## Arithmetic Sequence in C++

```
#include <iostream>
#include <vector>
#include <unordered_set>
#include <algorithm>
#include <climits>

using namespace std;

bool isArithmeticSequence(const vector<int>& arr) {
    if (arr.size() <= 1) {
        return true;
    }

    int minVal = INT_MAX;
    int maxVal = INT_MIN;
    unordered_set<int> elements;

    for (int val : arr) {
        minVal = min(val, minVal);
        maxVal = max(val, maxVal);
        elements.insert(val);
    }

    int d = (maxVal - minVal) / (arr.size() - 1);

    for (size_t i = 0; i < arr.size(); ++i) {
        int ai = minVal + i * d;
        if (elements.find(ai) == elements.end()) {
            return false;
        }
    }

    return true;
}

int main() {
    vector<int> arr = {17, 9, 5, 29, 1, 25, 13, 37, 21, 33};
    cout << (isArithmeticSequence(arr) ? "true" :
"false") << endl;

    return 0;
}
```

**Dry Run**

**Input:**

arr = {17, 9, 5, 29, 1, 25, 13, 37, 21, 33}

Here is a step-by-step dry run of your C++ code, focusing on loop iterations and index-wise updates:

**Step-by-Step Execution Table**

**First Loop (Finding `minVal`, `maxVal`, and Filling `unordered_set`)**

| Index (`i`) | Current `arr[i]` | Updated `minVal` | Updated `maxVal` | Updated `elements` |
|---|---|---|---|---|
| 0 | 17 | 17 | 17 | {17} |
| 1 | 9 | 9 | 17 | {9, 17} |
| 2 | 5 | 5 | 17 | {5, 9, 17} |
| 3 | 29 | 5 | 29 | {5, 9, 17, 29} |
| 4 | 1 | 1 | 29 | {1, 5, 9, 17, 29} |
| 5 | 25 | 1 | 29 | {1, 5, 9, 17, 25, 29} |
| 6 | 13 | 1 | 29 | {1, 5, 9, 13, 17, 25, 29} |
| 7 | 37 | 1 | 37 | {1, 5, 9, 13, 17, 25, 29, 37} |
| 8 | 21 | 1 | 37 | {1, 5, 9, 13, 17, 21, 25, 29, 37} |
| 9 | 33 | 1 | 37 | {1, 5, 9, 13, 17, 21, 25, 29, 33, 37} |

- After this loop:
  - `minVal = 1`
  - `maxVal = 37`
  - `elements = {1, 5, 9, 13, 17, 21, 25, 29, 33, 37}`
  - `d = (37 - 1) / (10 - 1) = 4`

**Second Loop (Verifying Arithmetic Sequence)**

| Index (`i`) | Expected Value `ai = minVal + i * d` | Check in `elements` | Result |
|---|---|---|---|
| 0 | `1 + 0*4 = 1` | ✓ Found in `{1, 5, 9, 13, 17, 21, 25, 29, 33, 37}` | Continue |
| 1 | `1 + 1*4 = 5` | ✓ Found | Continue |
| 2 | `1 + 2*4 = 9` | ✓ Found | Continue |
| 3 | `1 + 3*4 = 13` | ✓ Found | Continue |
| 4 | `1 + 4*4 = 17` | ✓ Found | Continue |
| 5 | `1 + 5*4 = 21` | ✓ Found | Continue |
| 6 | `1 + 6*4 =` | ✓ Found | Continue |

| Index (`i`) | Expected Value `ai = minVal + i * d` | Check in `elements` | Result |
|---|---|---|---|
| | `25` | | |
| 7 | `1 + 7*4 = 29` | ✓ Found | Continue |
| 8 | `1 + 8*4 = 33` | ✓ Found | Continue |
| 9 | `1 + 9*4 = 37` | ✓ Found | Continue |

- Since all expected values exist in `elements`, the function returns **`true`**.

Output:
true