骨牌覆盖

题意

对于一个长度为 m 的非负整数序列 $a_1, \ldots, a_n$，考察一个 m 行的棋盘，其中第 i 行有 a_i 列，若存在一种用 1×2 和 2×1 的多米诺骨牌覆盖整个棋盘的方式，则称序列 a 是好的。

给出一个长度为 n 的非负整数序列 $b_1, \ldots, b_n$，求有多少个 (l, r) 满足 $1 \leq l \leq r \leq n$ 且 $b_l, \ldots, b_r$ 可以被删空。

输入格式

第一行一个正整数 T，表示测试数据组数。

接下来对于每组数据，输入两行：

第一行一个正整数 n。

第二行 n 个非负整数 $b_1, \ldots, b_n$。

输出格式

对于每组数据，输出一行一个非负整数，表示答案。

样例

样例 1

样例 1 输入

```
9
5
5 6 6 5 3
9
3 7 1 8 4 4 0 6 9
3
3 1 0
3
3 0 1
3
2 0 2
1
0
10
4 7 6 6 7 6 1 2 5 5
6
5 5 5 4 3 3
6
6 4 4 6 4 1
```

样例 1 输出

```
7
22
3
1
6
1
12
7
15
```

样例 1 说明

对于第一组数据，$b = [5, 6, 6, 5, 3]$，合法的 (l, r) 有：$(2, 2)$，$(3, 3)$，$(2, 3)$，$(4, 5)$，$(3, 5)$，$(1, 4)$，$(2, 5)$。

样例 2

见下发文件。

数据范围

对于所有数据，满足 $1 \le T \le 100$，$1 \le n \le 5 \times 10^5$，$\sum n \le 10^6$，$0 \le b_i \le 10^9$。

- subtask 1(5%)：$n \le 10$。
- subtask 2(20%)：$n \le 100$，$\sum n \le 5 \times 10^3$。
- subtask 3(20%)：$\sum n \le 5 \times 10^3$。
- subtask 4(20%)：$\sum n \le 10^5$。
- subtask 5(35%)：无特殊限制。

题解

首先考虑判断 a 能不能被删空。

将题目转化一下，变成可以进行如下两种操作：

- 将任意 a_i 减少 2
- 将任意 $a_i = a_{i+1}$ 都减少 1

这显然和原题是等价的。

注意到这些操作显然和奇偶性有很大关系，最后我们的目标就是 $a_i = 0$，借助操作 1，我们只需要达成 $a_i \equiv 0 \pmod 2$，进一步地，我们只需要达成 $a_1 \equiv a_2 \equiv \ldots \equiv a_n \pmod 2$，先不管最后 $a_i \ge 0$ 的限制，变成最大化此时的 $\min\{a_i\}$，只要其 ≥ 0 就合法。

于是我们有这样一个初步的想法：对于 $a_i \equiv a_{i+1} \pmod 2$，用操作 1 将他们都操作成 $\min\{a_i, a_{i+1}\}$，随后就可以使用操作 1 了；进一步地，对于 $a_i = a_{i+1}$，我们可以近似地看作这这两个数消失了，这是由于当我们想用操作 1 将 a_{i-1} 和 a_{i+2} 都减少 1 时，我们可以分别操作 $a_{i-1} = a_i$ 和 $a_{i+1} = a_{i+2}$，前提是 $a_{i-1}, a_{i+2} \le a_i = a_{i+1}$。

这提示我们可能要进行一个类似括号匹配的过程，不断消去相邻两个奇偶性相同的元素，直到剩余一个元素或者删空。我们可以得到如下的一个贪心：顺序遍历所有元素，同时维护一个栈，表示当前还没有被消去的元素位置，这样就可以得到一组匹配，容易证明，如果最后栈里剩下了至少 2 个元素，那么必然是不能够删空的（如此 $|\#(a_i + i \equiv 0 \pmod 2) - \#(a_i + i \equiv 1 \pmod 2)| \ge 2$，而这显然是一个不变量，但最终全 0 时这个值只能是 0 或 1）。

我们断言：如果存在解，那么必然存在一组解，每次经过若干个操作 1 和一些操作 2，会额外让一个匹配中间的数变得全相同。具体地，我们在栈中额外维护栈中相邻两个元素之间的空档最大都变成了几，那么对于新匹配的 $a_i \equiv a_j \pmod 2$，若其中的空档都变成了 x，那么必然有 $a_i \not\equiv x \pmod 2$，此时这些数全部变成 $\min\{a_i, a_j, x-1\}$，若某次这个数字 < 0，则说明无解。

不难发现，对于 $a_i > a_{i-1}, a_{i+1}$，我们总是需要进行操作 1，那么我们进行完全部这样的操作 1 之后，就会有一些匹配的 $a_i = a_{i+1} > a_{i-1}$，取最小的如是 i 进行操作 2，不断进行这个过程，容易证明这和上文提到的贪心是完全等价的，而通过简单的调整法可以证明对这样的 i 使用操作 2 而不使用操作 1 不影响答案（可以发现使用操作 1 而不能使用操作 2 的情况会构成一个向下的阶梯状物，但底端一定是两个操作 2，从而简单调整即可）。

注意到这个匹配构成了一个树状结构，且将上述判断方式展开，最终 i 对判定合法的贡献是要求 $a_i - dep_i \geq 0$，即在匹配到 i 的时候 $a_i - siz \geq 0$，这导出了一个 $O(n)$ 的判别方法。

近一步地，由于栈的形态比较固定，只有形如“结尾为奇数/偶数，长度为 k”这一种形式，因而可以维护 $f(k, 0/1)$ 表示栈大小为 k、栈顶是奇数/偶数的栈个数（开始位置个数），每个 a_i 对栈的贡献和限制形如：

- 对于无法匹配的栈，要求 $a_i - siz \geq 0$，且会使 siz 增加 1、栈顶奇偶性翻转
- 对于可以匹配的栈，要求 $a_i - siz + 1 \geq 0$，且会使 siz 减少 1、栈顶奇偶性翻转

因而可以直接维护两个数组表示对应的答案，只需要支持 $\geq k$ 推平、整体偏移和单点修改、单点查询，推平操作由答案的连续性可以直接暴力（存在 $siz = k$ 的就必然存在 $siz = k - 1$ 的），总复杂度 $O(n)$。

参考代码：

```
#include <iostream>

const int N = 1.5e6 + 305;

int n, a[N];
int cnt[2][N], offset[2];

void solve() {
    std::cin >> n;
    for(int i = 1; i <= n; ++i) std::cin >> a[i];
    offset[0] = offset[1] = -(n + 50);
    for(int i = 0; i <= 3 * n + 150; ++i) cnt[0][i] = cnt[1][i] = 0;
    int zero = 0, flip = 0;
    long long answer = 0;
    for(int i = 1; i <= n; ++i) {
        int x = a[i];
        int same = flip ^ (x & 1);
        int diff = same ^ 1;
        for(int j = std::min(n, x) - offset[same] + 2; cnt[same][j]; ++j)
cnt[same][j] = 0;
        for(int j = std::min(n, x) - offset[diff] + 1; cnt[diff][j]; ++j)
cnt[diff][j] = 0;
        ++zero;
        cnt[diff][0 - offset[diff]] += zero;
        zero = 0;
        --offset[same];
        ++offset[diff];
        zero += cnt[same][0 - offset[same]];
```

```
            cnt[same][0 - offset[same]] = 0;
            flip ^= 1;
            answer += zero + cnt[0 ^ flip][1 - offset[0 ^ flip]];
        }
        std::cout << answer << "\n";
    }

    int main() {
        std::ios::sync_with_stdio(false);

        int t; std::cin >> t;
        while(t--) solve();

        return 0;
    }
```

参考资料

同北大附中同学和老师的一些讨论（无集训队成员）。