

```

1  /*
2  给定一个整数数组和一个整数 k，判断数组中是否存在两个不同的索引 i 和 j，使得 nums[i] =
   nums[j]，并且 i 和 j 的差的绝对值最大为 k。
3
4  示例 1:
5
6  输入: nums = [1,2,3,1], k = 3
7  输出: true
8
9  示例 2:
10
11 输入: nums = [1,0,1,1], k = 1
12 输出: true
13
14 示例 3:
15
16 输入: nums = [1,2,3,1,2,3], k = 2
17 输出: false
18
19 来源: 力扣 (LeetCode)
20 链接: https://leetcode-cn.com/problems/contains-duplicate-ii
21 著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。
22 */

```

分析:根据题意,只要满足两个相等的元素之间的索引差值 $\leq k$ 即可.

- 方法一:滑动窗口法,时间复杂度 $O(n * k)$,空间复杂度 $O(n)$,其中 n 是数组长度,但执行时超时了.
- 方法二:map数据结构保存法,时间复杂度 $O(n)$ (未包含map操作的时间复杂度),空间复杂度 $O(n)$.
- 方法三(方法二优化):实际上不需要用vector来保存每次出现的位置,只要保存最近一次出现的次数即可.
- todo:其实我不会计算map相关的时间复杂度和空间复杂度.
- 方法四:维护一个size只有k的set,来模拟滑动窗口法(参考了某个题解,但我找不到具体是哪个题解了).

方法一:C++_滑动窗口法

```

1  class Solution
2  {
3
4      private:
5          /* (left,right)*/
6          bool haveSame(vector<int>& nums,int left , int right)
7          {
8              int i = 0 ;
9              for(i = left+1;i<=right;i++)
10             {
11                 if(nums[left]==nums[i])
12                 {
13                     return true;
14                 }
15             }
16         }
17     }
18 }

```

```

16         return false;
17     }
18
19     public:
20     bool containsNearbyDuplicate(vector<int>& nums, int k)
21     {
22         int right = 0 ;
23         int i = 0 ;
24         int len = nums.size() ;
25         for(i = 0; i < len; i++)
26         {
27             right = (i+k < len)? (i+k) : (len-1);
28             if(haveSame(nums, i, right))
29             {
30                 return true;
31             }
32         }
33
34         return false;
35     }
36 }
37 };
38
39 /*
40 22 / 23 个通过测试用例
41 状态: 超出时间限制
42
43 */

```

方法二:C++_map数据结构保存法

```

1  class Solution
2  {
3
4      public:
5      bool containsNearbyDuplicate(vector<int>& nums, int k)
6      {
7          if(nums.size() < 2)
8          {
9              return false;
10         }
11         else
12         {
13             map<int, vector<int>>> mivi;
14             for(int i = 0 ; i < nums.size(); i++)
15             {
16                 if(mivi.count(nums[i]))
17                 {
18                     mivi[nums[i]].push_back(i);
19                     if(mivi[nums[i]][mivi[nums[i]].size()-1]-
20 mivi[nums[i]][mivi[nums[i]].size()-2] <=k)
21                     {
22                         return true;
23                     }
24                 }
25             }
26         }
27     }
28 }

```

```

25         {
26             vector<int> vi;
27             vi.push_back(i);
28             mivi[nums[i]] = vi;
29         }
30     }
31 }
32 return false;
33
34 }
35 };
36
37 /*
38 执行结果:
39 通过
40 显示详情
41 执行用时 :64 ms, 在所有 cpp 提交中击败了25.25% 的用户
42 内存消耗 :22.4 MB, 在所有 cpp 提交中击败了5.03%的用户
43 */

```

方法三:C++_map数据结构保存法(优化存储)

```

1  class solution
2  {
3
4      public:
5          bool containsNearbyDuplicate(vector<int>& nums, int k)
6          {
7              if(nums.size() < 2)
8              {
9                  return false;
10             }
11             else
12             {
13                 map<int,int> mii;
14                 for(int i = 0 ; i < nums.size();i++)
15                 {
16                     if(mii.count(nums[i]))
17                     {
18                         if(i - mii[nums[i]] <=k)
19                         {
20                             return true;
21                         }
22                         else
23                         {
24                             mii[nums[i]] = i;
25                         }
26                     }
27                     else
28                     {
29                         mii[nums[i]] = i;
30                     }
31                 }
32             }
33             return false;
34 }

```

```

35     }
36 };
37
38 /*
39 执行结果:
40 通过
41 显示详情
42 执行用时 :56 ms, 在所有 cpp 提交中击败了39.80% 的用户
43 内存消耗 :15.3 MB, 在所有 cpp 提交中击败了28.08%的用户
44 */

```

方法四:维护固定大小的集合来模拟滑动平均法

```

1  /*
2  参考了某个题解，但我忘记了具体是哪个题解.
3  class Solution
4  {
5      public boolean containsNearbyDuplicate(int[] nums, int k) {
6          HashSet<Integer> set = new HashSet<>();
7          for(int i = 0; i < nums.length; i++) {
8              if(set.contains(nums[i])) {
9                  return true;
10             }
11             set.add(nums[i]);
12             if(set.size() > k) {
13                 set.remove(nums[i - k]);
14             }
15         }
16         return false;
17     }
18 }
19 */
20
21 class Solution
22 {
23     public:
24         bool containsNearbyDuplicate(vector<int>& nums, int k)
25         {
26             set<int> s;
27             for(int i = 0 ; i < nums.size();i++)
28             {
29                 if(s.count(nums[i]))
30                 {
31                     return true;
32                 }
33                 s.insert(nums[i]);
34                 if(s.size()>k)
35                 {
36                     s.erase(nums[i-k]);
37                 }
38             }
39             return false;
40         }
41     };
42 };
43

```

```
44  /*
45  执行结果:
46  通过
47  显示详情
48  执行用时 :48 ms, 在所有 cpp 提交中击败了52.62% 的用户
49  内存消耗 :15.3 MB, 在所有 cpp 提交中击败了38.48%的用户
50  */
```

AlimyBreak
2019.11.05