Minimum removals from array to make max – min <= K

> 给定数组arr（长度N）与整数K，求需要从数组中移除的数的最小个数，使得移除后的数组满足MAX-MIN ⩽ K

dp[i][j]表示使得arr[j] - arr[i] ⩽ K所需要移除的元素数量，（i, ...,j）表示移除元素后的剩余元素的索引

minRemoves函数返回使得（i,...,j）满足arr[j] - arr[i] ⩽ K所需要移除的元素的数量

对于dp[i][j]来说，有两种途径：

1. 移除arr[i]，则剩余元素为（i + 1, ...,j）
2. 移除arr[j]，则剩余元素为（i, ..., j - 1）

取两者之间的最小值并+ 1

```
#include <bits/stdc++.h>

using namespace std;

vector<vector<int>> dp(100, vector<int>(100, -1));
int minRemoves(vector<int> &arr, int i, int j, int K) {
    if (i >= j || arr[j] - arr[i] <= K) return 0;
    if (dp[i][j] != -1) return dp[i][j];
    else if (arr[j] - arr[i] > K) {
        dp[i][j] = 1 + min(minRemoves(arr, i + 1, j, K), minRemoves(arr, i, j - 1, K));
    }
    return dp[i][j];
}

int main() {
    int T;
    scanf("%d", &T);
    while (T--) {
        int N, K;
        scanf("%d %d", &N, &K);
        vector<int> arr(N);
        for (int i = 0; i < N; ++i) scanf("%d", &arr[i]);
        sort(arr.begin(), arr.end());
        printf("%d\n", minRemoves(arr, 0, N - 1, K));
    }
}
```

O(nlogn)

采用双指针法

（i，j）维护满足arr[j] - arr[i] ⩽ K的关系，而maxLen保存最长的（i，j）

最终返回arr.size() - maxLen即为要移除的元素个数

```

#include <bits/stdc++.h>

using namespace std;

int minRemoves(vector<int> &arr, int K) {
    int i = 0, j = 0;
    int maxLen = 0;
    while (j < arr.size()) {
        if (arr[j] - arr[i] <= K) {
            maxLen = max(j - i + 1, maxLen);
            j++;
        } else {
            i++;
        }
    }
    return arr.size() - maxLen;
}

int main() {
    int T;
```

```
    scanf("%d", &T);
    while (T--) {
        int N, K;
        scanf("%d %d", &N, &K);
        vector<int> arr(N);
        for (int i = 0; i < N; ++i) scanf("%d", &arr[i]);
        sort(arr.begin(), arr.end());
        printf("%d\n", minRemoves(arr, K));
    }
}
```