

# **UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE**

**DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN**

**ESTRUCTURA DE DATOS**



**BÚSQUEDA BINARIA EJERCICIO 2**

**NOMBRE:** QUISHPE CHAVEZ DENISSE PAULINA

**NRC:** 29852

**FECHA:** 02/12/2025

**DOCENTE:** SOLIS ACOSTA EDGAR FERNANDO

## DISTRIBUCIÓN DE AGUA EN TANQUES

Una ciudad tiene  $N$  casas ubicadas en una línea recta (cada casa en una posición diferente en metros). Se van a instalar  $K$  tanques de agua, y cada tanque puede abastecer a todas las casas que estén a una distancia máxima  $D$  metros de él.

Objetivo: Encontrar la distancia mínima  $D$  que permite que todas las casas tengan acceso a agua con los  $K$  tanques disponibles.

Entrada:

- $N$ : Número de casas
- $K$ : Número de tanques disponibles
- Posiciones de las casas (en metros)

Salida: La distancia mínima  $D$

```
#include <iostream>
using namespace std;
long long* posiciones;
int N, K;
// Ordenar array con Bubble Sort
void ordenar() {
    for (int i = 0; i < N - 1; i++) {
        for (int j = 0; j < N - i - 1; j++) {
            if (posiciones[j] > posiciones[j + 1]) {
                long long temp = posiciones[j];
                posiciones[j] = posiciones[j + 1];
                posiciones[j + 1] = temp;
            }
        }
    }
}
```

```

    }
}
}

// BÚSQUEDA BINARIA: Verificar si con distancia D podemos cubrir todas las casas
bool puedeCubrir(long long D) {
    int tanquesUsados = 1;
    long long ultimoTanque = posiciones[0]; // Primer tanque en primera casa
    for (int i = 1; i < N; i++) {
        // Si la casa está muy lejos del último tanque
        if (posiciones[i] - ultimoTanque > D) {
            tanquesUsados++;
            ultimoTanque = posiciones[i]; // Colocamos nuevo tanque aquí
            if (tanquesUsados > K) {
                return false; // Necesitamos más tanques
            }
        }
    }
    return true;
}

int main() {
    cout << "=== DISTRIBUCION DE TANQUES DE AGUA ===\n";
    cout << "Numero de casas: ";
    cin >> N;
    cout << "Numero de tanques: ";
    cin >> K;
    posiciones = new long long[N];
    cout << "Posiciones de las casas (metros):\n";
    for (int i = 0; i < N; i++) {

```

```

    cout << " Casa " << (i + 1) << ": ";
    cin >> posiciones[i];
}
ordenar(); // Ordenar posiciones
// BÚSQUEDA BINARIA sobre la distancia D
// Rango: [0, distancia entre primera y última casa]
long long left = 0;
long long right = posiciones[N - 1] - posiciones[0];
long long respuesta = right;
cout << "\n--- BUSQUEDA BINARIA ---\n";
cout << "Rango: [" << left << ", " << right << "]\n\n";
int iter = 0;
while (left <= right) {
    long long mid = left + (right - left) / 2;
    iter++;
    cout << "Iter " << iter << ": Probando D=" << mid << " -> ";
    if (puedeCubrir(mid)) {
        cout << "SI cubre\n";
        respuesta = mid;
        right = mid - 1; // Intentar con menos distancia
    } else {
        cout << "NO cubre\n";
        left = mid + 1; // Necesitamos más distancia
    }
}
cout << "\nDistancia minima: " << respuesta << " metros\n";
delete[] posiciones;
return 0; }

```