Policemen catch thieves

1. 选择下标最小的p, t，构成（p，t），如果 | p - t | ≤ k 计数，然后p++，t++
2. 如果最小的（p，y）组无法满足，则更新min（p，t）
3. 重复上述过程

```
// C++ program to find maximum number of thieves
// caught
#include <iostream>
#include <vector>
#include <cmath>

using namespace std;


int policeThief(char arr[], int n, int k)
{
    int res = 0;
    vector<int> thi;
    vector<int> pol;

    // store indices in the vector
    for (int i = 0; i < n; i++) {
        if (arr[i] == 'P')
            pol.push_back(i);
        else if (arr[i] == 'T')
            thi.push_back(i);
    }

    // track lowest current indices of
    // thief: thi[l], police: pol[r]
    int l = 0, r = 0;
    while (l < thi.size() && r < pol.size()) {

        // can be caught
        if (abs(thi[l] - pol[r]) <= k) {
            res++;
            l++;
            r++;
        }

            // increment the minimum index
        else if (thi[l] < pol[r])
            l++;
        else
            r++;
    }

    return res;
}
```

```
// Driver program
int main()
{
    int k, n;

    char arr1[] = { 'P', 'T', 'T', 'P', 'T' };
    k = 2;
    n = sizeof(arr1) / sizeof(arr1[0]);
    cout << "Maximum thieves caught: "
         << policeThief(arr1, n, k) << endl;

    char arr2[] = { 'T', 'T', 'P', 'P', 'T', 'P' };
    k = 2;
    n = sizeof(arr2) / sizeof(arr2[0]);
    cout << "Maximum thieves caught: "
         << policeThief(arr2, n, k) << endl;

    char arr3[] = { 'P', 'T', 'P', 'T', 'T', 'P' };
    k = 3;
    n = sizeof(arr3) / sizeof(arr3[0]);
    cout << "Maximum thieves caught: "
         << policeThief(arr3, n, k) << endl;

    return 0;
}
```