A.杠上开花！（1）

考虑第1个敌人：

- 如果血量为1，一定是攻击第2个敌人，并使第3个敌人的血量减少1
- 如果血量为2，要么是直接攻击第1个敌人，使第2个敌人血量减少1，要么是攻击第2个敌人，然后攻击新的第2个敌人。如果第3个敌人血量为1，这样做会消灭第2~4个敌人，使第5个敌人血量减少1；如果第3个敌人血量为2，这样做会消灭第2、3个敌人，使第4个敌人血量减少1。

记$f_{i,0/1}$表示考虑第i个以后的敌人，第i个敌人的血量是否减少1的最小操作数。直接dp即可。

时间复杂度$O(n)$

A.杠上开花！（2）

注意到如果$n \leq 3$，一次攻击至少能造成3伤害。

那么我们只需要考虑让造成4伤害的攻击尽可能多，而造成4伤害只能攻击血量为2的敌人。

我们考虑选出一个血量为2的敌人的子集，要求它们都作为攻击的目标，然后最大化这个子集的大小M。

我我们考虑这个子集合法的必要条件：

- 不能有处于最左侧或最右侧的
- 不能有相邻的
- $4M \leq \sum_{i=1}^{n} a_i$

可以证明满足上述条件也是充分的。具体证明如下：

我们在序列中找到血量和最大的一段不包含被选中为目标的敌人的敌人，这个段的血量和一定大于1。

那么我们攻击与这个段相邻的某个目标敌人后，一定依然满足上述性质。同时目标的个数减少了一个。

以此类推，一定能攻击所有的目标敌人。

考虑怎么求满足上述条件的最大子集。

首先挑出所有血量为2的不在最左侧和最右侧的敌人，然后对于一个连续的长度为l的段，我们至少要删去$\lfloor \frac{l}{2} \rfloor$个。

删完之后即满足前两个条件。然后此时如果依然不满足第三个条件，子集大小就是$\lfloor \frac{\sum_{i=1}^{n} a_i}{4} \rfloor$

然后最终的答案是$M + \lceil \frac{\sum_{i=1}^{n} a_i - 4M}{3} \rceil$

时间复杂度$O(n)$

B.公交星槎

考虑将线路上的每双向条边拆成两条单向边，再对每条单向边新开一个节点，记第i条线路上从u到v的单向边的节点为(i,u,v)。

那么连边就是：

- (i,u,v)向v连代价为t的边
- (i,u,v)向(i,v,w)连代价为1的边，其中w是线路上的下一个节点。

- u向(i,u,v)连一条边，这条边的权值不定，如果1到u的最短路长度为dis_u，那么会使$dis_{(i,u,v)}$更新为dis_u之后第一个 u节点有开向v的星槎 的时刻+1。

跑dijkstra即可。

时间复杂度$O(n+\sum m_i \log)$

C.记忆碎片

令字符串总长为 m，字符集大小为 σ。

part 1

纯纯的暴力，每次加入之后 m^2 判断先前的每个记忆串是否能被凑出，复杂度 $O(nm^2)$。

part 2

不那么纯的暴力，写个哈希尊重一下，复杂度 $O(n^3+m)$。

part 3

由于题中限制所有事件串不互为前缀，不难发现对于一个记忆串，如果它能被分割成若干个事件串，那么分割方案一定唯一。所以我们甚至不用 AC自动机，只需写一个 trie 来维护所有的事件串，然后将每个新加入的放入 trie 树上进行匹配，每次走到底重新回到根节点，若某一次停在了非末尾节点则这个串无法被表示，否则则可以被表示。复杂度 $O(m)$。

part 4

对所有询问串建立 trie 树，令初始时的出发节点集合为 trie 树的根，每次读入一个事件串时，拿出集合中的每一个出发节点，以其作为起点用事件串去行走，若能走到底则将走到底后所在的节点加入起点集合。由于每次都枚举起点集合中的每一个点会时复杂度过大，我们需要同时维护每个起点还没走过的出边，使用一些技巧让总复杂度下降。复杂度 $O(m\sigma)$

part 5

考虑随着事件串的加入动态维护当前的 trie。当一个操作加入一个新的记忆串时，我们将它放在 trie 中能跑就跑直到跑不了为止，此时我们将这个串还未匹配的剩下部分挂在当前节点，类似于懒标记：具体来说，对于 trie 上的每一个点，我们新建 σ 个链表 `list[ch]`，表示当前节点存在若干无法继续匹配的事件串，失配的字符为 ch；当加入一个新的事件串时，我们将其插入 trie 中，若我们在某个节点 u 新建了一条出边 ch，则将`list[ch]` 中的所有事件串接着进行匹配。其算法的核心思想就是由于每个串的分割方式唯一，我们可以谈薪匹配能走就走，然后失配之后没有必要重新开始，有边能走之后再继续走。不难发现插入 trie 的总复杂度为 $O(m)$，而匹配的总复杂度也是 $O(m)$，至此本题结束。

> 如果代码在实现上不精细容易将代码复杂度写成 $O(m\sigma)$，但不影响题目的通过。

D.金人旧巷

考虑随便找一个根，然后把1操作变换为子树操作：记$M(u,w,p)$表示给u子树中距离u为d的节点增加$\lfloor \frac{w}{p^d} \rfloor$。

当$p=1$，是全局加操作，可以打一个全局标记来解决。

当$p\neq 1$:

记u的i级祖先为$fa(u,i)$，$fa(u,0)=u$，那么一个1操作相当于：

$M(u,w,p)$

$M(fa(u,1),\frac{w}{p},p),M(fa(u,0),-\frac{w}{p^2},p)$

$M(fa(u,2),\frac{w}{p^2},p),M(fa(u,1),-\frac{w}{p^3},p)$

以此类推。由于$p\neq 1$，所以只会有$O(\log w)$个有效的操作。

然后考虑回答询问。每次询问中有一个是求子树和，另一问是求全局总和-子树和。考虑怎么求u的子树和。

第一种情况是修改在u的子树内，这种情况可以计算每次修改在所有节点上增加的权值总和，然后做一次子树丘壑。

第二种情况是修改在u的祖先上。那么我们记$vl_{v,d}$表示所有v节点上的修改对v子树内的**某个**距离为d的节点产生的贡献，用$num_{u,d}$表示u子树内距离为d的节点个数。然后枚举深度j和u,v之间的距离i，v上的修改产生的贡献就是$vl_{v,i+j}\times num_{u,j}$。同样，$i,j$都是$O(\log w)$级别。

时间复杂度$O(n\log^2 w)$