1　扫地机器人 题解

考虑二分答案。

二分清扫时间 t，考虑时间 t 内每个机器人的路线分配。

从最左边的机器人开始，注意到它左边的区域由它清理是最优的。如果有剩余时间，它应该在保证能回来的前提下尽可能向右移动。

对于第二个机器人也是同样的，如果它左边（即与第一个机器人间隔的区域）有未被清理的方格， 应该首先清理这些方格。如果有剩余时间，也应该在保证能回来的前提下尽可能向右移动。以此类推。

一个时间 t 不可行，当且仅当满足以下几个条件的至少一个：

- 存在某一个间隔，其中有一些方格未被清理。

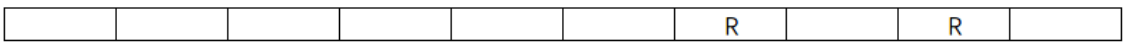
  举个例子，对于如图所示数据

  当时间 $t = 6$ 时，第一个机器人能够清理 1, 2, 4 三个格子。然而对第二个机器人，如果要保证回到起点，尽可能往左走也只能走 6, 7, 8 三个格子。这样第五个格子未被清理，$t = 6$ 不可行。

- 起点到第一个机器人之间的方格未被清理。

  举个例子，对于如图所示数据

  当时间 $t = 2$ 时，第一个机器人显然不能清理完毕前 6 个格子，$t = 2$ 不可行。

- 最后一个机器人到终点之间的方格未被清理。

举个例子，对于如图所示数据

当时间 $t = 6$ 时，第一个机器人清理了 1, 2, 4 三个格子，但是显然最后一个机器人只能清理 6, 7, 8 三个格子，最后两个格子无法清理，$t = 6$ 不可行。

一种 **check** 函数的实现如下：

```
#define lint long long
int check(lint x) {
    lint cur = 1; // cur 代表上一个机器人未清理的最前的方格坐标
    for (int i = 1; i <= k; ++i) { // 枚举 k 个机器人
        lint delta = x; // delta 准备用于存放剩余时间
        if (cur < a[i]) { // 当未被清理的方格坐标在机器人前，即该机器人前有方格未被清理
            lint t = (a[i] - cur) * 2; // 先清理这些方格，处理耗时
            if (t > x) // 判断处理前面全部未处理方格是否超时
                return 0;
            delta -= t; // 计算剩余时间
        }
        cur = a[i] + (delta / 2) + 1; // 重新计算 cur
    }
    return cur >= n + 1; // 判断是否已经清理到了 n 点
}
```

最后进行二分答案求解即可。右边界可以取 $2n$，这是一个非常稳妥的边界，因为从 0 跑到 n 再返回总耗时为 $2n$，即总工作时间一定会小于 $2n$。

```
#include <bits/stdc++.h>

using namespace std;

#define lint long long

lint n;
int k;
lint a[1000005];

int check(lint x) {
```

```
    lint cur = 1;
    for (int i = 1; i <= k; ++i) {
        lint delta = x;
        if (cur < a[i]) {
            lint t = (a[i] - cur) * 2;
            if (t > x)
                return 0;
            delta -= t;
        }
        cur = a[i] + (delta / 2) + 1;
    }
    return cur >= n + 1;
}

int main() {
    cin >> n >> k;
    for (int i = 1; i <= k; ++i)
        cin >> a[i];
    sort(a + 1, a + k + 1);
    lint l = 0, r = (n * 2), ans = 0;
    while (l <= r) {
        lint mid = (l + r) >> 1;
        if (check(mid)) {
            ans = mid;
            r = mid - 1;
        } else
            l = mid + 1;
    }
    cout << ans << endl;
    return 0;
}
```