ACM 第二场新生赛题解

Peterlits Zo

2020 年 12 月 17 日

目录

1.1.6 示例

输入:

```
2
3
1 2 3
4
1 2 3 4
```

输出:

```
6
32
```

2 1579 题 – peter 的质数分解

2.1 题面

2.1.1 Time Limit

2s

2.1.2 Memory Limit

128M

2.1.3 题目

有一天老彼得对小彼得说:“小彼得，我这里有 t 个数组，每一个数组 A_j（$1 \le j \le t$），不妨设数组 A_j 的长度为 L_j，数组的上界为 R_j（上界不一定在数组里，但是大于等于所有的数组元素），那么我可以断言它的每一个元素 a_i（$1 \le i \le L_j$）满足 $2 \le a_i \le R_j$。”

老彼得知道小彼得最近学习了有手就行的算数基本定理（即：任何整数都存在，且只存在一种的分解为质数乘积的形式），老彼得的妻子安娜· 卡特琳娜因为一个让人羞愤的理由离开了老彼得，老彼得突然对儿子的教育燃起了新的热情，他补充到:“我知道，给定一个数 $E = p_1^{b_1} \cdot p_2^{b_2} \cdot \cdots \cdot p_n^{b_n}$，它有且只有这么一种表达形式，所以 $S(E) = b_1 + b_2 + \cdots + b_n$ 的值也是唯一的，比如说 $72 = 2^3 \times 3^2$，那么 $S(72) = 3 + 2 = 5$。小彼得，我希望你把这个数组对应的 S 值求出来。

“小彼得，我希望你好好学习，要多看看一些数学书什么的！比如《初等数论》！在这周五我会来检查你的作业情况的!”

但是在周五，小彼得收到了一台由 AMD 强力驱动的，并搭载了 Ubuntu 的计算机，根本无心写那个即愚蠢又简单的、有手就行的问题。请问你能帮帮他吗?

2.1.4 输入

输入的第一行包含了一个整数 t（$1 \le t \le 10$），代表了有 t 个测试样例，每一个测试样例具有如下的格式:

每一个测试样例的第一行包含了两个整数，分别代表了 A_j 数组的数组长度 L_j（$1 \le L_j \le 5 \times 10^2$，且 $1 \le t \times L_j \le 5 \times 10^4$），和 R_j（每一个测试样例的数字范围边界，$1 \le R_j \le 1 \times 10^9$）。

每一个测试样例的第二行包含 L_j 个元素 $a_1, a_2, \ldots, a_{L_j}$，表示 A_j，对于任意 $i \in [1, L_j]$，有: $a_i \le R_j$。

2.1.5 输出

输出 t 个结果, 每个结果对应于输入的每一个测试样例。对于每一个结果包含了 L_j 个结果 $b_1, b_2, \ldots, b_{L_j}$ 。其中 b_i 为对应的 $a_i = p_1^{c_1} \cdot p_2^{c_2} \cdot \cdots \cdot p_n^{c_n}$ 中所有素数对应的指数之和 $S(a_i) = c_1 + c_2 + \cdots + c_n$ 。

2.1.6 示例

输入:

```
1
10 100
3 12 4 34 25 78 90 45 23 21
```

输出:

```
1 3 2 2 2 3 4 3 1 2
```

2.1.7 说明

对于 3 , 有: $3 = 3 \Longrightarrow S(3) = 1$ 。
对于 12 , 有: $12 = 2^2 \cdot 3 \Longrightarrow S(12) = 3$ 。
对于 4 , 有: $4 = 2^2 \Longrightarrow S(4) = 2$ 。
对于 34 , 有: $34 = 2 \cdot 17 \Longrightarrow S(34) = 2$ 。
对于 25 , 有: $25 = 5^2 \Longrightarrow S(25) = 2$ 。
对于 78 , 有: $78 = 2 \cdot 3^2 \cdot 13 \Longrightarrow S(78) = 3$ 。
对于 90 , 有: $90 = 2 \cdot 3^2 \cdot 5 \Longrightarrow S(90) = 4$ 。
对于 45 , 有: $45 = 3^2 \cdot 5 \Longrightarrow S(45) = 3$ 。
对于 23 , 有: $23 = 23 \Longrightarrow S(23) = 1$ 。
对于 21 , 有: $21 = 3 \cdot 7 \Longrightarrow S(21) = 2$ 。

2.2 题解

不保证题解的完全正确，如果有错误、或希望补充，请访问 *https://github.com/PeterlitsZo/Answer* 修改、评论

对于每一个测试样例，我们有上界 R 和数据 $a_1, a_2, \ldots, a_L$ 。

2.2.1 筛法

使用筛法，我们可以在 $O(R + L)$ 的时间复杂度中完成操作（证明略）。但是明显时间复杂度度过大。

2.2.2 朴素分解质因数

本来这个是过不去的，但是学长说不能出太难了，所以改了一下，这个应该能过得过去。

注意，因为 米勒罗宾 *+rho* 算法只适用于 S 值较小的大合数分解（因为它的常数太大了），所以更改了数据之后的，它的解只能用朴素的分解质因数的方法

利用反证法我们可以知道, 如果一个数 N 有非平凡因子 a , 那 N 一定存在一个非平凡因子 $b \le \sqrt{N}$ 。

> 如果 $a \le \sqrt{N}$, 那么令 $b = a$; 如果 $a > \sqrt{N}$, 因为 a 为 N 的因子，有存在: $ab = N$, 其中 b 即为所求: b 为 N 的因子，且 $b \le \sqrt{N}$ 。

我们可以分类讨论：如果 N 不存在素因子 $a > \sqrt{N}$，那么我们可以在 $[2, \lfloor\sqrt{N}\rfloor]$ 的循环中解出所有的素因子；如果 N 存在素因子 $a > \sqrt{N}$，那么可以证明（见下）满足这个条件的素因子 a 有且只有一个，我们可以在循环之后判断。

如果有一个素因子 $a > \sqrt{N}$，那么 $\frac{N}{a} \leq \sqrt{N}$，因为素数整除性质，如果有另一个素因子 $b \mid N$，那么有 $b \mid a$，或者 $b \mid \frac{N}{a}$。

如果 $b \mid a$，因为素数的性质，则 $a = b > \sqrt{N}$，有 $ab > N$，矛盾；如果 $b \mid \frac{N}{a}$，那么有 $b \leq \frac{N}{a} < \sqrt{N}$。

综上，如果有一个素因子 $a > \sqrt{N}$，那么有且只有素因子 a 满足 $a > \sqrt{N}$。

可以写代码如下：

```cpp
using ll = long long;
ll solve(ll N) {
    assert (N >= 2);
    ll result = 0;
    for (ll i = 2; i*i < N; i ++) {
        ll sub = 0;
        if (N % i == 0) {
            while(N % i == 0) N /= i, sub ++;
        }
        result += sub;
    }
    if (N > 1) {
        result ++;
    }
    return result;
}

/*
int main() {
    cout << solve(ll(1e9)+7);
}
*/
```

或者 python 代码

```python
def RR(num: int):
    if num <= 1:
        return 0
    result = 0
    for i in range(2, num):
        if i**2 > num:
            break
        while num % i == 0:
            num /= i
            result += 1
    if num != 1:
        result += 1
    return result

def solve():
    _, _ = [int(i) for i in input().split()]
    nums = [int(i) for i in input().split()]
    for i, v in enumerate(nums):
        nums[i] = RR(v)
    print(*nums)

if __name__ == '__main__':
    T = int(input())
    for i in range(T):
        solve()
```

分析可知，平均的时间复杂度为 $O(\sqrt{N})$。如果每一个元素分布得比较平均，则一个测试样例的时间复杂度应为 $O(LR^{\frac{1}{2}})$（需要一点微积分知识，证明略），时间复杂度还是有点大。

2.2.3 优化的朴素分解质因数

我们注意到 $[2, \lfloor\sqrt{N}\rfloor]$ 的循环中，只有 i 为素数时才可能满足 $i \mid N$（证明略）。给定了上界 R，我们可以用欧拉筛在 $O(\sqrt{R})$ 的时间复杂度中筛出所有的小于等于 $\sqrt{N}$ 的素数（证明略），根据素数分布定理，$\sqrt{N}$ 的素数数量约为 $\frac{\sqrt{N}}{\ln\sqrt{N}}$。我们可以在 $O(\sqrt{R}+L\frac{\mathrm{Ei}(\frac{3\ln R}{2})}{R})$ 的复杂度中完成（由 https://www.wolfra-malpha.com/ 给出，我也不大会），$\mathrm{Ei}(x)$ 可以近似地看作 e^x，那么有近似 $O(\sqrt{R}+LR^{\frac{1}{2}})$。复杂度没有变化，但的确有明显的优化，常数比较小。

```cpp
using ll = long long;
template<typename T>
using vc = vector<T>;

list<ll> get_primes(ll R) {
    vc<bool> is_prime(R, true);
    list<ll> primes;
    for (ll i = 2; i <= R; i ++) {
        if (is_prime[i]) {
            primes.push_back(i);
        }
        for (auto j = primes.begin(); j != primes.end(); i ++) {
            if (i * (*j) >= R) break;
            is_primes[i * (*j)] = false;
            if (i % *j == 0) break;
        }
    }
    return primes;
}

ll solve(ll N, vc<ll>& primes) {
    assert (N >= 2);
    ll result = 0;
    for (auto i = primes.begin(); (*i)*(*i) < N; i ++) {
        ll sub = 0;
        if (N % *i == 0) {
            while(N % i == 0) N /= *i, sub ++;
        }
        result += sub;
    }
    if (N > 1) {
        result ++;
    }
    return result;
}

/*
int main() {
    list<ll>&& primes = get_primes((ll)sqrt(1e9 + 7) + 10);
    cout << solve(ll(1e9)+7, primes);
}
*/
```

2.2.4 rho 算法与 miller-rabin 素数检测

rho 算法是一种用来分解非平凡因子的随机算法，平均的时间复杂度较低。关于 rho 算法，可以查阅 https://www.cs.colorado.edu/ srirams/courses/csci2824-spr14/pollardsRho.html 或者 https://oi-wiki.org/math/polla rho/。

对于数 N，如果它不是素数的话，那我们可以使用 rho 算法将它分解为两个非平凡因子 p 与 $q=\frac{N}{p}$，有：$S(N)=S(p)\times S(q)$。

出人意料的是，虽然目前没有一个能在多项式复杂度里分解质因数的算法（不过网上说量子计算机可以在多项式的时间复杂度里分解质因数?），但是却已经可以在多项式里证明一个数是否为素数。RSA 算法就是依托于两者难度不匹配的情况设计了一个简单的公私匙系统保证通话安全。这里使用易于实现的 miller-rabin 随机算法。miller-rabin 算法可以在 https://oi-wiki.org/math/prime/#miller-rabin 进行学习。它的时间复杂度大抵在 $O(lnn)$ 左右。

```cpp
using ll = long long;

ll fmul(ll a, ll b, ll N) {
    ll result = 0;
    while (b) {
        if (b & 1) {
            result = (result + a) % N;
        }
        b >>= 1, a = a * 2 % N;
    }
    return result;
}

ll fpow(ll b, ll p, ll N) {
    ll tmp = b, result = 1;
    while (p) {
        if (p & 1) {
            result = fmul(result, tmp, N);
        }
        p >>= 1, tmp = fmul(tmp, tmp, N);
    }
    return result;
}

const ll TEST_TIMES = 20;
bool is_prime(ll N) {
    if (N <= 1) return false;

    ll s=0, d = N-1;
    while (d % 2 == 0) d/=2, s++;
    for (int _ = 0; _ < TEST_TIMES; _ ++) {
        ll a = fmul(rand(), rand(), N-2) + 2;
        ll s_ = s, d_ = d;
        ll R = fpow(a, d_, N);
        while (s_ >= 1 && R != N-1 && R != 1) {
            R = fmul(R, R, N);
            s_ --;
        }
        if (s_ == 0) return false;
    }
    return true;
}
```

不过，这种素数检测方法的常数较高，在数较小的时候不比优化后的试除法，但是题目给的数据范围较大，所以应该使用这种素数检测方法，下面是仅供参考的优化试除法：

```cpp
using ll = long long;

bool is_prime(ll N) {
    if (N <= 1) return false;
    if (N <= 3) return true;
    for (ll i = 5; i * i <= N; i += 6) {
        if (N % i == 0 || N % (i + 2) == 0) {
            return false;
        }
    }
    return true;
}
```

接下来是 rho 算法，它期望得到一个合数，并且会返回一个非平凡因子，在函数体里用到了一些上面的函数，和 ‘gcd‘ 函数：

```cpp
ll gcd(ll a, ll b) {
    return a ? gcd(b % a, a) : b;
}
```

下面是 rho 算法的定义：

```C++
ll rho(ll N) {
    while (true) {
        ll c = fmul(rand(), rand(), N-1) + 1;
```

```
        ll l = 2, cnt = 0;
        ll x = 0, y = 0;
        while (true) {
            x = (fmul(x, x, N) + c) % N;
            if (x == y) break;
            ll G = gcd(abs(x - y), N);
            if (G == N) break;
            if (G != 1) return G;
            if ((++cnt) == l) {
                y = x, cnt = 0, l *= 2;
            }
        }
    }
}
```

使用 rho 和检测素数的算法 is_prime，时间复杂度大约在 $O(N^{\frac{1}{4}})$。

2.2.5 参考代码

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;

ll fmul(ll a, ll b, ll N) {
    ll tmp = a, result = 0;
    while (b) {
        if (b & 1) {
            result = (result + a) % N;
        }
        b >>= 1, a = (a << 1) % N;
    }
    return result;
}

ll fpow(ll b, ll p, ll N) {
    ll tmp = b, result = 1;
    while (p) {
        if (p & 1) {
            result = fmul(result, tmp, N);
        }
        p >>= 1, tmp = fmul(tmp, tmp, N);
    }
    return result;
}

bool is_prime(ll n) {
    if (n <= 20) {
        for (ll i : {2, 3, 5, 7, 11, 13, 17, 19})
            if (n == i) return 1;
        return 0;
    } else for (ll i : {2, 3, 5, 7, 11, 13, 17, 19}) {
        if (n % i == 0) return 0;
    }

    if (n < 3) return n == 2;
    ll a = n - 1, b = 0;
    while (a % 2 == 0) a /= 2, ++b;
    for (ll i = 1, j; i <= 16; ++i) {
        ll x = rand() % (n - 2) + 2, v = fpow(x, a, n);
        if (v == 1 || v == n - 1)
            continue;
        for (j = 0; j < b; ++j) {
            v = fmul(v, v, n);
            if (v == n - 1) break;
        }
        if (j >= b) return 0;
    }
    return 1;
}

ll gcd(ll a, ll b) {
    return a ? gcd(b % a, a) : b;
}
```

```
55
56 ll rho(ll N) {
57     while (true) {
58         ll c = fmul(rand(), rand(), N - 1) + 1;
59         ll l = 2, cnt = 0;
60         ll x = 0, y = 0;
61         while (true) {
62             x = (fmul(x, x, N) + c) % N;
63             if (x == y)
64                 break;
65             ll G = gcd(abs(x - y), N);
66             if (G == N) break;
67             if (G != 1) return G;
68             if ((++cnt) == l)
69                 y = x, l *= 2, cnt = 0;
70         }
71     }
72 }
73
74 ll rr(ll N) {
75     if (is_prime(N)) {
76         return 1;
77     } else {
78         ll R = rho(N);
79         return rr(R) + rr(N / R);
80     }
81 }
82
83
84 #define F(X) (#X) << ": " << (X) << ", "
85
86 int main() {
87     ll t, n, _, N;
88     for(cin >> t; t--; ) {
89         cerr << F(t) << endl;
90         for (cin >> n >> _; n --; ) {
91             cin >> N;
92             cerr << F(n) << F(_) << F(N) << endl;
93             cout << rr(N) << " ";
94         }
95         cout << endl;
96     }
97 }
```

3 1580 题 – cgg 的秀发

3.1 题面

3.1.1 Time Limit

5s

3.1.2 Memory Limit

128M

3.1.3 题目

小心你的魔法把你给毁了。

《炉石传说》是 pzgg 最喜欢玩的游戏 , 并且 pzgg 已经达到了炉火纯青的地步。前两天，炉石的策划师给 pzgg 打电话，告知他要出一张新的魔法卡牌，想听听 pzgg 的意见。卡牌的作用如下:

假设敌方战场上有 n 只随从，第 i 只随从（ $1 \leq i \leq n$ ）具有生命值 a_i 。

当 $n \geq 3$ 时，己方英雄可以打出该卡牌，然后选择一个敌方随从 i（$2 \leq i \leq n-1$），可以直接将该随从秒杀 (之后移出战场)，并为己方英雄回复 $a_{i-1} \times a_i \times a_{i+1}$ 的血量。

为了测试这张卡牌会不会违反游戏平衡，策划师给了 pzgg 若干个战场，并且给了 pzgg 无数张该魔法卡牌，策划师想知道在每个战场上，己方英雄最多可回复多少滴血量。

pzgg 是 acm 高手，觉得这太简单了 , 所以现在他想来考考聪明的你。

3.1.4 输入

输入包括 $2 \times T + 1$ 行；第一行包括一个整数 T（$1 \leq T \leq 10$），即以下有 T 个战场需要你去求出结果；接下去 T 组数据，每组数据有两行，第一行是一个整数 n （J1 $\leq n \leq 200$）表示场上有 n 个随从，第二行有 n 个数，第 i 个数 a_i（$1 \leq a_i \leq 1 \times 10^5$）表示第 i 个随从的生命值。

3.1.5 输出

对于每组数据输出一行，表示己方英雄可以最多回复的血量。

3.1.6 示例

输入:

```
2
3
1 2 3
4
1 2 3 4
```

输出:

```
6
32
```

4 1581 题 – 韭菜

4.1 题面

4.1.1 Time Limit

5s

4.1.2 Memory Limit

128M

4.1.3 题目

小心你的魔法把你给毁了。

《炉石传说》是 pzgg 最喜欢玩的游戏 , 并且 pzgg 已经达到了炉火纯青的地步。前两天，炉石的策划师给 pzgg 打电话，告知他要出一张新的魔法卡牌，想听听 pzgg 的意见。卡牌的作用如下:

假设敌方战场上有 n 只随从，第 i 只随从（$1 \leq i \leq n$）具有生命值 a_i。

当 $n \geq 3$ 时，己方英雄可以打出该卡牌，然后选择一个敌方随从 i（$2 \leq i \leq n-1$），可以直接将该随从秒杀 (之后移出战场)，并为己方英雄回复 $a_{i-1} \times a_i \times a_{i+1}$ 的血量。

为了测试这张卡牌会不会违反游戏平衡，策划师给了 pzgg 若干个战场，并且给了 pzgg 无数张该魔法卡牌，策划师想知道在每个战场上，己方英雄最多可回复多少滴血量。

pzgg 是 acm 高手，觉得这太简单了 , 所以现在他想来考考聪明的你。

4.1.4 输入

输入包括 $2 \times T + 1$ 行；第一行包括一个整数 T（ $1 \leq T \leq 10$ ），即以下有 T 个战场需要你去求出结果；接下去 T 组数据，每组数据有两行，第一行是一个整数 n （ J1 $\leq n \leq 200$ ）表示场上有 n 个随从，第二行有 n 个数，第 i 个数 a_i（ $1 \leq a_i \leq 1 \times 10^5$ ）表示第 i 个随从的生命值。

4.1.5 输出

对于每组数据输出一行，表示己方英雄可以最多回复的血量。

4.1.6 示例

输入:

```
2
3
1 2 3
4
1 2 3 4
```

输出:

```
6
32
```

5 1582 题 – 卷王的终结

5.1 题面

5.1.1 Time Limit

5s

5.1.2 Memory Limit

128M

5.1.3 题目

小心你的魔法把你给毁了。

《炉石传说》是 pzgg 最喜欢玩的游戏 , 并且 pzgg 已经达到了炉火纯青的地步。前两天，炉石的策划师给 pzgg 打电话，告知他要出一张新的魔法卡牌，想听听 pzgg 的意见。卡牌的作用如下:

假设敌方战场上有 n 只随从，第 i 只随从（ $1 \leq i \leq n$ ）具有生命值 a_i 。

当 $n \geq 3$ 时，己方英雄可以打出该卡牌，然后选择一个敌方随从 i（ $2 \leq i \leq n-1$ ），可以直接将该随从秒杀 (之后移出战场) ，并为己方英雄回复 $a_{i-1} \times a_i \times a_{i+1}$ 的血量。

为了测试这张卡牌会不会违反游戏平衡，策划师给了 pzgg 若干个战场，并且给了 pzgg 无数张该魔法卡牌，策划师想知道在每个战场上，己方英雄最多可回复多少滴血量。

pzgg 是 acm 高手，觉得这太简单了 , 所以现在他想来考考聪明的你。

5.1.4 输入

输入包括 $2 \times T + 1$ 行；第一行包括一个整数 T（$1 \leq T \leq 10$），即以下有 T 个战场需要你去求出结果；接下去 T 组数据，每组数据有两行，第一行是一个整数 n （J1 $\leq n \leq 200$）表示场上有 n 个随从，第二行有 n 个数，第 i 个数 a_i（$1 \leq a_i \leq 1 \times 10^5$）表示第 i 个随从的生命值。

5.1.5 输出

对于每组数据输出一行，表示己方英雄可以最多回复的血量。

5.1.6 示例

输入:

```
2
3
1 2 3
4
1 2 3 4
```

输出:

```
6
32
```

6 1583 题 – 食堂打饭

6.1 题面

6.1.1 Time Limit

5s

6.1.2 Memory Limit

128M

6.1.3 题目

小心你的魔法把你给毁了。

《炉石传说》是 pzgg 最喜欢玩的游戏 , 并且 pzgg 已经达到了炉火纯青的地步。前两天，炉石的策划师给 pzgg 打电话，告知他要出一张新的魔法卡牌，想听听 pzgg 的意见。卡牌的作用如下:

假设敌方战场上有 n 只随从，第 i 只随从（$1 \leq i \leq n$）具有生命值 a_i 。

当 $n \geq 3$ 时，己方英雄可以打出该卡牌，然后选择一个敌方随从 i（$2 \leq i \leq n-1$），可以直接将该随从秒杀 (之后移出战场) ，并为己方英雄回复 $a_{i-1} \times a_i \times a_{i+1}$ 的血量。

为了测试这张卡牌会不会违反游戏平衡，策划师给了 pzgg 若干个战场，并且给了 pzgg 无数张该魔法卡牌，策划师想知道在每个战场上，己方英雄最多可回复多少滴血量。

pzgg 是 acm 高手，觉得这太简单了 , 所以现在他想来考考聪明的你。

6.1.4 输入

输入包括 $2 \times T + 1$ 行；第一行包括一个整数 T（$1 \leq T \leq 10$），即以下有 T 个战场需要你去求出结果；接下去 T 组数据，每组数据有两行，第一行是一个整数 n （J1 $\leq n \leq 200$）表示场上有 n 个随从，第二行有 n 个数，第 i 个数 a_i（$1 \leq a_i \leq 1 \times 10^5$）表示第 i 个随从的生命值。

6.1.5 输出

对于每组数据输出一行，表示己方英雄可以最多回复的血量。

6.1.6 示例

输入:

```
2
3
1 2 3
4
1 2 3 4
```

输出:

```
6
32
```

7 1584 题 – 完全二叉树

7.1 题面

7.1.1 Time Limit

5s

7.1.2 Memory Limit

128M

7.1.3 题目

小心你的魔法把你给毁了。

《炉石传说》是 pzgg 最喜欢玩的游戏 , 并且 pzgg 已经达到了炉火纯青的地步。前两天，炉石的策划师给 pzgg 打电话，告知他要出一张新的魔法卡牌，想听听 pzgg 的意见。卡牌的作用如下:

假设敌方战场上有 n 只随从，第 i 只随从（$1 \leq i \leq n$）具有生命值 a_i 。

当 $n \geq 3$ 时，己方英雄可以打出该卡牌，然后选择一个敌方随从 i（$2 \leq i \leq n-1$），可以直接将该随从秒杀 (之后移出战场) ，并为己方英雄回复 $a_{i-1} \times a_i \times a_{i+1}$ 的血量。

为了测试这张卡牌会不会违反游戏平衡，策划师给了 pzgg 若干个战场，并且给了 pzgg 无数张该魔法卡牌，策划师想知道在每个战场上，己方英雄最多可回复多少滴血量。

pzgg 是 acm 高手，觉得这太简单了 , 所以现在他想来考考聪明的你。

7.1.4 输入

输入包括 $2 \times T + 1$ 行；第一行包括一个整数 T（$1 \leq T \leq 10$），即以下有 T 个战场需要你去求出结果；接下去 T 组数据，每组数据有两行，第一行是一个整数 n （J1 $\leq n \leq 200$）表示场上有 n 个随从，第二行有 n 个数，第 i 个数 a_i（$1 \leq a_i \leq 1 \times 10^5$）表示第 i 个随从的生命值。

7.1.5 输出

对于每组数据输出一行，表示己方英雄可以最多回复的血量。

7.1.6 示例

输入:

```
2
3
1 2 3
4
1 2 3 4
```

输出:

```
6
32
```

8 1585 题 – 我真的不想拿外卖

8.1 题面

8.1.1 Time Limit

5s

8.1.2 Memory Limit

128M

8.1.3 题目

小心你的魔法把你给毁了。

《炉石传说》是 pzgg 最喜欢玩的游戏 , 并且 pzgg 已经达到了炉火纯青的地步。前两天，炉石的策划师给 pzgg 打电话，告知他要出一张新的魔法卡牌，想听听 pzgg 的意见。卡牌的作用如下:

假设敌方战场上有 n 只随从，第 i 只随从（$1 \leq i \leq n$）具有生命值 a_i 。

当 $n \geq 3$ 时，己方英雄可以打出该卡牌，然后选择一个敌方随从 i（$2 \leq i \leq n-1$），可以直接将该随从秒杀 (之后移出战场) ，并为己方英雄回复 $a_{i-1} \times a_i \times a_{i+1}$ 的血量。

为了测试这张卡牌会不会违反游戏平衡，策划师给了 pzgg 若干个战场，并且给了 pzgg 无数张该魔法卡牌，策划师想知道在每个战场上，己方英雄最多可回复多少滴血量。

pzgg 是 acm 高手，觉得这太简单了 , 所以现在他想来考考聪明的你。

8.1.4 输入

输入包括 $2 \times T + 1$ 行；第一行包括一个整数 T（$1 \leq T \leq 10$），即以下有 T 个战场需要你去求出结果；接下去 T 组数据，每组数据有两行，第一行是一个整数 n （J1 $\leq n \leq 200$）表示场上有 n 个随从，第二行有 n 个数，第 i 个数 a_i（$1 \leq a_i \leq 1 \times 10^5$）表示第 i 个随从的生命值。

8.1.5 输出

对于每组数据输出一行，表示己方英雄可以最多回复的血量。

8.1.6 示例

输入:

```
2
3
1 2 3
4
1 2 3 4
```

输出:

```
6
32
```

9 1586 题 – 噪音控制

9.1 题面

9.1.1 Time Limit

5s

9.1.2 Memory Limit

128M

9.1.3 题目

小心你的魔法把你给毁了。

《炉石传说》是 pzgg 最喜欢玩的游戏 , 并且 pzgg 已经达到了炉火纯青的地步。前两天，炉石的策划师给 pzgg 打电话，告知他要出一张新的魔法卡牌，想听听 pzgg 的意见。卡牌的作用如下:

假设敌方战场上有 n 只随从，第 i 只随从（$1 \leq i \leq n$）具有生命值 a_i 。

当 $n \geq 3$ 时，己方英雄可以打出该卡牌，然后选择一个敌方随从 i（$2 \leq i \leq n-1$），可以直接将该随从秒杀 (之后移出战场) ，并为己方英雄回复 $a_{i-1} \times a_i \times a_{i+1}$ 的血量。

为了测试这张卡牌会不会违反游戏平衡，策划师给了 pzgg 若干个战场，并且给了 pzgg 无数张该魔法卡牌，策划师想知道在每个战场上，己方英雄最多可回复多少滴血量。

pzgg 是 acm 高手，觉得这太简单了 , 所以现在他想来考考聪明的你。

9.1.4 输入

输入包括 $2 \times T + 1$ 行；第一行包括一个整数 T（$1 \leq T \leq 10$），即以下有 T 个战场需要你去求出结果；接下去 T 组数据，每组数据有两行，第一行是一个整数 n （J1 $\leq n \leq 200$）表示场上有 n 个随从，第二行有 n 个数，第 i 个数 a_i（$1 \leq a_i \leq 1 \times 10^5$）表示第 i 个随从的生命值。

9.1.5 输出

对于每组数据输出一行，表示己方英雄可以最多回复的血量。

9.1.6 示例

输入:

```
2
3
1 2 3
4
1 2 3 4
```

输出:

```
6
32
```

10 1587 题 – 走出自己的舒适区

10.1 题面

10.1.1 Time Limit

5s

10.1.2 Memory Limit

128M

10.1.3 题目

小心你的魔法把你给毁了。

《炉石传说》是 pzgg 最喜欢玩的游戏 , 并且 pzgg 已经达到了炉火纯青的地步。前两天，炉石的策划师给 pzgg 打电话，告知他要出一张新的魔法卡牌，想听听 pzgg 的意见。卡牌的作用如下:

假设敌方战场上有 n 只随从，第 i 只随从（$1 \leq i \leq n$）具有生命值 a_i 。

当 $n \geq 3$ 时，己方英雄可以打出该卡牌，然后选择一个敌方随从 i（$2 \leq i \leq n-1$），可以直接将该随从秒杀 (之后移出战场) ，并为己方英雄回复 $a_{i-1} \times a_i \times a_{i+1}$ 的血量。

为了测试这张卡牌会不会违反游戏平衡，策划师给了 pzgg 若干个战场，并且给了 pzgg 无数张该魔法卡牌，策划师想知道在每个战场上，己方英雄最多可回复多少滴血量。

pzgg 是 acm 高手，觉得这太简单了 , 所以现在他想来考考聪明的你。

10.1.4 输入

输入包括 $2 \times T + 1$ 行；第一行包括一个整数 T（$1 \leq T \leq 10$），即以下有 T 个战场需要你去求出结果；接下去 T 组数据，每组数据有两行，第一行是一个整数 n （J1 $\leq n \leq 200$）表示场上有 n 个随从，第二行有 n 个数，第 i 个数 a_i（$1 \leq a_i \leq 1 \times 10^5$）表示第 i 个随从的生命值。

10.1.5 输出

对于每组数据输出一行，表示己方英雄可以最多回复的血量。

10.1.6 示例

输入:

```
2
3
1 2 3
4
1 2 3 4
```

输出:

```
6
32
```

11 1588 题 – cgg 的秀发（Easy Version）

11.1 题面

11.1.1 Time Limit

5s

11.1.2 Memory Limit

128M

11.1.3 题目

小心你的魔法把你给毁了。

《炉石传说》是 pzgg 最喜欢玩的游戏 , 并且 pzgg 已经达到了炉火纯青的地步。前两天，炉石的策划师给 pzgg 打电话，告知他要出一张新的魔法卡牌，想听听 pzgg 的意见。卡牌的作用如下:

假设敌方战场上有 n 只随从，第 i 只随从（$1 \leq i \leq n$）具有生命值 a_i 。

当 $n \geq 3$ 时，己方英雄可以打出该卡牌，然后选择一个敌方随从 i（$2 \leq i \leq n-1$），可以直接将该随从秒杀 (之后移出战场) ，并为己方英雄回复 $a_{i-1} \times a_i \times a_{i+1}$ 的血量。

为了测试这张卡牌会不会违反游戏平衡，策划师给了 pzgg 若干个战场，并且给了 pzgg 无数张该魔法卡牌，策划师想知道在每个战场上，己方英雄最多可回复多少滴血量。

pzgg 是 acm 高手，觉得这太简单了 , 所以现在他想来考考聪明的你。

11.1.4 输入

输入包括 $2 \times T + 1$ 行；第一行包括一个整数 T（$1 \leq T \leq 10$），即以下有 T 个战场需要你去求出结果；接下去 T 组数据，每组数据有两行，第一行是一个整数 n （J1 $\leq n \leq 200$）表示场上有 n 个随从，第二行有 n 个数，第 i 个数 a_i（$1 \leq a_i \leq 1 \times 10^5$）表示第 i 个随从的生命值。

11.1.5 输出

对于每组数据输出一行，表示己方英雄可以最多回复的血量。

11.1.6 示例

输入:

```
2
3
1 2 3
4
1 2 3 4
```

输出:

```
6
32
```

12 1589 题 – 初识网络流

12.1 题面

12.1.1 Time Limit

5s

12.1.2 Memory Limit

128M

12.1.3 题目

小心你的魔法把你给毁了。

《炉石传说》是 pzgg 最喜欢玩的游戏 , 并且 pzgg 已经达到了炉火纯青的地步。前两天，炉石的策划师给 pzgg 打电话，告知他要出一张新的魔法卡牌，想听听 pzgg 的意见。卡牌的作用如下:

假设敌方战场上有 n 只随从，第 i 只随从（$1 \leq i \leq n$）具有生命值 a_i 。

当 $n \geq 3$ 时，己方英雄可以打出该卡牌，然后选择一个敌方随从 i（$2 \leq i \leq n-1$），可以直接将该随从秒杀 (之后移出战场) ，并为己方英雄回复 $a_{i-1} \times a_i \times a_{i+1}$ 的血量。

为了测试这张卡牌会不会违反游戏平衡，策划师给了 pzgg 若干个战场，并且给了 pzgg 无数张该魔法卡牌，策划师想知道在每个战场上，己方英雄最多可回复多少滴血量。

pzgg 是 acm 高手，觉得这太简单了 , 所以现在他想来考考聪明的你。

12.1.4 输入

输入包括 $2 \times T + 1$ 行；第一行包括一个整数 T（$1 \leq T \leq 10$），即以下有 T 个战场需要你去求出结果；接下去 T 组数据，每组数据有两行，第一行是一个整数 n （J1 $\leq n \leq 200$）表示场上有 n 个随从，第二行有 n 个数，第 i 个数 a_i（$1 \leq a_i \leq 1 \times 10^5$）表示第 i 个随从的生命值。

12.1.5 输出

对于每组数据输出一行，表示己方英雄可以最多回复的血量。

12.1.6 示例

输入：

```
2
3
1 2 3
4
1 2 3 4
```

输出：

```
6
32
```