# 过河

## 对于前 50% 的数据

我们考虑贪心，每次从左往右送的时候，我们取前 $L$ 大的(如果需要往返的话，大的体力肯定够)，然后运送前 $R-L$ 小的。从右往左运时，我们令前 $L$ 大的人回去（送回去的人越多，总的体力减少的越多，不优）。用堆来维护两边的人，模拟划船过程即可。

## 对于另外 20% 的数据

显然是有 $n-2$ 个人去一次回一次去一次，$2$ 个人去一次。我们只需判断体力小于 $3$ 的人数是否小于 $3$ 即可。

## 对于前 100% 的数据

经过简单计算可以发现，需要从河的右侧往回运送的趟数最小值：$S=\lceil\frac{n-R}{R-L}\rceil$

令 $a_i=\lfloor\frac{h_i-1}{2}\rfloor$ 为第 $i$ 个人能多来回的趟数

一个必要条件是 $\sum_{i=1}^{n}min(a_i,S)\geq SL$

下述贪心保证条件的充分性：每次将体力值最高的若干个人从一侧运输到另一侧。

贪心成立性的原因在于每次从右侧往回运的过程相当于每次将 $a_1,a_2,\dots a_n$ 中最大的 $L$ 个元素减一，更新后的数组 $a_i'$ 满足 $\sum_{i=1}^{n}min(a_i',S-1)\geq(S-1)L$，由数学归纳法可知正确性。

# 距离

## 树的情况

仿照求直径，我们可以求出某个点子树内两条最长的路径，相加即可。枚举每个点作为根，可以算出根的答案。然后换根即可求出所有的答案（换根则需要记录最大值和次大值用于转移）。

## 基环树的情况

暴力的话，我们可以考虑每次删掉环上的一个边，然后做换根dp，所有的情况取max即可。

我们不妨先把环断开，对于每个子树先用树的方法求一遍答案，并求出每个点往子树走能走的最远距离 $maxd$ 。那么对于环上点 $i$ 的子树内的点的答案，我们可以找到另一个环上的点 $j$ ，用 $max\{|i-j|+maxd_j\}+maxd_i$ ，这里 $|i-j|$ 表示在环上的最大距离，可以用拆环倍长的方法做。对于环上的点 $k$，我们也可以找两个环上的点 $i,j$ 求答案。拆开环我们得到一个序列，用 $i$ 表示序列的第 $i$ 个点，分两种情况。

- $maxd_i+max_j+j-i$ 对区间 $[i,j]$ 有贡献。
- $maxd_i+max_j+n-j+i$ 对区间 $[1,i]$ 和 $[j,n]$ 有贡献，这里的 $n$ 表示环长。

都可以通过维护前后缀最大值解决，最后多种结果取max就是答案。

时间复杂度为 $O(n\log n)$。

# 维护数据库

## 对于第二档数据

用trie树处理即可。

## 对于第三、四档数据

多了一个异或操作，对于第三档，我们可以维护一个全局的异或和，在查询的时候同时异或一下这个即可。

对于第四档，又多了一个插入操作，我们希望后面查询的时候查询这个值，但是直接插入会被全局变量给打乱。所以我们希望插入的值异或上全局变量是真实值，那么我们直接插入真实值异或上全局变量的值即可。

## 对于第五档数据

与和或操作类似，只介绍与操作。

与操作相当于，把trie树中某一位全变成0，相当于合并两颗子树。我们不妨建立一个时间轴，记录下每个时间的操作（全变0/1，查询）。然后我们给树上每个节点记录一个修改时间，当查询道某个结点的时候我们判断一下这个节点的修改时间是否在这一位全局的上次修改之前，如果是的话我们就暴力合并一下两棵子树，之后更新一下时间。这样做复杂度是对的，接下来我们证明这个复杂度。

我们每合并一次节点，会使字典树的节点个数少一。而只有插入操作会产生节点，也就是最多产生 $31(n+m)$ 个节点，因此我们的合并操作次数少于 $31(n+m)$。

## 对于第六档数据

与第五档相比，多了个异或操作，我们可以用当前时间，某一位的修改时间，某个节点的修改时间这三个数来维护我们的更新。如果遍历到一个点时，如果节点修改时间早于这一位的修改时间，我们暴力的合并一下，然后更新节点修改时间为这一位的修改时间。之后，如果节点修改时间早于当前时间，我们可以用前缀和的方法算出这两个时间点之间异或了多少个1，奇数个的话，我们交换一下左右儿子即可，之后再次更新节点修改时间为当前时间。复杂度仍为 $O((n+m)\log a)$。

# 快速排序

考虑算每个位置对答案的贡献，其实就是它递归下去的次数。

我们重新定义一下分割点的概念，第 $i$ 个元素和第 $i+1$ 个元素之间的那条线叫做分割点 $i$。

先给出结论，一个位置停止递归当且仅当他前面与后面的分割点都已经出现，即对于位置 $i$，分割点 $i$ 和分割点 $i-1$ 均已出现,不妨设第 $i$ 个分割点出现的时间为 $t_i$ 那么每一个位置 $i$ 对于答案的贡献次数就是 $max\{t_{i-1},t_i\}$。

那么现在我们就要考虑这个 $t_i$ 要如何计算，对于一个分割点 $i$，它出现的时间就是所有 $1\sim i$ 的数都在 $[1,i]$ 之间的第一个时刻。

观察每次冒泡的过程，设所有小于等于 $i$ 的数的最大的位置为 $p_i$，那么在分割点 $i$ 出现之前每次冒泡后 $p_i$ 必然会减少 $1$，所以 $t_i=p_i-i$。

每个位置加起来，对于一个确定的排列的答案就是（$p$ 的意义和上面一样）：

$$\sum_{i=1}^{n}\max\{p_{i-1}-(i-1),p_i-i\}$$

然后分别考虑 $t_{i-1}\le t_i, t_{i-1}>t_i$，两种情况的式子分别为：

$$\sum_{i=1}^{n}\sum_{j=i}^{n}\binom{n-i}{n-j}(j-1)!(n-j)!(j-i)$$

$$\sum_{i=1}^{n}\sum_{j=i}^{n}(i-1)\binom{n-i}{n-j}(j-1)!(n-j)!(j-i+1)$$

（推的方法就是对于 $i$ 考虑数 $i$ 放在哪个位置，然后 $j$ 是枚举的 $p_i/p_{i-1}$）

两个式子作和：

$$\sum_{i=1}^{n}\sum_{j=i}^{n}(i-1)\binom{n-i}{n-j}(j-1)!(n-j)!(ij-i^2+i-1)$$

以有无 $j$ 分类，分别推出：

$$\sum_{i=1}^{n}\sum_{j=i}^{n}(j-1)!(n-j)!\binom{n-i}{n-j}(-i^2+i-1)=\sum_{i=1}^{n}(-i^2+i-1)(n-i)!(i-1)!\binom{n}{i}$$

$$\sum_{i=1}^{n}\sum_{j=i}^{n}(j-1)!(n-j)!\binom{n-i}{n-j}ij=\sum_{i=1}^{n}i(i)!(n-i)!\binom{n+1}{i+1}$$

最后把组合数用阶乘表示，发现预处理的东西只和 $i$ 有关，直接预处理前缀和即可，时间复杂度 $O(R)$。