```
import random
import timeit

# A dizisi: 1, 2, 3, ..., 1000 şeklinde sıralı
A = list(range(1, 1001))

# B dizisi: 1000, 999, 998, ..., 1 şeklinde tersten sıralı
B = list(range(1000, 0, -1))

# C dizisi: Rastgele sayılar
C = [random.randint(1, 1000) for _ in range(1000)]
```

Araya Yerleştirme (Insertion Sort):

Ortalama Zaman Karmaşıklığı: O(n^2) En İyi Durum Zaman Karmaşıklığı: O(n) (Dizi sıralıysa) En Kötü Durum Zaman Karmaşıklığı: O(n^2) (Dizi tersten sıralıysa) Kararlı bir sıralama algoritmasıdır. Küçük dizilerde iyi performans gösterir. Genellikle dizi nispeten küçükse veya nispeten sıralı ise tercih edilir.

Kabarcık Sıralama (Bubble Sort):

Ortalama Zaman Karmaşıklığı: O(n^2) En İyi Durum Zaman Karmaşıklığı: O(n) (Dizi zaten sıralıysa) En Kötü Durum Zaman Karmaşıklığı: O(n^2) (Dizi tersten sıralıysa) Kararlı bir sıralama algoritmasıdır. Genellikle çok verimli bir algoritma değildir ve büyük dizilerde kötü performans gösterebilir. Eğitim amaçlı veya küçük veri setleri için kullanışlı olabilir.

Birleştirme Sıralama (Merge Sort):

Ortalama Zaman Karmaşıklığı: O(n log n) En İyi Durum Zaman Karmaşıklığı: O(n log n) En Kötü Durum Zaman Karmaşıklığı: O(n log n) Kararlı bir sıralama algoritmasıdır. Büyük veri setleri için etkili bir algoritmadır. Genellikle hafıza kullanımı daha yüksektir, çünkü ekstra bellek alanı gerektirebilir. Sonuç olarak, performans açısından, Birleştirme Sıralama (Merge Sort) genellikle daha iyi bir seçenektir, özellikle büyük veri setleriyle çalışırken. Araya Yerleştirme ve Kabarcık Sıralama, daha küçük veri setleri veya eğitim amaçlı kullanıldığında tercih edilebilir, ancak büyük veri setlerinde zaman karmaşıklığı açısından etkili değildirler.

Araya Yerleştirme (Insertion Sort)

Algoritma, diziyi sıralanmış ve sıralanmamış iki bölüme böler. Başlangıçta, ilk eleman sıralı bölümde ve geri kalanı sıralanmamış bölümde bulunur. Sıralanmamış bölümdeki her eleman, sıralı bölüme eklenir ve uygun konuma yerleştirilir. Yani, elemanlar sıralı bölüme eklenirken dizinin sıralı olması sağlanır. Bu işlem, sıralanmamış bölümde eleman kalmayana kadar devam eder. Sonuç, sıralanmış bir dizi elde edilir. En iyi durumda, dizinin neredeyse sıralı olduğu durumda çok hızlıdır, ancak en kötü durumda (dizi tersten sıralıysa) çok yavaş olabilir.

```
#Araya Yerleştirme (Insertion Sort) Algoritması:

def insertion_sort(arr):
    for i in range(1, len(arr)):
        key = arr[i]
        j = i - 1
        while j >= 0 and key < arr[j]:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key

A_sorted = A.copy()
B_sorted = B.copy()
C_sorted = C.copy()

insertion_sort(A_sorted)
insertion_sort(B_sorted)
insertion_sort(C_sorted)
print(A_sorted)
print(B_sorted)
print(C_sorted)


time_A = timeit.timeit("insertion_sort(A_sorted)", globals=globals(), number=1)
time_B = timeit.timeit("insertion_sort(B_sorted)", globals=globals(), number=1)
time_C = timeit.timeit("insertion_sort(C_sorted)", globals=globals(), number=1)

print("A dizisinin sıralama süresi:", time_A, "saniye")
print("B dizisinin sıralama süresi:", time_B, "saniye")
print("C dizisinin sıralama süresi:", time_C, "saniye")
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
[2, 3, 5, 5, 6, 6, 7, 7, 9, 13, 15, 17, 18, 20, 21, 21, 21, 22, 23, 24, 25, 27, 28, 28, 29, 30, 31, 31, 31, 31, 31, 32, 32, 34, 34,
A dizisinin sıralama süresi: 0.00020005300007142068 saniye
```

```
B dizisinin sıralama süresi: 0.00021201399999881687 saniye
C dizisinin sıralama süresi: 0.0002051449998816679 saniye
```

Kabarcık Sıralama (Bubble Sort) Algoritma, her iki komşu elemanı karşılaştırır ve yanlış sıralanmışsa yer değiştirir. Başlangıçta, en büyük eleman en sona yerleşir. Ardışık geçişlerde, bir sonraki en büyük elemanın sona kaydığını görürüz. Bu işlem, dizide hiç yer değiştirme olmadığında sıralama tamamlanır. Yani, herhangi bir geçişte yer değiştirme yapılmazsa, dizi zaten sıralıdır. Kabarcık sıralama, her zaman n^2 karşılaştırma yapar, bu nedenle büyük diziler için etkisizdir.

```
#Kabarcık Sıralama (Bubble Sort) Algoritması:

def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]

A_sorted = A.copy()
B_sorted = B.copy()
C_sorted = C.copy()

bubble_sort(A_sorted)
bubble_sort(B_sorted)
bubble_sort(C_sorted)

print(A_sorted)
print(B_sorted)
print(C_sorted)


time_A = timeit.timeit("bubble_sort(A_sorted)", globals=globals(), number=1)
time_B = timeit.timeit("bubble_sort(B_sorted)", globals=globals(), number=1)
time_C = timeit.timeit("bubble_sort(C_sorted)", globals=globals(), number=1)

print("A dizisinin sıralama süresi:", time_A, "saniye")
print("B dizisinin sıralama süresi:", time_B, "saniye")
print("C dizisinin sıralama süresi:", time_C, "saniye")
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
[2, 3, 5, 5, 6, 6, 7, 7, 9, 13, 15, 17, 18, 20, 21, 21, 21, 22, 23, 24, 25, 27, 28, 28, 29, 30, 31, 31, 31, 31, 31, 32, 32, 34, 34,
A dizisinin sıralama süresi: 0.05224129399994126 saniye
B dizisinin sıralama süresi: 0.06207041800007573 saniye
C dizisinin sıralama süresi: 0.053904855999917345 saniye
```

Birleştirme Sıralama (Merge Sort) Algoritma, "böl ve fethet" stratejisi kullanır. Diziyi iki eşit parçaya böler, her iki parçayı ayrı ayrı sıralar ve ardından bu iki sıralı parçayı birleştirir. Bölme işlemi tekrarlanarak diziler daha küçük parçalara bölünür ve her iki alt dizi sıralanır. Birleştirme işlemi sıralı alt dizileri birleştirir ve büyük bir sıralı dizi oluşturur. Birleştirme sıralaması, her zaman O(n log n) zaman karmaşıklığına sahiptir ve büyük veri setleriyle iyi başa çıkar.

```
#Birleştirme Sıralama (Merge Sort) Algoritması:

def merge_sort(arr):
    if len(arr) > 1:
        mid = len(arr) // 2
        L = arr[:mid]
        R = arr[mid:]

        merge_sort(L)
        merge_sort(R)

        i = j = k = 0

        while i < len(L) and j < len(R):
            if L[i] < R[j]:
                arr[k] = L[i]
                i += 1
            else:
                arr[k] = R[j]
                j += 1
            k += 1

        while i < len(L):
            arr[k] = L[i]
            i += 1
            k += 1
```

```
        while j < len(R):
            arr[k] = R[j]
            j += 1
            k += 1

A_sorted = A.copy()
B_sorted = B.copy()
C_sorted = C.copy()

merge_sort(A_sorted)
merge_sort(B_sorted)
merge_sort(C_sorted)

print(A_sorted)
print(B_sorted)
print(C_sorted)


time_A = timeit.timeit("merge_sort(A_sorted)", globals=globals(), number=1)
time_B = timeit.timeit("merge_sort(B_sorted)", globals=globals(), number=1)
time_C = timeit.timeit("merge_sort(C_sorted)", globals=globals(), number=1)

print("A dizisinin sıralama süresi:", time_A, "saniye")
print("B dizisinin sıralama süresi:", time_B, "saniye")
print("C dizisinin sıralama süresi:", time_C, "saniye")
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
[2, 3, 5, 5, 6, 6, 7, 7, 9, 13, 15, 17, 18, 20, 21, 21, 21, 22, 23, 24, 25, 27, 28, 28, 29, 30, 31, 31, 31, 31, 31, 32, 32, 34, 34,
A dizisinin sıralama süresi: 0.0028348729999834177 saniye
B dizisinin sıralama süresi: 0.004312812000080157 saniye
C dizisinin sıralama süresi: 0.0028705269999136362 saniye
```