# Minimize The Heights I

| | |
|---|---|
| solved by | Senan |
| Platform | GeeksForGeeks |
| difficulty | Medium |
| tags | Sorting |
| language | C++ |
| solved on | @23/11/2024 |
| link | https://www.geeksforgeeks.org/problems/minimize-the-heights-i/1 |
| Completion | ☑ |

## Intuition

The task is to minimize the difference between the maximum and minimum heights of a modified array, where each height can be increased or decreased by `k`. The approach uses sorting and iterates over potential adjustments to maintain a balance between minimizing the range and preserving valid height adjustments.

## Approach

1. **Sort the array**:

   Sorting simplifies determining the smallest and largest values after modifying the array. It ensures the order of elements is preserved.

2. **Calculate the initial range**:

   Start with the difference between the maximum (`arr[n-1]`) and minimum (`arr[0]`) values of the array as the baseline answer.

3. **Iterate through the array**:

   For each element in the array (except the first), consider it as the potential boundary for the smallest (`mini`) and largest (`maxi`) values after adding or subtracting `k`:

   - `mini`: Minimum of the smallest element increased by `k` (`arr[0] + k`) or the current element decreased by `k` (`arr[i] - k`).
   - `maxi`: Maximum of the largest element decreased by `k` (`arr[n-1] - k`) or the previous element increased by `k` (`arr[i-1] + k`).
   - Update the `answer` to the smaller value between the current `answer` and the difference `maxi - mini`.

4. **Return the result**:

   After processing all potential boundaries, the `answer` contains the minimized maximum difference.

## Complexity

### Time Complexity:

- **Sorting**: **O(nlogn)**, where  is the size of the array.
- **Iteration**:  **O(n)**, for a single traversal to calculate the minimum difference.
- **Total**: **O(nlogn)**

### Space Complexity:

- **In-place modifications**: The array is sorted in place, so no additional space is used apart from a few variables.
- **Total**: **O(1)**

# Code

```
class Solution {
  public:
    int getMinDiff(int k, vector<int> &arr) {
        int n = arr.size();
        sort(arr.begin(), arr.end());

        int mini = arr[0];
        int maxi = arr[n - 1];
        int answer = maxi - mini;

        for (int i = 1; i < n; i++) {
            mini = min(arr[0] + k, arr[i] - k);
            maxi = max(arr[n - 1] - k, arr[i - 1] + k);
            answer = min(answer, maxi - mini);
        }

        return answer;
    }
};
```