**1425. Constrained Subsequence Sum: -**

Hard Accepted: 37K Submissions: 72.3K Acceptance Rate: 51.2%

Given an integer array nums and an integer k, return the maximum sum of a **non-empty** subsequence of that array such that for every two **consecutive** integers in the subsequence, nums[i] and nums[j], where i < j, the condition j - i <= k is satisfied.

A *subsequence* of an array is obtained by deleting some number of elements (can be zero) from the array, leaving the remaining elements in their original order.

**Example 1:**

**Input:** nums = [10,2,-10,5,20], k = 2

**Output:** 37

**Explanation:** The subsequence is [10, 2, 5, 20].

**Example 2:**

**Input:** nums = [-1,-2,-3], k = 1

**Output:** -1

**Explanation:** The subsequence must be non-empty, so we choose the largest number.

**Example 3:**

**Input:** nums = [10,-2,-10,-5,20], k = 2

**Output:** 23

**Explanation:** The subsequence is [10, -2, -5, 20].

**Constraints:**

* 1 <= k <= nums.length <= 105
* -104 <= nums[i] <= 104

**Code: -**

class Solution {

public:

    int constrainedSubsetSum(vector<int>& nums, int k) {

        int n = nums.size();

        priority\_queue<pair<int,int>> pq;

        vector<int> dp(n);

        for(int j = 0; j < n; ++j){

            int maxi = 0;

            while(pq.size() and j - pq.top().second > k)

                pq.pop();

            if(pq.size())

                maxi = max(maxi, pq.top().first);

            dp[j] = nums[j] + maxi;

            pq.push({dp[j], j});

        }

        return \*max\_element(dp.begin(), dp.end());

    }

};