arr[l + i];
        for (int j=0; j<n2; ++j)
            R[j] = arr[m + 1+ j];


        /* Geçici dizileri birleştir */

        // Birinci ve ikinci alt dizilerin başlangıç dizinleri
        int i = 0, j = 0;

        // Birleştirme alt dizisi dizisinin başlangıç dizini
        int k = l;
        while (i < n1 && j < n2)
        {
            if (L[i] <= R[j])
            {
                arr[k] = L[i];
                i++;
            }
            else
            {
                arr[k] = R[j];
                j++;
            }
            k++;
        }

        /* Varsa L [] öğesinin kalan öğelerini kopyala */
        while (i < n1)
        {
            arr[k] = L[i];
            i++;
            k++;
        }
```

```java
        /* Varsa R [] öğesinin kalan öğelerini kopyala */
        while (j < n2)
        {
            arr[k] = R[j];
            j++;
            k++;
        }
    }

     // arr [l..r] öğesini kullanarak sıralayan ana işlev
     // birleştir()
    void sort(int arr[], int l, int r)
    {
        if (l < r)
        {
            // Orta noktayı bulma
            int m = (l+r)/2;

            // İlk ve ikinci yarıları sırala
            sort(arr, l, m);
            sort(arr , m+1, r);

            // Sıralanan yarımları birleştir
            birleştir(arr, l, m, r);
        }
    }

    /* N * büyüklüğünde dizi yazdırmak için bir yardımcı program işlevi */
    static void printArray(int arr[])
    {
        int n = arr.length;
        for (int i=0; i<n; ++i)
            System.out.print(arr[i] + " ");
        System.out.println();
    }

    public static void main(String args[])
    {
        int arr[] = {84, 55, 27, 43, 16, 0, 8, 61, 66};

        System.out.println("Dizinin ilk hali");
        printArray(arr);

        MergeSort ob = new MergeSort();
        ob.sort(arr, 0, arr.length-1);

        System.out.println("\nDizinin sıralanmış hali");
        printArray(arr);
    }
}
```

-Ekran Çıktısı-

```
Dizinin ilk hali
84 55 27 43 16 0 8 61 66

Dizinin sıralanmış hali
0 8 16 27 43 55 61 66 84
```