URL VIDEOS

Explicación enunciado del problema: https://youtu.be/iRfOhe_foTk?feature=shared
Explicación código: https://youtu.be/G2bYQaAcJtQ?feature=shared
Análisis de complejidad: https://youtu.be/gQHAwNiGNSc?feature=shared

ENUNCIADO DEL PROBLEMA

Hay n alumnos en una clase escolar, la calificación del alumno i-ésimo en Codehorses es ai. Debes formar un equipo compuesto por k alumnos (1≤k≤n) de tal manera que las calificaciones de todos los miembros del equipo sean distintas.

Si es imposible formar un equipo adecuado, imprime «NO» (sin comillas). De lo contrario, imprime «SÍ» y, a continuación, imprime k números distintos que deben ser los índices de los alumnos del equipo que formes. Si hay varias respuestas, imprime cualquiera de ellas.

Entrada

La primera línea contiene dos números enteros n y k (1≤k≤n≤10^8): el número de estudiantes y el tamaño del equipo que debes formar.

La segunda línea contiene n números enteros a1, a2, …, an (1≤ai≤10^4), donde ai es la calificación del estudiante i-ésimo.

Salida

Si es imposible formar un equipo adecuado, imprime «NO» (sin comillas). De lo contrario, imprime «YES» y, a continuación, imprime k números enteros distintos del 1 al n, que deben ser los índices de los estudiantes del equipo que formes. Todas las calificaciones de los estudiantes del equipo deben ser distintas. Puedes imprimir los índices en cualquier orden. Si hay varias respuestas, imprime cualquiera de ellas.

Supón que los estudiantes están numerados del 1 al n.

URL DE GITLAB O GITHUB PROYECTO EN FORMA PRIVADA, USUARIO madarme@ufps.edu.co, con rol maintener

https://gitlab.com/ejercicios-proyecto-ada/ejercicio-8-diverse-team

MÉTODO 1

```
public int[] metodoCandido(int n, int k, int[] ratings) {
    int[] indices = new int[k];
    int contador = 0;
    for (int i = 0; i < n; i++) {
        boolean existe = false;
        for (int j = 0; j  < contador; j++) {
            if (ratings[i] == ratings[indices[j] - 1]) {
                existe = true;
                break;
            }
        }
        if (!existe && contador < k) {
            indices[contador] = i + 1;
            contador++;
        }
        if (contador == k) {
            break;
        }
    }
    if (contador < k) {
        return null;
    } else {
        return indices;
    }
}
```

Eficacia	El método es eficaz para las entradas de números enteros.
	Instancia 1: Número: 5 3 15 13 15 15 12 Resultado de la invocación: YES 1 2 5
	Instancia 2: Número: 1000000 1000 Arreglo generado de manera random Resultado de la invocación: YES Cantidad muy grande de índices
	Instancia 3: Numero: 100000000 10000 Arreglo generado de manera random Resultado de la invocación: YES Cantidad muy grande de índices
Eficiencia	El método en entradas pequeñas es eficiente, sin embargo no lo suficiente ya que itera sobre el arreglo de ratings y el de índices, haciendo que no sea eficiente en números grandes.
	Instancia 1:

	Número: 5 3 15 13 15 15 12 Resultado de la invocación: YES 1 2 5 **Tiempo del proceso: 107 microsegundos**
	Instancia 2: Número: 1000000 1000 Arreglo generado de manera random Resultado de la invocación: YES Cantidad muy grande de índices **Tiempo del proceso: 6107 microsegundos**
	Instancia 3: Numero: 100000000 10000 Arreglo generado de manera random Resultado de la invocación: YES Cantidad muy grande de índices **Tiempo del proceso: 240076 microsegundos**
Correctitud	El algoritmo es correcto, ya que da un resultado esperado para todas las entradas posibles
	Instancia 1: Número: 5 3

	15 13 15 15 12 Resultado esperado: YES Resultado método cándido: YES
	Instancia 2: Número: 1000000 1000 Arreglo generado de manera random Resultado esperado: YES Resultado método cándido: YES
	Instancia 3: Numero: 100000000 10000 Arreglo generado de manera random Resultado esperado: YES Resultado método cándido: YES
Completitud	El método es completo, ya que la lógica usada para hallar los equipos es correcta, aunque es lento asegura siempre un resultado correcto, si es que este existe
	Instancia 1: Número: 5 3 15 13 15 15 12 Resultado del método cándido: YES 1 2 5

	Instancia 2: Número: 1000000 1000 Arreglo generado de manera random Resultado del método cándido:: YES Cantidad muy grande de índices
	Instancia 3: Numero: 100000000 10000 Arreglo generado de manera random Resultado del método cándido: YES Cantidad muy grande de índices

MÉTODO 2

```
public int[] metodoOptimo(int n, int k, int[] ratings) {
    TablaHash<Integer, Integer> tabla = new TablaHash<>(10007);
    int[] resultado = new int[k];
    int contador = 0;

    for (int i = 0; i < n && contador < k; i++) {
        if (!tabla.esta(ratings[i])) {
            tabla.insertar(ratings[i], i + 1);
            resultado[contador++] = i + 1;
        }
    }
    if (contador < k) {
        return null;
    } else {
        return resultado;
    }
}
```

Eficacia	El método es eficaz para todas las entradas posibles.
	Instancia 1: Número: 5 3 15 13 15 15 12 Resultado de la invocación: YES 1 2 5
	Instancia 2: Número: 1000000 1000 Arreglo generado de manera random Resultado de la invocación: YES Cantidad muy grande de índices.
	Instancia 3: Numero: 100000000 10000 Arreglo generado de manera random Resultado de la invocación: YES Cantidad muy grande de índices.
Eficiencia	El método es eficiente ya que al utilizar una estructura de datos es más eficiente en las entradas grandes
	Instancia 1: Número: 5 3

	15 13 15 15 12 Resultado de la invocación: YES 1 2 5 **Tiempo del proceso: 3585 microsegundos**
	Instancia 2: Número: 1000000 1000 Arreglo generado de manera random Resultado de la invocación: YES Cantidad muy grande de índices **Tiempo del proceso: 2852 microsegundos**
	Instancia 3: Numero: 100000000 10000 Arreglo generado de manera random Resultado de la invocación: YES Cantidad muy grande de índices **Tiempo del proceso: 26021 microsegundos**
Correctitud	El método devuelve salidas esperadas para todas las entradas posibles.
	Instancia 1: Número: 5 3 15 13 15 15 12

	Resultado de la invocación: YES 1 2 5 Resultado esperado:YES 1 2 5
	Instancia 2: Número: 1000000 1000 Arreglo generado de manera random Resultado de la invocación: YES Cantidad muy grande de índices Resultado esperado:YES Cantidad muy grande de índices
	Instancia 3: Numero: 100000000 10000 Arreglo generado de manera random Resultado de la invocación: YES Cantidad muy grande de índices Resultado esperado:YES Cantidad muy grande de índices
Completitud	El método es completo ya que con la lógica usada se asegura una respuesta correcta si existe.
	Instancia 1:

	Número: 5 3 15 13 15 15 12 Resultado del método óptimo: YES 1 2 5
	Instancia 2: Número: 1000000 1000 Arreglo generado de manera random Resultado del método óptimo:: YES Cantidad muy grande de índices
	Instancia 3: Numero: 100000000 10000 Arreglo generado de manera random Resultado del método óptimo: YES Cantidad muy grande de índices