知识精炼（三）

主讲人：邓哲也

Codeforces 985E Pencils and boxes

你有 n 只铅笔，每个铅笔的饱和度是 a[i]。现在你要把铅笔放进盒子里，盒子可以有任意个，但是每个盒子里至少要放 k 只铅笔。并且对于一个盒子里任意两只铅笔 i 和 j 必须满足他们的饱和度差异不超过 d，即 $|a[i] - a[j]| \leqslant d$

问是否存在一种可行的放法。

$1 \leqslant k \leqslant n \leqslant 500000,\ d,\ a[i] \leqslant 10^9$

input
6 3 10 7 2 7 7 4 2
output
YES

input
6 2 3 4 5 3 13 4 10
output
YES

input
3 2 5 10 16 22
output
NO

Codeforces 985E Pencils and boxes

显然我们发现把 a 数列排序后按顺序划分一定是最优的。

因此我们可以先对 a 数列从小到大排序，然后用 f[i] 表示能否把前 i 个放进盒子里。

转移：

枚举 [j + 1, i] 放在一个盒子中，j 必须满足：

$$0 \leqslant j \leqslant i - k, a[i] - a[j + 1] \leqslant d$$

如果存在一个 j 满足 f[j] = 1，那么 f[i] = 1

初始状态 f[0] = 1

Codeforces 985E Pencils and boxes

计算 f[i] 的时候需要枚举 j:

$$0 \leqslant j \leqslant i - k,\ a[i] - a[j + 1] \leqslant d$$

可以发现这两个条件会把 j 限制在一个区间 [l, r] 中，且区间的长度最大可能为 i-k。

因此朴素的枚举需要 $O(n^2)$

Codeforces 985E Pencils and boxes

考虑优化：

计算 f[i] 时，只需要去找 f[l], …, f[r] 中是否有 1.

我们可以在计算 f 的同时，维护一个 f 的前缀和 sum。

这样只要检查 sum[r] - sum[l-1] 是否大于 0 即可。

时间复杂度 O(n)

Codeforces 985E Pencils and boxes

```
int ask(int l, int r){
    if (l > r) return 0;
    if (!l) return 1;
    return (sum[r] - sum[l - 1]) > 0;
}
```

Codeforces 985E Pencils and boxes

```
int main(){
    scanf("%d%d%d", &n, &k, &d);
    for(int i = 1;i <= n;i ++) scanf("%d", &a[i]);
    sort(a + 1, a + n + 1);
    int l = 0;
    f[0] = sum[0] = 1;
    for (int i = 1;i <= n;i ++){
        while (a[i] - a[l + 1] > d) l ++;
        f[i] = ask(l, i - k);
        sum[i] = sum[i - 1] + f[i];
    }
    printf("%s\n", f[n] ? "YES" : "NO");
    return 0;
}
```

下节课再见