2020 牛客 NOIP 赛前集训营-普及组（第二场）（题解）

T1 GCD

20pt

直接根据题意算即可

40pt

我们发现，将 x 进行素因子分解，如果只能分解出一个素数 y，那么 f(x) = y。否则 f(x) = 1。

所以我们可以枚举每个素数，然后枚举它的次方，复杂度约为 O(nlogn)

100pt

我们还可以通过修改线性筛的模板来预处理每个 x 的 f(x) 值，复杂度 O(n)

T2 包含

我们可以用 dp 进行预处理，dp[i] 代表对于询问的数字为 i 时的答案。(1 代表包含，0 代表不包含)

首先对于输入的所有数字 a[i] 可得 dp[a[i]] = 1。

然后我们倒着枚举值域，对于任意一个数字 x，如果 dp[x] = 1，那么去除其中任意一位二进制上的 1，对应的 dp 值仍为 1

用 O(nlogn) 的时间进行预处理，之后就可以 O(1) 回答每次查询了。详见代码：

```
for(int i = maxn; i >= 0; i--){
    for(int j = 0; j < 20; j++)//枚举去掉哪个位置的 1
        if(dp[i] && (i >> j & 1))
            dp[i-(1<<j)] = 1;
}
```

T3 前缀

30pt

如果 t 中未包含 s 中没有的字符，直接暴力循环匹配。否则就是-1

60pt

不处理高精度大数，用 int 储存出现的整数

100 pt

如果 s 串不循环，并且没有通配符"*"，那么这个问题就是一个经典的序列自动机问题。

序列自动机其实就是一个 next 指针数组。它储存了每个位置跳转到下一个最近的某字符的位置。

举个例子，比如字符串"acdea"中 next[1]['a']就是 5。这表示从第 1 个位置匹配一个最近的'a'是在第 5 个位置。

next 数组的预处理也相当简单，只要倒着 for 一遍记录每个字符最近的位置，然后拷贝到当前位置的 next 数组中即可。

比起用语言表达，可能看代码来的更快。

```
for(int i=n;i;i--){
    for(int j=0;j<26;++j)
        next[i-1][j]=next[i][j];
    next[i-1][s[i]-'a']=i;
}
```

有了这个思路之后我们就魔改序列自动机自己造一台新的自动机。

首先解决"s 串是一个无限循环串"的问题。

为了解决这个问题我们需要把原序列机的首尾相接变成环。

这个首先顺着遍历一遍查找每个字符第一次出现的位置，称为 first。

将 last 和 first 接起来，然后倒着遍历求 next。

然后想办法解决"*"的问题。一般来说自动机都不太能处理通配符，比如 AC 机和后缀机。

也不是说它们不能处理，而是"*"在其他的自动机中一般都代表 26 种出边。不和记忆化搜索配套使用复杂度基本接受不了。

但是序列自动机 next 数组的含义是能够满足匹配的下一个最近的位置，那对于 i 来说就是 i+1 嘛。

接下来想办法解决数字的问题。先假设出现的数字不是大数，比如在 int 范围内。

这个 next 指针说起来也不过就是链表结构。有线性结构的地方，就有倍增。

所以这里将链表结构改造成倍增链表。

如果输入的整数是大数，倍增链表总不能无止境的倍增下去吧。对于这个地方，因为 s 串是一个“循环”串。

这就代表我输入单一字符进行转移的时候，自动机总是成环的。所以预处理循环节，发现对于每个字符，循环节长度就是它在原本的 s 串中出现的次数。

所以这里要手写一个高精度除以单精度，要同时知道商和余数。

注意最后一轮不能跑满，而是倍增过去。

也可以直接算出要跳到哪个字符，记一下每种字符第 i 个在哪，每个字符是这种的第几个，就可以直接算不用倍增

T4 移动

这个题不会输出 -1，因为在最晚 1e9 秒后，所有闸门都将开启，这时候一定可以安全走到终点。一些人没有得满分的原因就是在某些测试点中输出了 -1。

20pt

我们可以离散化之后进行动态规划，$dp[i][j]$ 代表 i 时刻到达 j 位置是否可行(用 0/1 表示)。注意，此时的 i 不代表第 i 秒，而是离散化后的第 i 个时刻，离散化时只关注每个闸门开，关的时间，即我们会从 n 个信息提取出 2n 个时刻。

$dp[i][j]$ 一定是从 $dp[k][j-1]$ 转移过来的，我们只需要枚举 k，然后判断是否可以转移即可。但是这样复杂度是 $O(n^3)$。

40pt

使用前缀和优化状态转移，即可优化到 $O(n^2)$

100 pt

我们还是将时间离散化，记录每一道闸门开启的时间段以及对应的闸门编号，总共大约有 n+m 个时间段。f[i] 表示第 i 段时间内牛牛最早能在第几秒到达这一位置。(“这一位置”指的是此时间段对应的闸门编号)

转移就是牛牛往左/右走一步可以走到哪些闸门开启的时间段（就是可以先等一会再走）

用类似最短路的方法来实现 dp 的枚举。即向优先队列中塞入一个结点 Node{id,x,time} id 为时间段编号，x 为闸门编号，time 为当前时间。优先队列按照 time 排序，每次选出一个 time 最小的 Node ，向它周围的两个闸门进行转移。转移时找到对应闸门开启的时间，要同时满足当前闸门与被转移的闸门都开启才可以转移。