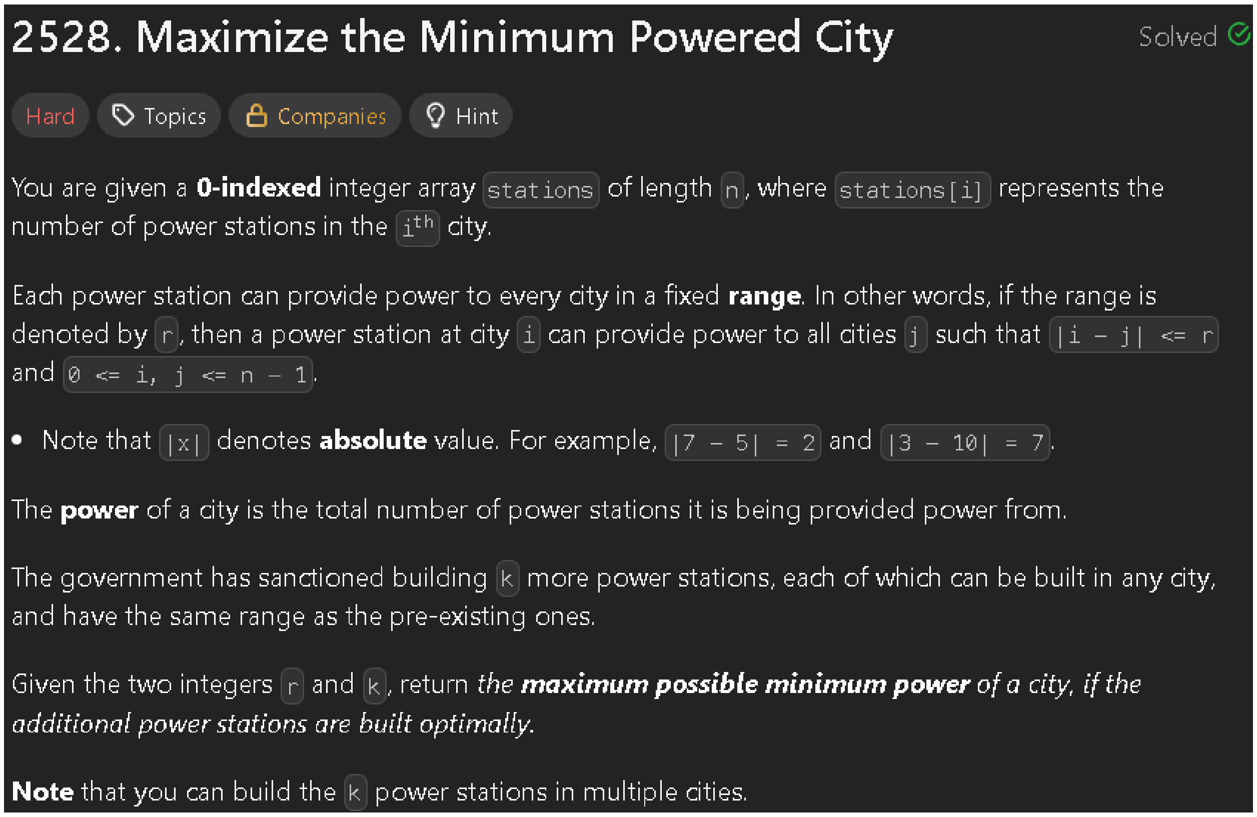


2528. Maximize the Minimum Powered City

Solved ✓

Hard · Topics · Companies · Hint

You are given a **0-indexed** integer array `stations` of length `n`, where `stations[i]` represents the number of power stations in the i^{th} city.

Each power station can provide power to every city in a fixed **range**. In other words, if the range is denoted by `r`, then a power station at city `i` can provide power to all cities `j` such that `|i - j| <= r` and `0 <= i, j <= n - 1`.

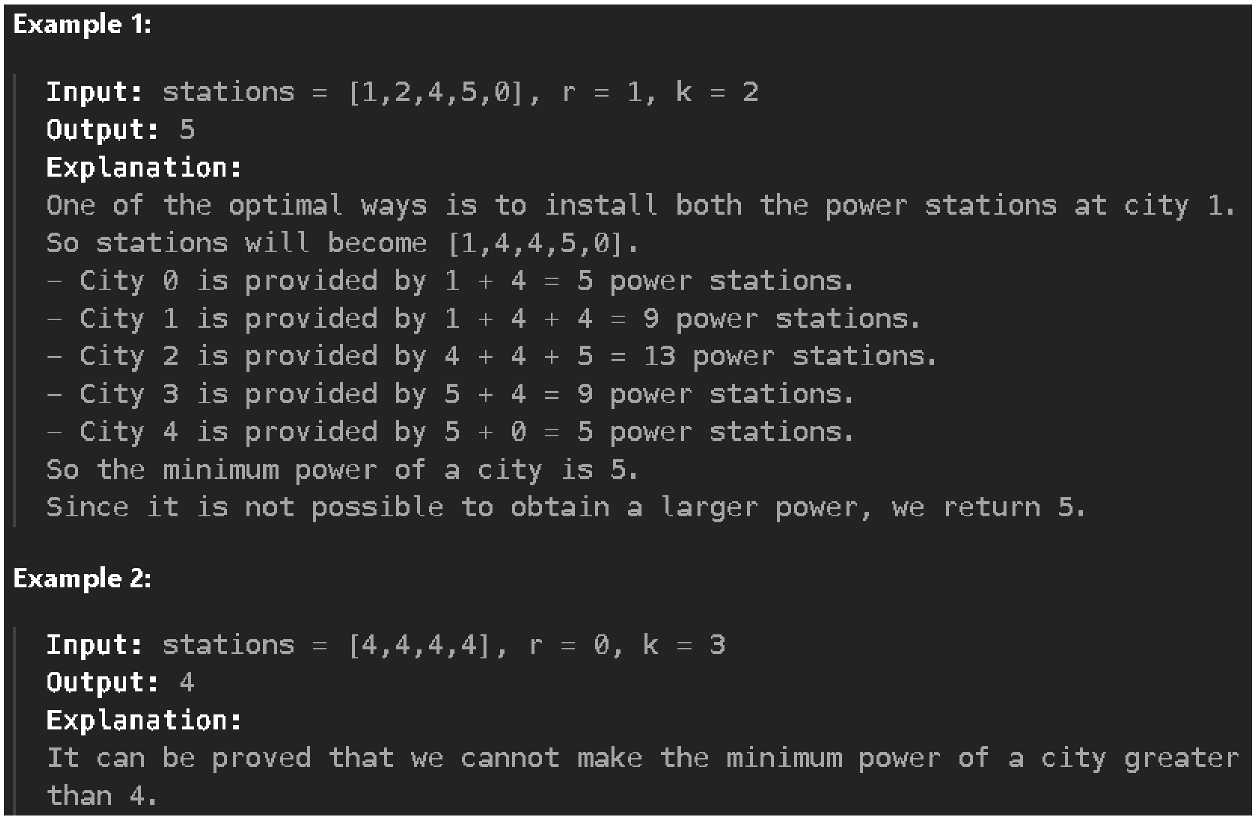
- Note that `|x|` denotes **absolute** value. For example, `|7 - 5| = 2` and `|3 - 10| = 7`.

The **power** of a city is the total number of power stations it is being provided power from.

The government has sanctioned building `k` more power stations, each of which can be built in any city, and have the same range as the pre-existing ones.

Given the two integers `r` and `k`, return *the* ***maximum possible minimum power*** *of a city, if the additional power stations are built optimally.*

Note that you can build the `k` power stations in multiple cities.

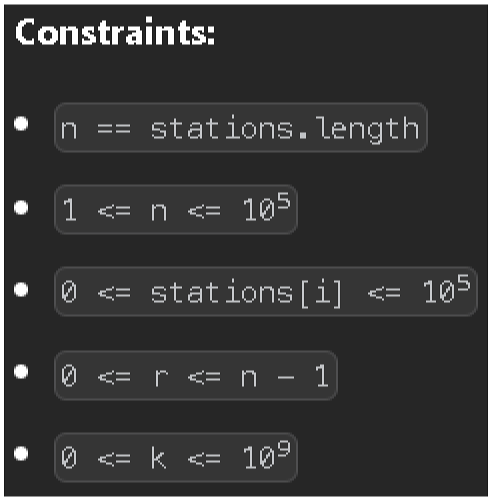


Example 1:

```
Input: stations = [1,2,4,5,0], r = 1, k = 2
Output: 5
Explanation:
One of the optimal ways is to install both the power stations at city 1.
So stations will become [1,4,4,5,0].
- City 0 is provided by 1 + 4 = 5 power stations.
- City 1 is provided by 1 + 4 + 4 = 9 power stations.
- City 2 is provided by 4 + 4 + 5 = 13 power stations.
- City 3 is provided by 5 + 4 = 9 power stations.
- City 4 is provided by 5 + 0 = 5 power stations.
So the minimum power of a city is 5.
Since it is not possible to obtain a larger power, we return 5.
```

Example 2:

```
Input: stations = [4,4,4,4], r = 0, k = 3
Output: 4
Explanation:
It can be proved that we cannot make the minimum power of a city greater
than 4.
```


Constraints:

- `n == stations.length`
- `1 <= n <= 10`^{5}
- `0 <= stations[i] <= 10`^{5}
- `0 <= r <= n - 1`
- `0 <= k <= 10`^{9}

Python:

```
class Solution:
    def maxPower(self, stations: List[int], r: int, k: int) -> int:
        n = len(stations)
        left, right = 0, k + sum(stations)
        while left <= right:
            x = (left + right) // 2
            use = 0
            # v is the stations after adding
            v = stations.copy()
            # s means the power of city i
            # at first, it record the sum of v[0,r)
            s = sum(stations[0: r])
            for i in range(n):
                # add to t if needed
                t = n - 1 if n - 1 < i + r else i + r
                # update s
                # find a city should be added
                if i + r < n: s += v[i+r]
                # find a city should be removed
                if i - r > 0: s -= v[i-r-1]
                # mising power stations
                diff = x - s if x - s > 0 else 0
                v[t] += diff
                s += diff
                use += diff

            if use <= k:
                left = x + 1
            else:
```

```
            right = x - 1
        return right
```

JavaScript:

```
/**
 * @param {number[]} stations
 * @param {number} r
 * @param {number} k
 * @return {number}
 */
var maxPower = function (stations, r, k) {
    const n = stations.length;
    const cnt = new Array(n + 1).fill(0);

    for (let i = 0; i < n; i++) {
        const left = Math.max(0, i - r);
        const right = Math.min(n, i + r + 1);
        cnt[left] += stations[i];
        cnt[right] -= stations[i];
    }

    const check = (val) => {
        const diff = [...cnt];
        let sum = 0;
        let remaining = k;

        for (let i = 0; i < n; i++) {
            sum += diff[i];
            if (sum < val) {
                const add = val - sum;
                if (remaining < add) {
                    return false;
                }
                remaining -= add;
                const end = Math.min(n, i + 2 * r + 1);
                diff[end] -= add;
                sum += add;
            }
        }
        return true;
    };

    let lo = Math.min(...stations);
    let hi = stations.reduce((a, b) => a + b, 0) + k;
    let res = 0;

    while (lo <= hi) {
        const mid = Math.floor(lo + (hi - lo) / 2);
        if (check(mid)) {
            res = mid;
            lo = mid + 1;
        } else {
            hi = mid - 1;
        }
    }
    return res;
};
```

Java:

```
class Solution {
    public long maxPower(int[] stations, int r, int k) {
        int n = stations.length;
        long left = 0, right = k;
        for (int x: stations)
            right += x;
        // v is the stations after adding
        long []v = new long[n];
        while (left <= right) {
            long x = (left + right) / 2;
            for (int i = 0; i < n; ++i)
                v[i] = stations[i];
            // s means the power of city i
            // at first, it record the sum of v[0,r)
            long s = 0, use = 0;
            for (int i = 0; i < r; ++i)
                s += v[i];
            for (int i = 0; i < n; ++i) {
                // add to t if needed
                int t = Math.min(n - 1, i + r);
                // update s
                // find a city should be added
                if (i + r < n) s += v[i + r];
                // find a city should be removed
                if (i - r > 0) s -= v[i - r - 1];
                // mising power stations
                long diff = Math.max(0, x - s);
                v[t] += diff;
                s += diff;
                use += diff;
            }
            if (use <= k) left = x + 1;
            else right = x - 1;
        }
        return right;
    }
}
```