# Problem 3013: Divide an Array Into Subarrays With Minimum Cost II

## Problem Information

**Difficulty:** Hard
**Acceptance Rate:** 31.22%
**Paid Only:** No
**Tags:** Array, Hash Table, Sliding Window, Heap (Priority Queue)

## Problem Description

You are given a **0-indexed** array of integers `nums` of length `n`, and two **positive** integers `k` and `dist`.

The **cost** of an array is the value of its **first** element. For example, the cost of `[1,2,3]` is `1` while the cost of `[3,4,1]` is `3`.

You need to divide `nums` into `k` **disjoint contiguous** subarrays, such that the difference between the starting index of the **second** subarray and the starting index of the `kth` subarray should be **less than or equal to** `dist`. In other words, if you divide `nums` into the subarrays `nums[0..(i1 - 1)], nums[i1..(i2 - 1)], ..., nums[ik-1..(n - 1)]`, then `ik-1 - i1 <= dist`.

Return _the**minimum** possible sum of the cost of these_ _subarrays_.

**Example 1:**

**Input:** nums = [1,3,2,6,4,2], k = 3, dist = 3 **Output:** 5 **Explanation:** The best possible way to divide nums into 3 subarrays is: [1,3], [2,6,4], and [2]. This choice is valid because ik-1 - i1 is 5 - 2 = 3 which is equal to dist. The total cost is nums[0] + nums[2] + nums[5] which is 1 + 2 + 2 = 5. It can be shown that there is no possible way to divide nums into 3 subarrays at a cost lower than 5.

**Example 2:**

**Input:** nums = [10,1,2,2,2,1], k = 4, dist = 3 **Output:** 15 **Explanation:** The best possible way to divide nums into 4 subarrays is: [10], [1], [2], and [2,2,1]. This choice is valid

because ik-1 - i1 is 3 - 1 = 2 which is less than dist. The total cost is nums[0] + nums[1] + nums[2] + nums[3] which is 10 + 1 + 2 + 2 = 15. The division [10], [1], [2,2,2], and [1] is not valid, because the difference between ik-1 and i1 is 5 - 1 = 4, which is greater than dist. It can be shown that there is no possible way to divide nums into 4 subarrays at a cost lower than 15.

**Example 3:**

**Input:** nums = [10,8,18,9], k = 3, dist = 1 **Output:** 36 **Explanation:** The best possible way to divide nums into 4 subarrays is: [10], [8], and [18,9]. This choice is valid because ik-1 - i1 is 2 - 1 = 1 which is equal to dist.The total cost is nums[0] + nums[1] + nums[2] which is 10 + 8 + 18 = 36. The division [10], [8,18], and [9] is not valid, because the difference between ik-1 and i1 is 3 - 1 = 2, which is greater than dist. It can be shown that there is no possible way to divide nums into 3 subarrays at a cost lower than 36.

**Constraints:**

* `3 <= n <= 105` * `1 <= nums[i] <= 109` * `3 <= k <= n` * `k - 2 <= dist <= n - 2`

## Code Snippets

**C++:**

```
class Solution {
public:
long long minimumCost(vector<int>& nums, int k, int dist) {

}
};
```

**Java:**

```
class Solution {
public long minimumCost(int[] nums, int k, int dist) {

}
}
```

**Python3:**

```
class Solution:
def minimumCost(self, nums: List[int], k: int, dist: int) -> int:
```