Fall with Fake Problem

考虑二分 T 串第一个与 S 串不同的字母的出现位置

check的过程中，我们希望构造出字典序尽量大的串，这样的串 T 从check的位置开始一定是一段下降的序列。考虑实际上第一个与原串不同的字母位置 [1] ，可能的位置至多有 25 种 [2]。假设 T 串中第一个与 S 串不同的字母为 c，则大于 c 的字母的位置已经全部确定了 [3] ，可以枚举 c 的出现次数 [4] ，剩余位置需要用小于 c 的字母填充，这部分可以暴力背包解决，使用bitset加速背包，check的复杂度可以达到 $O(25\sqrt{k}+25\sqrt{k}\frac{n}{64})$ 。事实上枚举 c 的过程是不满的，因为大于 c 的字母的出现次数通常不会是 k 的因数。

构造答案时， T 串前面的部分由之前的二分确定，后面的部分同样进行bitset加速背包，而后通过dp过程输出一组最小解即可。std的做法是枚举第一个不同的字母具体是哪个字母，而后进行与check中相同的dp，根据dp过程输出解。复杂度 $O(25\sqrt{k}\frac{n}{64})$。

单组测试点的复杂度为 $O(25\sqrt{k}\frac{n}{64}\log n)$。

注释：
[1] 例如 `zzzyy` 在 check(2) 时第一个与原串不同的位置至少是第 4 位。
[2] 从当前二分位置起， T 串一定是一段下降的序列，否则可以把后面的字母提前，使得字典序变大。因此至多下降25次，第一个与原串不同的字母位置只可能是下降序列中每种字母第一次出现的位置。
[3] 假如此位置往后还有大于 c 的字母，则可以将后面的字母往前移，得到字典序更大的串。
[4] 出现次数是 k 的因子，因此不太多，在复杂度中记为 $\sqrt{k}$。

Fall with Soldiers

简略版题解可只看加粗文字

首先注意到题目中的关键性质：**串长为奇数**。由于每次我们删除每个1的左右两边的数字，因此总串长奇偶性不变。容易发现，如果有一种操作方案使得最后剩下的全为1，那么我们可以将这个剩下的串删至只剩一个1，而如果我们有一种方案可以将该串删至只剩一个1，这种方案肯定是满足题目条件的，因此我们可以认为题目所求的条件与将串删至只剩一个1的条件是等价的。

现在我们来考虑一个串满足什么样的条件才能够有一种方案删至只剩一个1。容易发现，如果这个串中间位置有一个1，那么我们可以一直操作这个1达成条件。而如果这个串中间位置没有1，我们考虑中间一段0能够被删除的条件。串中每个1最多能操作的次数为其距离最近的边缘的距离，例如0010000最多只能操作两次，有一个非常显然的结论是我们应当选择尽可能靠近中心的1。我们取出最靠近中心的左右两边的两个1，分别考虑操作它能够删除的区间，当两个删除区间覆盖整个串的时候，我们发现此时必然有一种操作方案能够使整个串被删至只剩一个1 [1] ，而如果不能覆盖整个串，必然有一个位于两个1中间的0剩下，而且无论如何操作都无法将其删除 [2]。条件最后总结为：**最靠近中线左右两侧的两个1距离不超过$(n-3)/2$**。

得到了条件之后就可以进行计数，设待填位置总数为tot，分情况讨论：

- 若正中间有1则可任意填数，答案即为2^{tot}
- 若正中间没有1，我们沿中线将区间划分为左右两块，设左区间第i个位置左侧问号的数量为$cntl_i$，在右区间内距离该位置不超过$(n-3)/2$的问号的数量为$cntr_i$，左区间最靠右的1位置为$locl$，右区间最靠左的1位置为$locr$，右区间问号总数为$totr$，**考虑枚举左区间最靠右的1的位置**，则有：
  - 对于$i<locl$，不存在有问号填1后满足条件，因此不产生贡献
  - 对于$i\geq locl$，满足条件的情况为该点必须为1，其右侧全填0 [3] ，并且右区间至少有一个1满足距离限制，讨论$locr$与其位置关系得到两种情况答案为
    - $2^{cntl_i+totr-cntr_i}(2^{cntr_i}-1)=2^{cntl_i+totr}-2^{cntl_i+totr-cntr_i}$
    - 2^{cntl_i+totr}

容易发现，我们需要维护的信息为左区间每个问号位置的2^{cntl_i+totr}与$2^{cntl_i+totr-cntr_i}$，对于$locl$处的答案，还需维护$cntl_i$和$cntr_i$，改变一个位置的字符，将会产生对于一个区间信息乘除2或加减1的一些影响，我们可以简单地使用线段树完成以上信息的维护，并在询问操作的时候使用以上信息分类讨论求得答案。

注释：
[1] 以0010010为例，左右两个1分别删除的范围为DD1DD10和0010D1D，这两个区间重合了，因此我们可以通过操作完成目标，有一个显然的方式是先一直操作其中一个1，直到另一个1位于整个串的中间位置，然后再操作它。由于对其中一个1操作一次会使中间位置向另一个1偏移一位，而两个1的删除区间互相覆盖，意味着另一个1总能达到中间位置，因此该方案一定可行。
[2] 以0010001为例，左右两个1删除范围为DD1DD01和0010001（即为空），中间有一个位置00100X1没有被任何一个1覆盖，若存在其他1，它们的范围也不可能比中间两个大，因此该位置不可能被删掉，我们无法达成目标。
[3] 注意该情况分为当前点为问号和当前点为$locl$两种，但计算方法一样。

Fall with Trees

设第 2 层宽度为 $w_2=d$，则有第 k 层宽度为 $w_k=\sum_{i=2}^{k}d*2^{2-i}=d(2-2^{2-k})$，第 k 层与第 $k+1$ 层之间的面积为 $S_k=h\frac{w_k+w_{k+1}}{2}$。

于是我们要求 $S=\sum_{i=1}^{k-1}S_k=\frac{hd}{2}\sum_{i=1}^{k-1}4-3\times 2^{1-i}$，此部分暴力求和直至精度达标或使用等比数列求和均可。

Link with Balls

将可以取 $0\sim k-1$ 个球的框与只能取 k 的倍数个球的框合并为一个可以取任意个球的框，就得到了 n 个可以取任意个球的框和一个可以取 $0\sim n$ 个球的框。枚举 $0\sim n$ 个球的框中取出了多少个球，剩下的球的选取方案可以由插板法得到，于是答案为 $\sum_{i=0}^{n}C_{m-i+n-1}^{n-1}$，即 $C_{m+n}^{n}-C_{m-1}^{n}$。

Link with EQ

记 $f(x)$ 表示长度为 x 的两端封闭的 [1] 连续空区间中期望坐下的人数，则有：

$$f(x)=f(\lfloor\frac{(x-1)}{2}\rfloor)+f((x-1)-\lfloor\frac{(x-1)}{2}\rfloor)+1$$

$$f(1)=f(2)=0$$

类似的，设 $g(x)$ 表示长度为 x 的半开 [2] 连续空区间中期望坐下的人数，则有：

$$g(x)=\begin{cases}0 & x\leq 1\\ f(x-1)+1 & x\geq 2\end{cases}$$

最后设 $h(x)$ 表示长度为 x 的桌子期望坐下的人数，则有：

$$h(x)=\frac{1}{x}\sum_{i=1}^{x}g(i-1)+g(x-i)+1$$

$$h(x)=1+\frac{2}{x}\sum_{i=1}^{x-1}g(i)$$

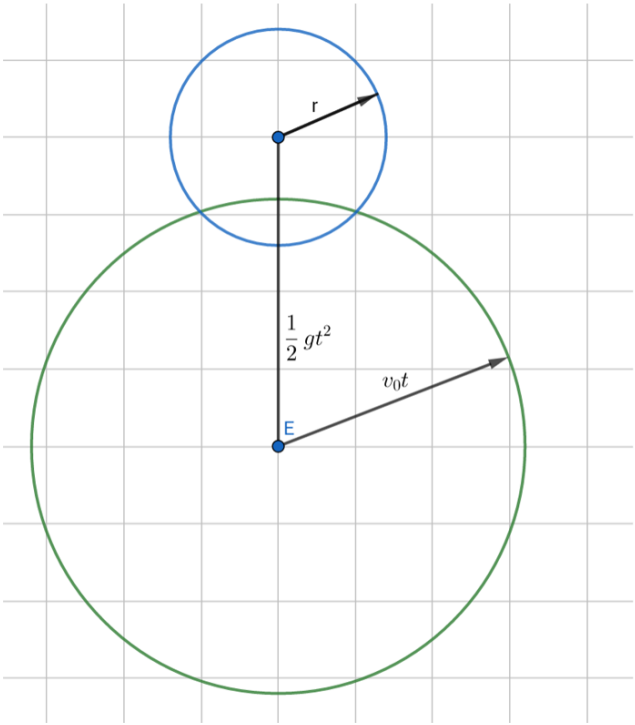
于是可以依次预处理 $f(x)$，$g(x)$，$g(x)$的前缀和，$h(x)$ ，查询时直接输出答案即可。

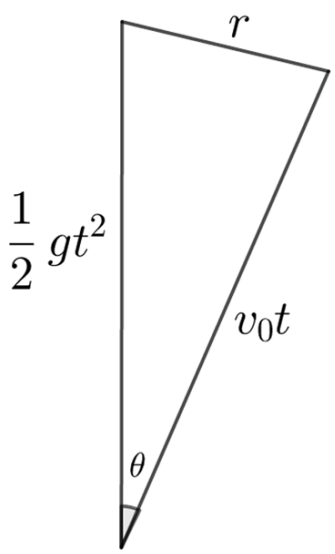
注释：
[1] 两端封闭：即最左空位的左侧和最右空位的右侧都有人了。
[2] 半开：即其中一侧是桌子的尽头，另一侧有人。

Link with Grenade

随机从某球面上取点进行随机和使用角度进行两次积分实质上是一样的，因为这两种方法单位球面的面密度相同。手雷在水平面上的方向并不会影响到其是否炸到人，因此我们只需要关心竖直面。

考虑转换参考系，给人和手雷都加上一个反向的重力加速度，则手雷做匀速直线运动，人做匀加速直线运动。t 秒后，人的位置是固定的，而手雷的位置是一个圆，手雷可以炸到人的范围也是一个圆（如左下图）。作出如右下图所示的三角形，可以用余弦定理求得手雷可以炸到人的仰角 θ 。





根据球冠表面积公式，可以炸到人的部分的表面积为 $2\pi(v_0t)^2(1-\cos\theta)$，球的表面积为 $4\pi(v_0t)^2$，故答案为 $\frac{1-\cos\theta}{2}$，将余弦定理得到的 $\cos\theta$ 的表达式代入即可（注意特判炸不到和一定会炸到的情况）。

Link with Limit

建图，点 x 向点 $f(x)$ 连边，那么 f 迭代的过程实质上是在图上行走的过程。原题实际上问的是每一个环的点权平均值是否相同，使用任意方法找出所有环并求出平均值即可。

Smzzl with Greedy Snake

首先， `f` 的数量显然是相邻两点间的曼哈顿距离，我们只需要减少转弯的次数即可。注意到题目中说相邻两点不在同一行同一列，也就意味着我们需要向两个相邻的方向前进。假如目前的朝向就是需要前进的方向，则直接前进即可；否则只需要旋转一次就可以到达一个需要前进的方向，接下来模拟即可。

Smzzl with Safe Zone

题目要求要在外凸包上找一个点，使得它到内凸包的最近距离最远，可以证明，外凸包上的点取在顶点一定不会更劣（证明见文末）。

故考虑枚举外凸包上的每一点，暴力枚举内凸包的每一条边求解最小距离后取max即可求得答案，此做法单个数据复杂度 $O(n^2)$ 。

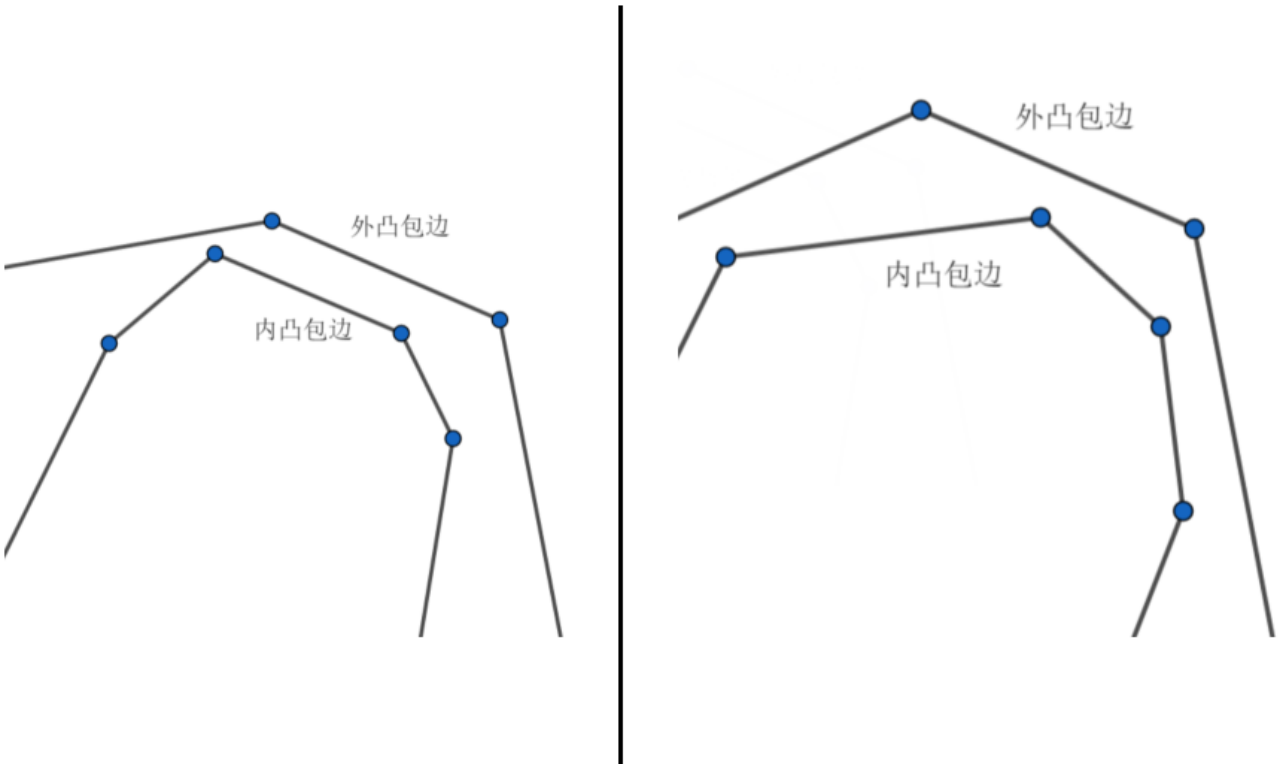
考虑利用旋转卡壳做法优化上述过程，可以发现，在外凸包上逆时针枚举每一个点的时候，与之匹配的内凸包上的最近点也是单调逆时针旋转的，故同旋转卡壳做法，当在外凸包上不断枚举下一个点的时候，不断维护与之对应的最近的内凸包上的边。此做法单个数据复杂度 $O(n)$ 。

以下证明“外凸包上的点取在顶点一定不会更劣”：

考虑若点取在外凸包的边上，

I.如左下图，内凸包上最近边与此边平行，此情况可以发现将外凸包所选点平移至端点不会更劣

II.如右下图，内凸包上最近边与此边不平行，此情况可以发现将外凸包所选点平移至端点更优



综上，外凸包上的点取在顶点一定不会更劣

Smzzl with Tropical Taste

池中冰红茶的体积满足微分方程：

$$\begin{cases} V'(t) = (q-p)V(t) \\ V(0) = 1 \end{cases}$$

解得：

$$V(t)=e^{(q-p)t}$$

我们事实上要判别反常积分 $\int_0^{+\infty}V(t)dt$ 是否收敛，对其求不定积分得：

$$\int V(t)dt=\begin{cases}(q-p)e^{(q-p)t} & q\neq p\\ t & q=p\end{cases}$$

于是当 $q<p$ 时，其收敛；当 $q\geq p$ 时，其发散。

Yiwen with Formula

设和为 x 的子序列有 cnt_x 个，则答案为 $\prod x^{cnt_x}$。考虑如何求 cnt_x，一种简单的方法是记 $f[i][j]$ 表示前 i 个数和为 j 的方案数进行dp，复杂度 $O(n^2)$。考虑优化这个dp，将一个数 a_i 写作多项式 $1+x^{a_i}$，即可用分治NTT优化。注意到多项式的模数其实是 $\varphi(998244353)=998244352$，因此需要任意模数NTT。三模数NTT或不加优化的拆位FFT可能会因常数过大而超时（出题人第一版代码跑的比暴力还慢），可参考2016年毛啸在国家集训队论文中提出的优化方法。

复杂度 $O(n\log^2 n)$。

Yiwen with Sqc

可以发现不同字母之间没有影响，分别考虑每一种字母，不妨考虑 a，假设字母a在原串中出现了cnt次，记字母 a 在原字符串中其出现的第 i 个位置为 a_i （$a_0=0,a_{cnt+1}=n+1$），记 $len_i=a_{i+1}-a_i$ 即两个相邻的a之间的距离。

首先固定左端点于 1，移动右端点，可以发现，答案为 $\sum_{k=1}^{cnt}len_k*k^2$（因为当右端点在相邻两个 a_i 之间移动端时候字母数量不变，故该表达式即每段长度乘以当前端到开头的字母个数）。

将左端点不断右移，记当前位置为 pos，则当 $pos\leq a_1$ 时均有答案为 $\sum_{k=1}^{cnt}len_k*k^2$，故此段对答案贡献为 $len_0*\sum_{k=1}^{cnt}len_k*k^2$。同样的，当 $pos\in(a_1,a_2]$ 时，答案 $\sum_{k=2}^{cnt}len_k*(k-1)^2$。可以发现，对 $pos_i\in(a_i,a_{i+1}]$，有 $\sum_{k=i+1}^{cnt}len_k*(k-i)^2$。

设 $ans_i=\sum_{k=i+1}^{cnt}len_k*(k-i)^2$

（若并不想看严格推导可以自己取几个数差分两次 ans 就能悟到解法）

设 dif_i 为两个相邻两个ans的差分值，则：

$$dif_i=ans_{i+1}-ans_i$$

$$=\sum_{k=i+2}^{cnt}len_k*(k-i-1)^2-\sum_{k=i+1}^{cnt}len_k*(k-i)^2$$

$$= \sum_{k=i+2}^{cnt} len_k * (k-i-1)^2 - \sum_{k=i+2}^{cnt} len_k * (k-i)^2 - len_{i+1} * (1)^2$$

$$= -\sum_{k=i+2}^{cnt} len_k * (2k-2i-1) - len_{i+1}$$

设 $diff_i$ 为相邻两个 dif 的差值，则：

$$diff_i = dif_{i+1} - dif_i$$

$$= -\sum_{k=i+3}^{cnt} len_k * (2k-2(i+1)-1) - len_{i+2} + \sum_{k=i+2}^{cnt} len_k * (2k-2i-1) + len_{i+1}$$

$$= -\sum_{k=i+3}^{cnt} len_k * (2k-2(i+1)-1) + \sum_{k=i+3}^{cnt} len_k * (2k-2i-1) + len_{i+2} * 3 + len_{i+1} - len_{i+2}$$

$$= \sum_{k=i+3}^{cnt} len_k * 2 + len_{i+2} * 3 + len_{i+1} - len_{i+2}$$

由上式可以轻易维护 $diff$ 值，进而维护 dif ，进而维护 ans

上述做法是 $O(n)$ 的，$O(26n)$ 的做法可能会被卡常。