اجاف في الفرز خوارزميات

تعد الفرز السريع وخوارزمية الدمج من أكثر خوارزميات الفرز شيوعًا واستخدامًا على نطاق واسع. كلاهما يستخدم نهج فرق تسد، ولكن لديهم نقاط قوة وضعف مختلفة. في هذه المقالة، سنناقش كل خوارزمية بالتفصيل، بما في ذلك تنفيذها وتعقيدها الزمني ومتى يجب استخدام كل منها.

1. الفرز السريع: نظرة عامة

الفرز السريع هو خوارزمية فرز فعالة للغاية تعمل عن طريق اختيار عنصر "محوري" من المصفوفة وتقسيم العناصر الأخرى إلى مصفوفتين فرعيتين، اعتمادًا على ما إذا كانت أقل من المحور أو أكبر منه. ثم يتم فرز المصفوفات الفرعية بشكل متكرر.

تنفيذ

```
public class QuickSort {
    public static void quickSort(int[] arr, int low, int high) {
        if (low < high) {
            int pi = partition(arr, low, high);
            quickSort(arr, low, pi - 1);
            quickSort(arr, pi + 1, high);
        }
    }

    private static int partition(int[] arr, int low, int high) {
        int pivot = arr[high];
        int i = low - 1;
        for (int j = low; j < high; j++) {
            if (arr[j] <= pivot) {
                i++;
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
        int temp = arr[i + 1];
        arr[i + 1] = arr[high];
        arr[high] = temp;
        return i + 1;
    }

    public static void main(String[] args) {
        int[] arr = {10, 7, 8, 9, 1, 5};
```

```
        quickSort(arr, 0, arr.length - 1);
        System.out.println("Sorted: " + java.util.Arrays.toString(arr));
    }
}
```

الخروج: `Sorted: [1, 5, 7, 8, 9, 10]`

2. بيترت: جاماندان الا بيترت رقتسم

بيترت جاماندان الا مسقي، يوم الم مصفوفة، بيترتب ين صنال امه عم جي ورة مرة. أخرى هنإ رقتسم نمض ي وة ديقع ة ينمز [illegible] [illegible].

ذيفنت [illegible]

```
public class MergeSort {
    public static void mergeSort(int[] arr, int l, int r) {
        if (l < r) {
            int m = (l + r) / 2;
            mergeSort(arr, l, m);
            mergeSort(arr, m + 1, r);
            merge(arr, l, m, r);
        }
    }

    private static void merge(int[] arr, int l, int m, int r) {
        int n1 = m - l + 1, n2 = r - m;
        int[] L = new int[n1], R = new int[n2];
        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];
        int i = 0, j = 0, k = l;
        while (i < n1 && j < n2) arr[k++] = (L[i] <= R[j]) ? L[i++] : R[j++];
        while (i < n1) arr[k++] = L[i++];
        while (j < n2) arr[k++] = R[j++];
    }

    public static void main(String[] args) {
        int[] arr = {12, 11, 13, 5, 6, 7};
        mergeSort(arr, 0, arr.length - 1);
        System.out.println("Sorted: " + java.util.Arrays.toString(arr));
```

```
    }
}
```

الخروج: Sorted: [5, 6, 7, 11, 12, 13]

3. بيترت: ةعاقفلا طيسب نكلو ءيطب

بيترت ةعاقفلا موقي ريغتب رصانعلا ةروازجملا لكشب رركتم إذا تناك جراخ. بيترتلا هنإ □□□□□ نكل ولو لهس. مهفلا

ذيفنت □□□□

```
public class BubbleSort {
    public static void bubbleSort(int[] arr) {
        int n = arr.length;
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (arr[j] > arr[j + 1]) {
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }
    }

    public static void main(String[] args) {
        int[] arr = {64, 34, 25, 12, 22, 11, 90};
        bubbleSort(arr);
        System.out.println("Sorted: " + java.util.Arrays.toString(arr));
    }
}
```

الخروج: Sorted: [11, 12, 22, 25, 34, 64, 90]