Fiche Algo : Problèmes types débutant

C'est une super idée d'aborder les algorithmes sous un angle pratique, avec des problèmes types qu'on retrouve souvent dans les tests techniques ! Voici une fiche avec des exercices d'algorithmes, allant du plus facile au plus complexe, accompagnés d'explications et de solutions en JavaScript et Java.

Fiche Algo : Problèmes types style **Codewars** (Développeur Junior)

1. **Addition de Deux Nombres**

Difficulté : Facile

Problème :

Écris une fonction qui prend deux nombres en entrée et retourne leur somme.

Exemple :

- `sum(1, 2)` doit retourner `3`.

Solution en JavaScript :

```javascript
function sum(a, b) {
    return a + b; // Renvoie la somme des deux nombres
}

console.log(sum(1, 2)); // 3
```

Solution en Java :

```java
public class Main {
    public static int sum(int a, int b) {
        return a + b; // Renvoie la somme des deux nombres
    }

    public static void main(String[] args) {
        System.out.println(sum(1, 2)); // 3
    }
}
```

---

2. **Inverser une Chaîne de Caractères**

Difficulté : Facile

Problème :

Écris une fonction qui prend une chaîne de caractères et retourne cette chaîne inversée.

Exemple :

- `reverse("hello")` doit retourner `"olleh"`.

Solution en JavaScript :

```
function reverse(str) {
    return str.split('').reverse().join(''); // Split en tableau, reverse puis join pour reformer la chaîne
}

console.log(reverse("hello")); // "olleh"
```

Solution en Java :

```
public class Main {
    public static String reverse(String str) {
        return new StringBuilder(str).reverse().toString(); // Utilisation de StringBuilder pour inverser
    }

    public static void main(String[] args) {
        System.out.println(reverse("hello")); // "olleh"
    }
}
```

---

3. Trouver le Nombre Maximum dans un Tableau

Difficulté : Moyenne

Problème :

Écris une fonction qui prend un tableau de nombres et renvoie le nombre maximum.

Exemple :

- `findMax([1, 5, 3, 9, 2])` doit retourner `9`.

Solution en JavaScript :

```javascript
function findMax(arr) {
    return Math.max(...arr); // Utilisation de l'opérateur de décomposition pour passer le tableau à Math.max
}

console.log(findMax([1, 5, 3, 9, 2])); // 9
```

Solution en Java :

```java
import java.util.Arrays;

public class Main {
    public static int findMax(int[] arr) {
        return Arrays.stream(arr).max().getAsInt(); // Utilisation de Stream pour trouver le max
    }

    public static void main(String[] args) {
        int[] arr = {1, 5, 3, 9, 2};
        System.out.println(findMax(arr)); // 9
    }
}
```

---

4. **Somme des Nombres Pairs**

Difficulté : Moyenne

Problème :

Écris une fonction qui prend un tableau de nombres et retourne la somme des nombres pairs.

Exemple :

- `sumEven([1, 2, 3, 4, 5])` doit retourner `6`.

Solution en JavaScript :

```javascript
function sumEven(arr) {
    return arr.filter(num => num % 2 === 0).reduce((acc, curr) => acc + curr, 0); // Filtre les pairs et fait la somme
}

console.log(sumEven([1, 2, 3, 4, 5])); // 6
```

Solution en Java :

```java
import java.util.Arrays;

public class Main {
    public static int sumEven(int[] arr) {
        return Arrays.stream(arr).filter(num -> num % 2 == 0).sum(); // Filtre les
pairs et fait la somme
    }

    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5};
        System.out.println(sumEven(arr)); // 6
    }
}
```

---

5. **Déterminer si une Chaîne est un Palindrome**

Difficulté : Moyenne

Problème :

Écris une fonction qui prend une chaîne et détermine si elle est un palindrome (se lit de la même manière à l'envers).

Exemple :

- `isPalindrome("racecar")` doit retourner `true`.

Solution en JavaScript :

```javascript
function isPalindrome(str) {
    const reversed = str.split('').reverse().join(''); // Inverse la chaîne
    return str === reversed; // Vérifie si la chaîne est égale à sa version inversée
}

console.log(isPalindrome("racecar")); // true
```

Solution en Java :

```java
public class Main {
    public static boolean isPalindrome(String str) {
        String reversed = new StringBuilder(str).reverse().toString(); // Inverse la
chaîne
        return str.equals(reversed); // Vérifie si la chaîne est égale à sa version
inversée
    }

    public static void main(String[] args) {
```

```
        System.out.println(isPalindrome("racecar")); // true
    }
}
```

---

6. **FizzBuzz**

Difficulté : Moyenne

Problème :

Écris une fonction qui prend un nombre et renvoie :

- "Fizz" si le nombre est divisible par 3,
- "Buzz" si le nombre est divisible par 5,
- "FizzBuzz" si le nombre est divisible par 3 et 5,
- Le nombre sinon.

Exemple :

- `fizzBuzz(15)` doit retourner `"FizzBuzz"`.

Solution en JavaScript :

```javascript
function fizzBuzz(num) {
    if (num % 3 === 0 && num % 5 === 0) return "FizzBuzz";
    if (num % 3 === 0) return "Fizz";
    if (num % 5 === 0) return "Buzz";
    return num;
}

console.log(fizzBuzz(15)); // "FizzBuzz"
```

Solution en Java :

```java
public class Main {
    public static String fizzBuzz(int num) {
        if (num % 3 == 0 && num % 5 == 0) return "FizzBuzz";
        if (num % 3 == 0) return "Fizz";
        if (num % 5 == 0) return "Buzz";
        return String.valueOf(num);
    }

    public static void main(String[] args) {
        System.out.println(fizzBuzz(15)); // "FizzBuzz"
    }
}
```

---

7. Trier un Tableau en Ordre Croissant

Difficulté : Moyenne

Problème :

Écris une fonction qui trie un tableau de nombres en ordre croissant.

Exemple :

- `sortArray([3, 1, 4, 1, 5])` doit retourner `[1, 1, 3, 4, 5]`.

Solution en JavaScript :

```javascript
function sortArray(arr) {
    return arr.sort((a, b) => a - b); // Trie le tableau en ordre croissant
}

console.log(sortArray([3, 1, 4, 1, 5])); // [1, 1, 3, 4, 5]
```

Solution en Java :

```java
import java.util.Arrays;

public class Main {
    public static int[] sortArray(int[] arr) {
        Arrays.sort(arr); // Trie le tableau en ordre croissant
        return arr;
    }

    public static void main(String[] args) {
        int[] arr = {3, 1, 4, 1, 5};
        System.out.println(Arrays.toString(sortArray(arr))); // [1, 1, 3, 4, 5]
    }
}
```

Voici quelques exercices supplémentaires, y compris celui que tu as mentionné, pour t'aider à renforcer tes compétences en algorithmes. Chaque problème est accompagné de solutions en JavaScript et en Java.

---

8. Trier les Nombres Impairs tout en Gardant les Pairs à leur Place

Difficulté : Moyenne

Problème :

Tu recevras un tableau de nombres. Tu dois trier les nombres impairs par ordre croissant tout en laissant les nombres pairs à leurs positions d'origine.

Exemples :

- `[7, 1]` doit retourner `[1, 7]`.
- `[5, 8, 6, 3, 4]` doit retourner `[3, 8, 6, 5, 4]`.
- `[9, 8, 7, 6, 5, 4, 3, 2, 1, 0]` doit retourner `[1, 8, 3, 6, 5, 4, 7, 2, 9, 0]`.

Solution en JavaScript :

```
function oddSort(arr) {
    const oddNumbers = arr.filter(num => num % 2 !== 0).sort((a, b) => a - b); // Filtre et trie les impairs
    return arr.map(num => (num % 2 !== 0 ? oddNumbers.shift() : num)); // Remplace les impairs triés dans le tableau original
}

console.log(oddSort([7, 1])); // [1, 7]
console.log(oddSort([5, 8, 6, 3, 4])); // [3, 8, 6, 5, 4]
console.log(oddSort([9, 8, 7, 6, 5, 4, 3, 2, 1, 0])); // [1, 8, 3, 6, 5, 4, 7, 2, 9, 0]
```

Solution en Java :

```
import java.util.Arrays;

public class Main {
    public static int[] oddSort(int[] arr) {
        int[] oddNumbers = Arrays.stream(arr).filter(num -> num % 2 != 0).sorted().toArray(); // Filtre et trie les impairs
        int index = 0;
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] % 2 != 0) {
                arr[i] = oddNumbers[index++]; // Remplace les impairs triés dans le tableau original
            }
        }
        return arr;
    }

    public static void main(String[] args) {
        System.out.println(Arrays.toString(oddSort(new int[]{7, 1}))); // [1, 7]
        System.out.println(Arrays.toString(oddSort(new int[]{5, 8, 6, 3, 4}))); // [3, 8, 6, 5, 4]
        System.out.println(Arrays.toString(oddSort(new int[]{9, 8, 7, 6, 5, 4, 3, 2, 1, 0}))); // [1, 8, 3, 6, 5, 4, 7, 2, 9, 0]
    }
}
```

---

9. **Compter les Voyelles dans une Chaîne**

Difficulté : Moyenne

Problème :

Écris une fonction qui compte le nombre de voyelles dans une chaîne de caractères.

Exemples :

- `countVowels("hello")` doit retourner `2`.
- `countVowels("algorithm")` doit retourner `3`.

Solution en JavaScript :

```
function countVowels(str) {
    return (str.match(/[aeiou]/gi) || []).length; // Utilise une expression
régulière pour compter les voyelles
}

console.log(countVowels("hello")); // 2
console.log(countVowels("algorithm")); // 3
```

Solution en Java :

```
public class Main {
    public static int countVowels(String str) {
        return (int) str.chars().filter(ch -> "aeiouAEIOU".indexOf(ch) !=
-1).count(); // Filtre les voyelles et compte
    }

    public static void main(String[] args) {
        System.out.println(countVowels("hello")); // 2
        System.out.println(countVowels("algorithm")); // 3
    }
}
```

---

10. **Trouver le Mot le Plus Long dans une Chaîne**

Difficulté : Moyenne

Problème :

Écris une fonction qui prend une chaîne de mots et retourne le mot le plus long.

Exemples :

- `longestWord("Hello world")` doit retourner `"Hello"`.
- `longestWord("I love programming")` doit retourner `"programming"`.

Solution en JavaScript :

```javascript
function longestWord(str) {
    return str.split(' ').reduce((a, b) => a.length >= b.length ? a : b); // Utilise reduce pour trouver le mot le plus long
}

console.log(longestWord("Hello world")); // "Hello"
console.log(longestWord("I love programming")); // "programming"
```

Solution en Java :

```java
public class Main {
    public static String longestWord(String str) {
        String[] words = str.split(" "); // Sépare la chaîne en mots
        String longest = "";
        for (String word : words) {
            if (word.length() > longest.length()) {
                longest = word; // Met à jour le mot le plus long
            }
        }
        return longest;
    }

    public static void main(String[] args) {
        System.out.println(longestWord("Hello world")); // "Hello"
        System.out.println(longestWord("I love programming")); // "programming"
    }
}
```

---

11. Calculer le Factoriel d'un Nombre

Difficulté : Moyenne

Problème :

Écris une fonction qui calcule le factoriel d'un nombre donné.

Exemples :

- `factorial(5)` doit retourner `120`.
- `factorial(0)` doit retourner `1`.

Solution en JavaScript :

```javascript
function factorial(n) {
    if (n === 0) return 1; // Le factoriel de 0 est 1
    return n * factorial(n - 1); // Calcul récursif
}

console.log(factorial(5)); // 120
console.log(factorial(0)); // 1
```

Solution en Java :

```java
public class Main {
    public static int factorial(int n) {
        if (n == 0) return 1; // Le factoriel de 0 est 1
        return n * factorial(n - 1); // Calcul récursif
    }

    public static void main(String[] args) {
        System.out.println(factorial(5)); // 120
        System.out.println(factorial(0)); // 1
    }
}
```

Conclusion :

Cette fiche d'algorithmes te donne des exercices pratiques qui reflètent des problèmes typiques rencontrés lors de tests techniques. En travaillant sur ces exemples, tu pourras améliorer ta compréhension des concepts algorithmiques et renforcer tes compétences en codage.